SKRIPSI

IMPLEMENTASI AGEN SELF-PLAY DI LUDII

Bryan Heryanto

NPM: 6181801031

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2022

UNDERGRADUATE THESIS

IMPLEMENTATION OF SELF-PLAY AGENT IN LUDII

Bryan Heryanto

NPM: 6181801031

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2022

LEMBAR PENGESAHAN

IMPLEMENTASI AGEN SELF-PLAY DI LUDII

Bryan Heryanto

NPM: 6181801031

Bandung, 30 Juni 2022

Menyetujui,

Pembimbing

Digitally signed by Lionov

Lionov, Ph.D.

Ketua Tim Penguji

Digitally signed by Keenan Adiwijaya Leman

Keenan Adiwijaya Leman, M.T.

Anggota Tim Penguji

Digitally signed by Husnul Hakim

Husnul Hakim, M.T.

Mengetahui,

Ketua Program Studi

Digitally signed by Mariskha Tri Adithia

Mariskha Tri Adithia, P.D.Eng

PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

IMPLEMENTASI AGEN SELF-PLAY DI LUDII

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,
Tanggal 30 Juni 2022

Bryan Heryanto
NPM: 6181801031

ABSTRAK

Gim adalah sebuah bentuk dari aktivitas bermain yang dilakukan oleh pemain dengan mengikuti serangkaian aturan untuk mencapai tujuan spesifik tertentu. Pada umumnya, pemain dari gim adalah manusia, tetapi gim juga dapat dimainkan oleh entitas lainya berupa sistem komputer yang dapat melakukan aktivitasnya secara otonom. Dalam bidang *Artificial Intelligence*(AI), sistem tersebut dikenal dengan sebutan agen sebutan *Game-playing agent* (GPA). Contoh GPA yang menunjukkan performa tinggi ketika bermain gim adalah DeepBlue (spesifik bermain pada permainan catur) dan AlphaGo (Go). Terdapat beberapa cara yang dapat digunakan untuk mengembangkan lebih lanjut GPA menjadi *General Game-Playing(GGP) agent* yaitu agen yang dapat digunakan untuk bermain pada lebih dari satu gim, contohnya adalah AlphaZero yaitu agen yang dapat bermain di gim Go, Shogi, dan Catur.

Pengembangan terhadap *GGP agent* juga berjalan beriringan dengan pengembangan terhadap general *game-playing system*, yaitu sistem yang dapat memodelkan lebih dari satu gim. Salah satu *general game-playing system* yang dapat digunakan untuk bermain pada beberapa gim adalah Ludii. Pada sistem Ludii sudah terdapat gim yang dimodelkan menggunakan *class grammar* dan dapat dimainkan oleh agen yang dikembangkan menggunakan bahasa pemrograman Java. Pada penelitian ini dibuat *GGP agent* yang dapat bermain pada lebih dari satu gim di Ludii. Untuk membuat agen tersebut digunakan algoritma yang terinspirasi dari AlphaZero yaitu *Monte Carlo Tree Search* (*MCTS*) dengan *neural network* yang dilatih secara *self-play*.

Walaupun begitu, agar agen dapat berjalan di Ludii, diperlukan beberapa penyesuaian dan modifikasi. Contohnya, digunakan *wrapper* yaitu sebuah cara untuk melakukan konversi antara fitur yang disediakan oleh Ludii dalam bentuk objek dan *tensor* untuk melakukan pelatihan terhadap *neural network*. Selanjutnya, dilakukan eksperimen untuk agen cerdas yang sudah dibangun menggunakan tiga gim yaitu Tic-Tac-Toe, Hex, dan Reversi. Untuk memperkecil *resource* yang dibutuhkan pada eksperimen, ukuran papan dari Hex dan Reversi yang dipakai telah diperkecil. Kemudian, agen dilatih menggunakan dua model *neural network* dan dua jenis *wrapper*. Agen yang telah dilatih kemudian diuji dengan bermain gim melawan agen yang mengimplementasikan algoritma berbeda.

Berdasarkan hasil dari pertandingan, didapatkan bahwa agen yang dibuat sudah cukup berhasil berlatih dan bermain di ketiga gim yang diambil. Pada Tic-Tac-Toe, agen dapat memberikan hasil seimbang pada 100 dari 100 pertandingan ketika bertanding melawan agen dengan algoritma optimal yaitu *alpha-beta pruning*. Sedangkan pada Hex 4×4, agen memenangkan seluruh pertandingan ketika bermain sebagai pemain pertama.

Kata-kata kunci: *Artificial Intelligence*, gim, *game-playing agent*, *general game-playing agent*, *self-play*, Ludii

ABSTRACT

A game is a form of playing that follows a series of rules for the player(s) to achieve specific goals. In general, humans usually are the player in the game. However, other entities like an autonomous computer system can also be a player in the game. In Artificial Intelligence(AI), that system is called a Game-playing Agent (GPA). Examples of GPAs with high performance when playing against humans are DeepBlue (specific to play only chess) and AlphaGo (Go). There are techniques to develop a GPA further into a General Game-playing (GGP) agent, an agent that can play more than one game. An example of the GGP agent is AlphaZero which can play Go, Shogi, and Chess.

The development of GGP agents has also gone in parallel with the development of a general game-playing system, a system with the capability to model more than one game. One example of such a system is Ludii. In Ludii, a game can be modeled using a class grammar and played by an agent developed using Java Programming Languages. In this research, we develop a GGP agent that can play more than one game in the Ludii systems. This agent will implement algorithms that are inspired by the AlphaZero: Monte Carlo Tree Search (MCTS) and neural networks trained by self-play.

However, to enable the agent to run in Ludii, we must make several adjustments and modifications to the agent. For example, we must use a wrapper to allow the conversion between the object that models the game features given by Ludii and the tensor needed to train the neural network. Subsequently, experiments are performed on the GGP agent using three games: Tic-Tac-Toe, Hex, and Reversi. To reduce the resources needed for experiments, we shrink the board size of Hex and Reversi. Afterward, we use two neural network models and two types of wrappers to train the agent. Finally, the trained agent is tested by playing games against agents that implement different kinds of algorithms.

The results show that our agent has been successfully trained and can play the three chosen games with moderate success. In Tic-Tac-Toe, our agent can achieve drawing results in all 100 games against agents with an optimal algorithm such as alpha-beta pruning. Whereas in 4x4 Hex, our agent wins all 100 games when playing as the first player.

Keywords: Artificial Intelligence, game, game-playing agent, general game-playing agent, self-play, Ludii

Dipersembahkan untuk kedua orang tua dan adik saya

KATA PENGANTAR

Segala puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa atas berkat-Nya, sehingga penulis mampu menyelesaikan skripsi dengan judul "Implementasi Agen Self-Play di Ludii" dengan tepat waktu. Penyelesaian skripsi ini, dapat terlaksana karena dukungan-dukungan yang telah diberikan berbagai pihak terhadap penulis. Oleh karena itu, secara tulus penulis ingin mengucapkan terima kasih yang tak ternilai kepada:

1. Kedua orang tua penulis, Bapak Dadang Heryanto dan Ibu Mimi Mariani Tanuwijaya, dan adik penulis, Jessica Britney Heryanto yang tak henti-hentinya memberikan dukungan, semangat, dan doa untuk menyelesaikan skripsi ini.
2. Bapak Lionov, Ph.D. sebagai dosen pembimbing yang telah meluangkan banyak waktunya untuk memberikan arahan dalam bentuk ide, saran, dan kritik selama pembuatan skripsi ini.
3. Bapak Keenan Adiwijaya Leman, M.T. dan Bapak Husnul Hakim, M.T. sebagai dosen penguji yang telah memberikan saran dan kritik yang membangun, sehingga menjadikan skripsi ini lebih baik.
4. Ibu Joanna Helga, M,Sc. dan Bapak Irvan Jahja, Ph.D. sebagai pelatih kompetisi pemrograman yang dengan tulus membimbing dan membagikan ilmu kepada penulis sehingga dapat berpartisipasi dalam lomba pemrograman.
5. Seluruh dosen-dosen Universitas Katolik Parahyangan, khususnya Ibu Mariskha Tri Adithia, P.D.Eng, Ibu Luciana Abednego, M.T., dan Bapak Kristopher David Harjono, M.T.. yang berperan besar dalam proses belajar mengajar penulis.
6. Teman - teman seperjuangan dalam tim kompetisi pemrograman, Jiang Han dan Warren Mazmur yang menemani, berbagi ide, dan bekerja sama selama berpartisipasi dalam berbagai macam perlombaan.
7. Teman - teman kelompok belajar, Deddy Chandra dan Chris Ardiansyah yang telah saling mengingatkan dan mendukung proses belajar di Universitas Katolik Parahyangan.
8. Teman-teman dan sahabat-sahabat penulis yang telah memberikan dukungan kepada penulis yaitu Surya, Albert, Reinardus, Jasen, Stefani, Stella, Valencia, Christella, dan Sherly.
9. Seluruh pihak yang tidak disebutkan dan baik secara langung maupun tidak langsung memberikan dukungannya kepada penulis.

Penulis berharap skripsi ini dapat berkontribusi untuk meningkatkan wawasan pembaca khususnya dalam bidang pengembangan Artificial Intelligence dalam domain Gim dan memicu pengembangan selanjutnya. Akhir kata, penulis mengucapkan terima kasih dan mohon maaf apabila ada keslahan dan kekurangan.

Bandung, Juni 2022

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

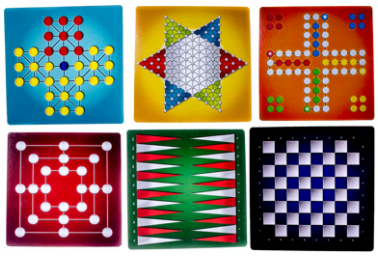
DAFTAR KODE PROGRAM

BAB 1

PENDAHULUAN

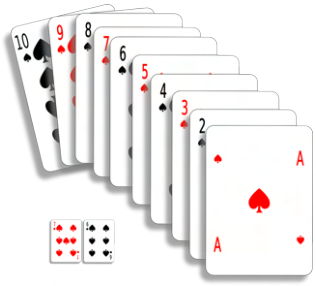
1.1 Latar Belakang

Gim adalah bentuk aktivitas yang dilakukan dengan bermain mengikuti aturan tertentu. Pada dasarnya, gim bisa dibagi menjadi dua jenis yaitu gim non-kooperatif dan gim kooperatif. Pada gim non-kooperatif subjek yang bermain atau sering disebut *player* akan berusaha sendiri untuk memaksimalkan hasil dari *payoff function* [1]. Secara sederhana, *payoff function* adalah fungsi yang mendeskripsikan penghargaan yang diberikan kepada suatu *player* ketika gim telah selesai. Biasanya, semakin besar nilai *payoff function* yang didapatkan, semakin besar juga keuntungan yang didapatkan *player* setelah gim tersebut selesai. Sedangkan, pada gim kooperatif *player* akan bekerja sama untuk mencapai suatu tujuan tertentu [2].



(a) Berbagai jenis papan gim

(b) Berbagai jenis dadu



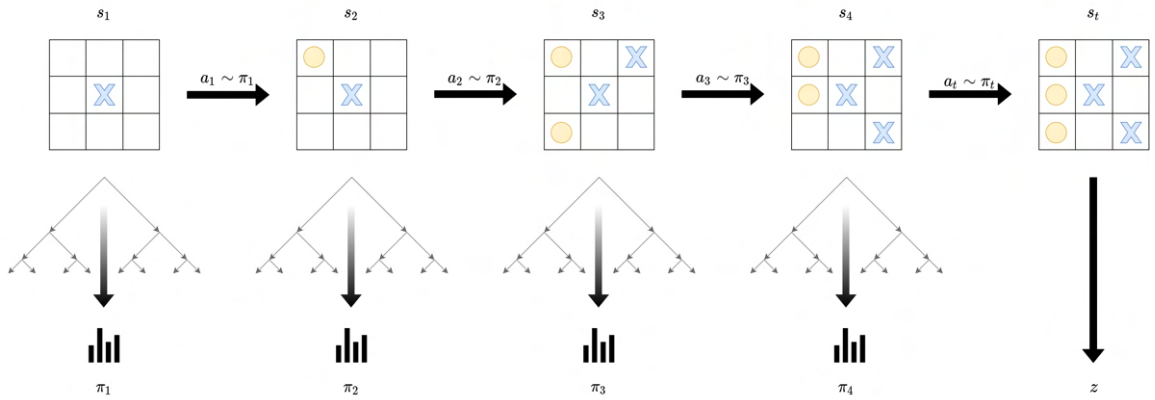
(c) Kartu Remi

Gambar 1.1: Media-media yang digunakan untuk bermain gim strategi tradisional.

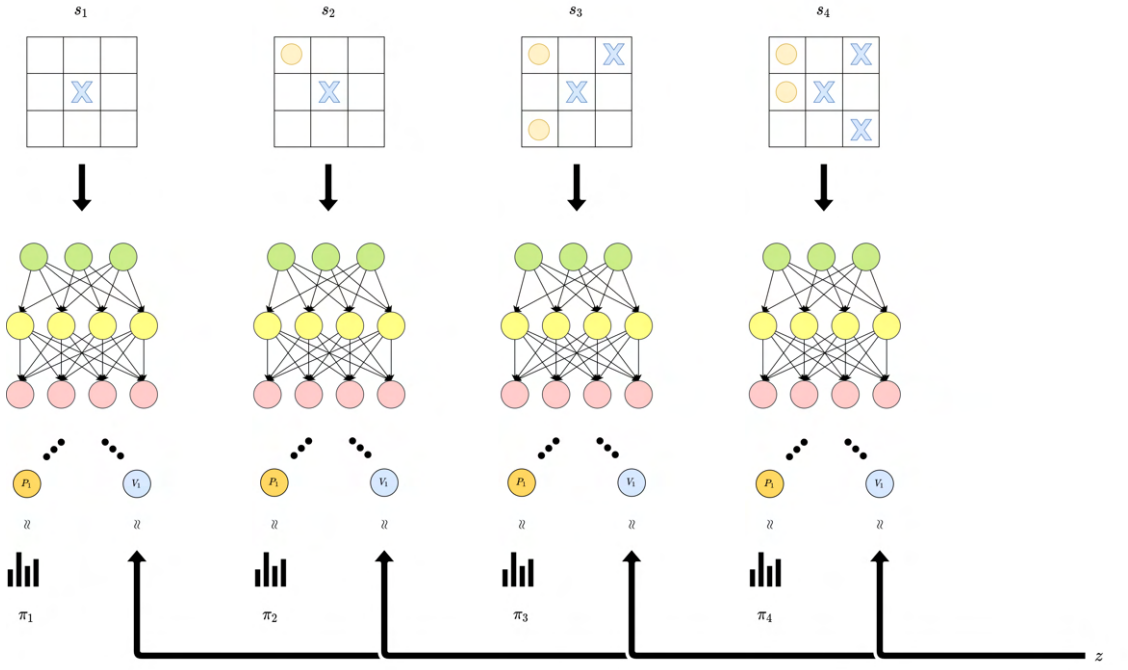
Dilihat dari sejarahnya, gim sudah mengalami banyak pengembangan dari berbagai aspek seperti jenis, aturan, dan media yang digunakan. Pengembangan tersebut mengakibatkan terdapat perbedaan yang sangat mencolok terutama dari segi media yang digunakan pada gim yang dikembangkan sebelum adanya gawai elektronik dengan gim yang dikembangkan setelah adanya gawai elektronik. Sebelum adanya gawai elektronik gim dimainkan menggunakan media fisik tertentu meliputi papan, dadu, dan kartu seperti yang ditunjukan Gambar 1.1a[1], 1.1b[2], dan 1.1c[3]. Sedangkan setelah adanya gawai elektronik, gim dapat dimodelkan oleh komputer sehingga dapat dimainkan secara digital. Gim yang dimainkan dengan menggunakan media fisik berupa papan, dadu, kartu, dll dapat dikategorikan sebagai gim strategi tradisional [3].

Komputer adalah gawai elektronik yang paling banyak digunakan manusia untuk bermain gim. Salah satu alasan komputer banyak digunakan untuk bermain gim adalah kemampuan permodelan yang dimilikinya. Kemampuan itu menyebabkan dengan cara tertentu berbagai variasi gim dapat dimodelkan dan disimpan untuk dapat dimainkan oleh *player* yang nantinya berinteraksi dengan komputer tersebut melalui *interface*. Bahkan, komputer juga dapat digunakan untuk memodelkan gim yang bersifat tradisional sehingga dapat dimainkan secara digital.

[1] https://pixabay.com/illustrations/backgammon-background-board-chance-313388/
[2] https://upload.wikimedia.org/wikipedia/commons/c/c8/Wuerfel5.jpg
[3] https://pixabay.com/vectors/card-deck-deck-of-cards-poker-cards-155284/



(a) Ilustrasi ketika algoritma Monte Carlo Tree Search (MCTS) dijalankan



(b) Ilustrasi proses pelatihan dengan *Deep Learning*

Gambar 1.2: Ilustrasi keseluruhan algoritma dijalankan pada gim Tic-Tac-Toe

Player adalah istilah yang merujuk pada subjek yang bermain dalam suatu gim dan mengikuti aturan tertentu. Gim yang dimainkan oleh satu *player* disebut *single-player game*, sedangkan gim yang dimainkan lebih dari satu *player* disebut *multi-player game*. Terdapat juga *multi-player game* yang spesifik dimainkan oleh dua orang disebut sebagai *two-player game*. Perlu diperhatikan, *player* yang bermain dalam gim tertentu tidak berarti selalu manusia. Dalam bermain pada suatu gim di komputer, manusia dapat diwakilkan dengan suatu subjek lainnya yang sering disebut sebagai agen.

Agen adalah sistem komputer yang disituasikan dalam suatu *environment* dan memiliki kemampuan untuk melakukan *autonomous action* tertentu untuk mencapai tujuan *design*-nya [4]. Dari pernyataan tersebut bisa diambil dua atribut penting dalam pengembangan agen yaitu *action* dan *environment*. *Action* adalah tindakan yang dilakukan oleh agen dengan menggunakan *effector* untuk mencapai *goal* tertentu dalam suatu *environment*. Sedangkan *environment* adalah lingkungan di mana agen tersebut berada. Untuk mendapatkan informasi kondisi dari *environment* atau biasa disebut sebagai *state*, agen harus dapat melakukan *percept* yaitu mendapatkan *state* pada *environment* di waktu tertentu dengan menggunakan *sensor*.

Interaksi dilakukan agen dengan cara memetakan *percept* ke *action* menggunakan *agent function* tertentu. Secara umum, agen tersebut dapat disebut sebagai agen cerdas (*intelligent agent*) apabila memiliki *agent function* yang memungkinkan agen untuk mengambil *action* terbaik dalam situasi tertentu [5]. Agar dapat melakukan aksi "terbaik", agen tersebut memerlukan suatu kecerdasan(*intelligence*). Definisi kecerdasan ini sudah banyak diteliti, salah satunya yang dilakukan oleh Alan Turing melalui *Turing Test* [6]. Untuk menyimpulkan, suatu hal dapat disebut cerdas(*intelligent*) apabila dapat berpikir dan melakukan *action* secara rasional. Agen yang rasional atau disebut *rational agent* dapat didefinisikan sebagai agen yang melakukan *action* untuk mendapat hasil terbaik atau hasil yang diperkirakan terbaik apabila terdapat *uncertainty* [5].

Artifficial Intelligence(*AI*) adalah bidang yang mempelajari pembuatan agen cerdas. Bidang tersebut sudah banyak dikembangkan untuk diaplikasikan ke berbagai domain seperti kesehatan [7], transportasi [8], dan *trading* [9]. Salah satu aplikasi *AI* yang menarik banyak perhatian adalah aplikasinya dalam bidang gim [10, 11, 12, 13, 14, 15]. Dalam bidang gim peneliti *AI* berusaha mengembangkan agen cerdas yang dapat bermain dan mengalahkan manusia. Agen yang dapat bermain suatu gim tanpa bantuan langsung dari manusia disebut sebagai *game-playing agent*.

Sudah dibuktikan bahwa *game-playing agent* dapat bermain dan mengalahkan manusia pada gim spesifik seperti catur, Go, dan sebagainya. Pada tahun 1997, Deep Blue telah berhasil membuat *game-playing agent* yang dapat memainkan catur dan mengalahkan Garry Kasparov (juara dunia catur) dalam 4 dari 6 pertandingan gim catur [10]. Lalu, pada 2016 AlphaGo yang menggunakan *Monte Carlo Tree Search*(*MCTS*) dan *Deep Learning* berhasil mengalahkan Lee Sendol (pemain Go proffesional dengan *rank* 9 dan) dalam 4 dari 5 pertandingan gim Go 19×19 [11]. Kedua pertandingan antaragen cerdas dan manusia tersebut membuktikan bahwa kecerdasan yang diberikan kepada agen memberikannya kemampuan untuk menyelesaikan masalah pada *state* permasalahan yang kompleks (masalah bisa disebut kompleks apabila memiliki ukuran *state spaces* yang besar) khususnya dalam bermain dalam suatu gim dengan tingkat kemampuan yang sama atau bahkan melebihi manusia.

Apabila diperhatikan *game-playing agent* yang digunakan oleh *Deep Blue* [10] dan AlphaGo [11] dirancang sedemikian rupa sehingga bekerja hanya pada gim spesifik. Sehingga, untuk dapat bekerja pada gim lain-nya manusia harus merancang ulang agen tersebut dan memberikannya kecerdasan berdasarkan strategi tertentu. Berbeda dengan manusia, manusia memiliki kemampuan untuk mencari strategi untuk memenangkan suatu gim dengan cara memainkan-nya secara terus-menerus. Proses mencari cara untuk memenangkan sebuah gim tersebut disebut *learning* (belajar). Dengan adanya *machine learning*, proses belajar itu dapat juga dilakukan oleh komputer. Dari pernyataan tersebut muncul ide bahwa dengan cara memberikan agen kemampuan untuk belajar dapat dibuat agen yang dapat bermain tidak hanya pada satu jenis gim spesifik atau sering disebut disebut sebagai *General Game-Playing* (*GGP*) [16].

Proses belajar dapat dilakukan apabila suatu subjek terlebih dahulu diberikan informasi dasar. Informasi dasar yang didapatkan oleh manusia pada saat bermain gim adalah peraturan. Peraturan tersebut bervariatif tergantung dari fitur gim yang dimainkan misalnya papan, jumlah bidak, cara pergerakan bidak, dan sebagainya. Pada kebanyakan gim strategi tradisional peraturan yang digunakan hanya dimodifikasi sedikit dari gim strategi tradisional lainnya sehingga membuat pola sifat tertentu. Contohnya pada gim catur dan *checker*, keduanya dimainkan di papan dengan ukuran 8×8 berwarna hitam dan putih, namun pergerakan dan jenis bidak yang digunakan berbeda. Dari contoh itu, diketahui bahwa pola peraturan tersebut dapat dikelompokan. Berdasarkan pengelompokan itu, sistem yang dapat memodelkan lebih dari satu gim strategi tradisional secara *general* dapat dibuat. Sistem ini diperlukan sebelum membuat *general game-playing agent*.

Terdapat beberapa sistem yang sudah ada dan dapat menunjang permodelan lebih dari satu gim secara *general*, antara lain Stanford General Game Players, Regular Board Game(RBG), dan Ludii [14, 17]. Pada skripsi ini, Ludii dipilih sebagai sistem permodelan *general game* yang dipakai. Ludii diambil karena mempunyai banyak keunggulan dibanding sistem lainnya yang dapat dilihat dalam *paper* yang ditulis oleh Éric Piette dkk [13]. Salah satu perbedaan paling mencoloknya adalah kemampuannya untuk memberikan informasi mengenai gim secara *object*

oriented. Dengan paradigma *object oriented* informasi dari gim seperti *state* dan *action* dapat dimodelkan menggunakan kelas dan objek. Selain itu, paradigma *object-oriented* memungkinkan permasalahan untuk dapat dipecah ke bentuk permasalahan yang lebih kecil secara terstruktur, sehingga penyelesaian masalah dapat dilakukan secara lebih mudah.

Éric Piette dkk [13, 14] menjelaskan bahwa Ludii adalah sebuah *general game system* yang merupakan hasil dari projek penelitian yang didanai oleh ERC yaitu Digital Ludeme Project(DLP)[4] yang dilakukan di Maastricht University. DLP bertujuan untuk memodelkan 1000 gim strategi tradisional dunia agar dapat dimainkan di dalam satu *database*. Untuk merealisasikan itu, dibuatlah Ludi General Description Langguage (L-GDL) yaitu *Game Description Langguage* yang dipakai oleh Ludii dengan menggunakan pendekatan *class grammar*. Dengan adanya L-GDL, Ludii menjadi mampu memodelkan berbagai macam jenis gim dengan sifat dan peraturan yang berbeda-beda.

Game Description Langguage adalah bahasa yang digunakan khusus oleh *general game system* tertentu untuk melakukan pemodelan gim. Pada Ludii, L-GDL digunakan untuk mendeskripsikan gim dengan struktur hierarki dalam format yang mirip dengan LISP [17]. L-GDL sudah menggunakan *class grammar*, sehingga pembuat gim tidak perlu mengetahui implementasi pembuatan gim secara detail [17]. Melainkan, untuk melakukan pemodelan gim, pembuat gim hanya perlu menggunakan contructor dari hirarki class yang disediakan oleh Ludii atau disebut dengan *grammar* [14, 17]. Kemudian pemodelan dari gim dalam bentuk *grammar* akan dikompilasi oleh Ludii dengan melakukan pemetaan 1:1 dengan *source code* dalam bentuk Java [14].

Ludii menyediakan sebuah perangkat lunak dalam bentuk `.jar` yaitu Ludii Player yang dapat diunduh di halaman situs resminya yaitu Ludii Portal[5]. Ludii Player ini merupakan perangkat lunak yang harus diunduh untuk mengakses sistem dari Ludii. Perangkat ini dapat dipakai untuk bermain gim, memodelkan gim baru dengan menggunakan *game description language*-nya, dan membuat *custom agent*. Apabila ingin langsung bermain tanpa melakukan kustomisasi, Ludii Player ini juga sudah mengimplementasikan berbagai jenis gim dan agen yang dapat dipakai untuk bermain. Pada skripsi ini, hal yang akan dikembangkan hanyalah pengembangan pada *custom agent*, sedangkan gim yang dipakai untuk dimainkan oleh *custom agent* tersebut dipilih dari gim yang sudah disediakan Ludii di Ludii Player.

Selain dibutuhkan sistem untuk memodelkan gim secara *general*, agen juga harus dirancang sedemikian rupa sehingga dapat belajar untuk bermain gim bedasarkan informasi yang diberikan sistem tersebut. Untuk itu, sudah terdapat beberapa penelitian yang meneliti rancangan agen yang dapat digunakan untuk bermain pada lebih dari satu gim. Salah satunya pada tahun 2017, David Silver dkk [12] (*team* dari Deep Mind) mengeluarkan *preprint* yang mengenalkan AlphaZero yaitu versi lebih *general* dari AlphaGo. Pada penelitian tersebut agen cerdas dirancang sedemikian rupa sehingga dapat bermain dalam lebih dari satu gim yaitu catur, shogi, dan Go. Agen AlphaZero mengkombinasikan *MCTS* dan *machine learning* untuk bermain pada salah satu dari ketiga yang didukung-nya dengan hanya diperlukan satu rancangan agen yang sama. Dengan demikian manusia tidak perlu merancang ulang agen baru apabila ingin dibuat agen yang dapat digunakan untuk bermain pada gim lainnya.

Learning atau bisa disebut juga pelatihan merupakan salah satu hal yang dapat dilakukan oleh agen apabila sudah memiliki suatu informasi dasar dari suatu domain. Dalam konteks pengembangan agen, *learning* dapat didefinisikan sebagai cara agen untuk meningkatkan performa pekerjaan di masa yang akan mendatang setelah membuat observasi dari suatu *environment* [5]. Berbagai jenis cara dalam bidang *Machine Learning* telah dikembangkan untuk meningkatkan performa dari agen seperti *Deep Learning*, *reinforcement learning*, *supervised learning*, dan *unsupervised learning*[18]. Untuk membangun agen cerdas, cara *learning* tersebut dapat digabungkan atau ditambahkan dengan algoritma lainnya seperti contohnya *search algorithm* [11, 12].

[4] http://www.ludeme.eu/
[5] https://ludii.games/

Menurut Aurélien Géron [18] ada dua jenis *learning* yang dapat dilakukan yaitu *batch learning* dan *online learning*. Pada *batch learning* atau disebut juga *offline learning* sistem tidak dapat melakukan *learning* secara *incremental* sehingga harus dilatih menggunakan seluruh data yang ada. Berbeda dengan pada *online learning*, data diberikan secara *sequential* satu per satu atau dikelompokan menjadi sebuah grup yang dikenal sebagai *mini-batches*. Secara keseluruhan *learning* yang dipakai pada skripsi ini berjenis *online learning*.

Menurut Miguel Morales [19] *Artificial Neural Network*(*ANN*) atau biasa disebut *neural network* adalah *multi-layered non-linear function approximators* yang terinspirasi dari jaringan saraf pada otak binatang. *ANN* bukanlah algoritma melainkan struktur yang disusun atas lebih dari satu *layer* tranformasi matematika yang dilakukan kepada *input*. *Neural network* dapat dilatih menggunakan *Deep Learning* dan dipasangkan dengan algoritma lainnya untuk menyelesaikan suatu masalah. Dalam mendukung *General Game-Playing*, biasanya *neural network* dipasangkan dengan *search algorithm* salah yang paling banyak digunakan adalah *MCTS* (*Monte Carlo Tree Search*).

Maciej Świechowski dkk [20] memberikan penjelasan tentang *Monte Carlo Tree Search* (*MCTS*) sebagai *decision-making algorithm* yang digunakan untuk mencari dalam *combinatorial spaces* yang besar dengan menggunakan representasi *tree*. *MCTS* merupakan algoritma yang sering digunakan dalam domain gim. Hal itu disebabkan oleh algoritma *MCTS* tidak membutuhkan *heuristic* yang didefinisikan oleh manusia, sehingga untuk memakai *MCTS* hanya diperlukan peraturan dari gim yang ingin dimainkan. Selain itu algoritma *MCTS* juga bisa menjaga keseimbangan antara eksploitasi dan eksplorasi dengan menggunakan metode tertentu.

MCTS dapat dibagi menjadi empat tahap yaitu *selection*, *expansion*, *simulation*, dan *backpropagation*. Pada tahap *selection*, algoritma *MCTS* akan mencari dan kemudian memilih *node* pada *tree* sesuai dengan *selection policy*. Kemudian, kecuali berada di *terminal state expansion* akan dilakukan dengan menambah *node* baru setelah berada di *action* terakhir pada tahap *selection*. Setelah itu dilakukanlah *simulation* yaitu melakukan simulasi dari suatu gim atau masalah hingga mencapai *terminal state*. Terakhir, ketika mencapai *terminal state* dilakukan *update* pada *statistic* dari *tree* dengan melakukan *backpropagation* pada suatu nilai.

Pada *MCTS* standar atau *Monte Carlo Tree Search* dengan *Upper Confidence bounds applied to Trees* (*MCTS/UCT*) *selection policy* dapat dihitung menggunakan Upper Confidence Bound (*UCB*). David Silver dkk [12] menyatakan bahwa *alpha-beta pruning* atau sering disingkat *alpha-beta* menjadi algoritma paling kuat untuk bermain catur selama 4 dekade, *MCTS* standar ini terbukti tidak dapat mengalahkan algoritma *alpha-beta*. Namun, algoritma *alpha-beta* ini memerlukan bantuan dari manusia untuk membuat *heuristic*-nya. Karena setelah dicoba menggunakan *neural network*, tidak terbukti bahwa *neural network* dapat menemukan fungsi evaluasi yang lebih baik dari heuristic buatan manusia untuk pencarian *heuristic* dalam *alpha-beta pruning*. Pada akhirnya, setelah *MCTS* digabungkan dengan *neural network* hasil yang diberikan menjadi lebih baik, bahkan dapat menyaingi algoritma *alpha-beta prunning* dalam bermain di gim catur.

David Silver dkk [12] menunjukan bahwa *MCTS* dapat digabungkan dengan *neural network* yang memiliki dua head yaitu *policy head* dan *value head*. Pada *MCTS* yang digabungkan dengan *neural network*, persamaan *UCB* akan dimodifikasi dengan menambahkan nilai prediksi probabilitas dilakukanya suatu action $P(s, a)$. Nilai prediksi probabilitas *action* tersebut akan diambil dari nilai pada *policy head* yang diprediksi menggunakan *neural network* untuk setiap *state* yang telah direpresentasikan pada tahap *selection*. Sedangkan *value head* digunakan pada tahap *simulation* untuk memprediksi *reward* V(s) dari *state* yang ingin di *expand*.

Selain memberikan cara untuk menggabungkan *MCTS* dan *neural network*, David Silver dkk [12] juga memberikan cara untuk melatih agen tersebut dengan cara *self-play*. *Self-play* pada dasarnya diklasifikasikan sebuah algoritma *reinforcement learning*. *Reinforcement learning* adalah pelatihan yang dilakukan dengan cara mencoba melakukan sejumlah aksi untuk menemukan aksi dengan *reward* terbesar [21]. Menggunakan *reinforcement learning*, agen yang dilatih secara *self-play* dapat mempelajari cara bermain satu gim hanya melalui aturan dari gim seperti posisi bidak, ukuran papan, dll.

Gambar 1.2 menunjukkan secara garis besar proses yang dilakukan pada pelatihan dengan *self-play*. Pada *self-play*, agen belajar dengan melakukan pertandingan melawan dirinya sendiri. Pertama, agen yang menggunakan algoritma MCTS dengan *neural network* yang memiliki parameter sama akan bertanding. Kemudian ketika melakukan pertandingan dengan algoritma tersebut, didapatkan sekumpulan data *state* s, beserta sekumpulan data target berupa probabilitas dilakukannya suatu *action* $\pi(s, a)$ dan reward z. Setelah itu untuk menambah performa dari agen dengan memberikan s kepada *neural network* untuk dilatih secara *Deep Learning* dengan cara membandingkan nilai prediksi probabilitas *action* $P(s, a)$ dan prediksi *reward* $V(s)$ dengan data target $\pi(s, a)$ dan z.

Cara yang dipakai oleh penelitian dari David Silver dkk [12] terbukti cukup efektif sehingga cara tersebut menjadi salah satu cara yang paling banyak ditemukan apabila membahas *General Game-Playing*. Salah satunya, penelitian tersebut dibahas ulang oleh Alvaro Gunawan dkk [22] yang melanjutkan penelitian tersebut dengan mengubah arsitektur *neural network* menjadi lebih sederhana dan menyesuaikannya degan sistem *general game-playing* yang dipakainya yaitu GDL. Pada penelitian Alvaro Gunawan dkk [22] *state* yang menjadi *input neural network* berasal dari *fluents* (predikat yang bernilai benar) yang nilainya didapatkan dengan menggunakan *propositional neural network* dari *game description langguage* GDL. Karena informasi pada *game description langguage* yang dipakai sistem Ludii dikonversi menjadi objek, maka *propositional neural network* tidak diperlukan. Pada Ludii sudah disediakan *wrapper* yang dapat digunakan untuk mendapatkan *state* yang pada suatu gim sebagaimana ditulis oleh Dennis Soemers dkk [23].

Pada skripsi ini dibuat *General Game-Playing agent* yang dapat bermain di dalam beberapa gim yang telah disediakan oleh Ludii tanpa bantuan dari manusia (hanya diberikan aturan dari permainan). Sebagai batasan sifat dari gim diambil dari aturan pada kompetisi Ludii pada COG 2021[6]. Pada kompetisi tersebut, agen yang dibuat harus dirancang sedemikian rupa sehingga dapat bermain pada gim yang bersifat *two-player*, *zero-sum*, *adversarial*, *turn-based*, *sequential*, *deterministic*, dan *fully-observable*. Batasan pada sifat ini membuat gim yang dipilih hanyalah gim yang dapat dimainkan oleh dua agen yang saling bertanding dengan secara bergiliran melakukan *action*. Selain itu, karena sifat gim deterministic dan *fully observable* maka gim tidak melibatkan *random* (seperti pemutaran dadu, pengambilan kartu tersembunyi, dll) serta informasi dari *state* selalu diberikan secara utuh.

Berdasarkan batasan yang diberikan, sebagai perwakilan diambil tiga gim yang memenuhi sifat tersebut yaitu Tic-Tac-Toe, Hex, dan Reversi. Ketiga gim ini sudah disediakan di Ludii Player sehingga pemodelan terhadap gim tidak perlu dilakukan terlebih dahulu. Selain itu, aturan dari gim seperti ukuran papan, konfigurasi peletakan bidak awal, dan diperbolehkannya melakukan *swap move* dapat diatur dengan cara tertentu. Pengaturan ini membuat ukuran gim Reversi dan Hex yang digunakan di skripsi ini lebih kecil ketimbang ukuran normal, yaitu 4×4 sampai 6×6 untuk Hex dan 6×6 untuk Reversi.

Metode yang skripsi ini pakai untuk membuat *general game-playing agent* adalah *MCTS* digabungkan dengan *neural network* yang terinspirasi dari penelitian David Silver dkk [12]. Namun, untuk menyesuaikannya dengan sistem Ludii untuk mendapatkan *input* dari *network* akan digunakan cara pendekatan yang dilakukan oleh Dennis Soemers dkk [23]. Selain itu, karena gim yang dipakai memiliki jumlah *state* yang relatif lebih sedikit ketimbang AlphaZero maka arsitektur *neural network* akan dibuat lebih sederhana sebagaimana yang ditulis oleh Alvaro Gunawan dkk [22]. Kemudian, akan digunakan juga arsitektur *convolutional neural network* yang terinspirasi dari VGG-16 [24]. Setelah selesai dirancang akan dilakukan pelatihan pada agen tersebut secara *self-play*.

Setelah agen berhasil dilatih, pengujian kemudian dilakukan dengan melangsungkan beberapa pertandingan antara *general game-playing agent* dengan agen yang memilih *action* menggunakan algoritma lainnya di ketiga gim yang dipilih (Tic-Tac-Toe, Hex, dan Reversi). Dari pertandingan, evaluasi tingkat kecerdasan suatu agen di gim dilakukan dengan menganalisis jumlah menang, seri, dan kalah. Fokus utama skripsi ini adalah membangun agen yang dapat belajar untuk bermain secara rasional, memiliki sifat *general*, serta dapat dilatih menggunakan *self-play* menggunakan

[6] https://ieee-cog.org/2021/index.html#competitions_section

Ludii. Sehingga, pengujian di skripsi ini akan berfokus pada eksplorasi dari model dan rancangan agen. Eksplorasi tersebut dilakukan dengan melakukan pengujian terhadap *general game-playing agent* menggunakan dua jenis *wrapper* dan dua jenis *model* yang berbeda. Selain itu, untuk menyesuaikan penelitian dengan *resource hardware* yang dipakai, ketika melakukan pengujian cakupan dari masalah dikurangi dengan mengecilkan *state space* yang digunakan pada gim dengan memakai ukuran papan yang lebih kecil dari masing-masing gim (kecuali Tic-Tac-Toe).

1.2 Rumusan Masalah

Berikut beberapa masalah yang dibahas pada skripsi ini:

1. Bagaimana membuat *general game-playing agent* yang menggunakan *neural network* untuk memperkirakan *action* terbaik agar dapat digunakan untuk bermain beberapa gim di Ludii Player?
2. Seperti apa model *neural network* yang baik, agar dapat dipakai oleh *game-playing agent* untuk bermain beberapa permainan pada Ludii?
3. Bagaimana melakukan pelatihan pada agen agar agen dapat memperkirakan *action* optimal pada suatu *state* gim?

1.3 Tujuan

Tujuan yang ingin dicapai oleh skripsi ini adalah:

1. Mempelajari 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.
2. 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.
3. Melakukan pelatihan pada *neural network* secara *online* dengan mendapatkan data dari *MCTS*. Kemudian memperbaharui *neural network* dengan cara menandingkan agen dengan cara *self-play*.

1.4 Batasan Masalah

Berikut batasan-batasan masalah yang diberikan dalam skripsi ini:

1. Gim yang dipakai adalah Tic-Tac-Toe, Hex, dan Reversi yang memiliki sifat *two-player*, *zero-sum*, *adversarial*, *turn-based*, *sequentia*l, *deterministic*, dan *fully-observable*.
2. Kustomisasi yang dilakukan pada Ludii hanyalah dilakukan pada *custom-agent*, penelitian ini menggunakan gim yang telah tersedia di Ludii Player.
3. Evaluasi dilakukan dengan melakukan pertandingan menggunakan *custom agent* yang diimplementasikan dan *agen* yang disediakan Ludii.

1.5 Metodologi

Berikut langkah-langkah yang dilakukan untuk mencapai tujuan dari skripsi ini:

1. Mempelajari informasi mengenai sistem Ludii dengan cara membaca dokumentasi Ludii dan mengevaluasi gim-gim yang disediakan di Ludii Player.
2. Mempelajari cara untuk membuat agen sederhana untuk digunakan di Ludii Player.
3. Melakukan studi literatur mengenai algoritma-algoritma yang dapat digunakan untuk membuat *general game-playing agent*.
4. Menganalisis cara Ludii melakukan permodelan *state* dan *action* untuk suatu gim.
5. Mencoba mendapatkan model *action* dan *state* dari API yang disediakan oleh Ludii.

6. Mempelajari arsitektur *neural network* yang dapat digunakan untuk memperkirakan *action* optimal pada suatu *state* di Ludii.
7. Mempelajari cara untuk menggabungkan algoritma yang dipilih pada studi literatur yaitu *Monte Carlo Tree Search* (*MCTS*) dengan *neural network* untuk dijalankan di Ludii.
8. Mempelajari mekanisme dan *library* yang dapat digunakan untuk melakukan *Deep Learning* (pelatihan *neural network*) di GPU menggunakan bahasa pemrograman Java.
9. Mengimplementasikan agen *MCTS/UCT*.
10. Mengintegrasikan seluruh *library* yang digunakan dengan lingkungan atau *platform* pelatihan.
11. Memodifikasi tahap *selection* dan *simulation* pada tahap *MCTS/UCT* sehingga MCTS dapat memilih *action* dan memprediksi *reward* meggunakan *neural network*
12. Mengimplementasikan perangkat lunak yang dapat digunakan untuk melakukan pelatihan agen dengan menggunakan *self-play*.
13. Melakukan pelatihan dengan *self-play*.
14. Melakukan modifikasi pada *library* agar agen dapat berjalan di Ludii Player.
15. Melakukan pengujian dan eksperimen terhadap dengan menandingkan *game-playing agent* yang sudah tersedia di Ludii dengan *game-playing agent* yang sudah diimplementasikan.
16. Menulis dokumen skripsi

1.6 Sistematika Pembahasan

Pembahasan penelitian yang dilakukan oleh skripsi ini akan dibagi ke dalam 7 Bab yaitu:

- **Bab 1 Pendahuluan**
  Pada Bab 1 dibahas latar belakang yang berisi definisi gim, jenis-jenis gim, agen, *game-playing agent*, penelitian-penelitian *game-playing agent* sebelumnya, beberapa general *game-playing system*, deskripsi sistem Ludii, beberapa cara yang dapat digunakan untuk membuat *general game-playing agent*, dan beberapa gim yang dipakai dalam skripsi. Selain itu pada bab ini juga dibahas rumusan masalah, tujuan, batasan masalah, dan metodologi yang digunakan untuk mencapai tujuan skripsi.
- **Bab 2 Dasar Teori**
  Pada Bab 2 dibahas teori berkaitan dengan gim khususnya gim strategi tradisional, sifat-sifat gim, agen dan hubungannya dengan *general game-playing agent*, detail mengenai algoritma *MCTS* beserta persamaan *UCB* yang dipakai, *neural network*, cara pelatihan dengan *self-play*, detail Ludii Player, serta cara pengembangan agen di Ludii.
- **Bab 3 Analisis**
  Pada Bab 3 dijelaskan analisis terhadap *task environment* dan definisi *general* yang dipakai oleh agen. Kemudian, diberikan analisis terhadap pelatihan menggunakan *self-play*. Di akhir, dibahas analisis terhadap cara beserta sistem yang dibutuhkan oleh agen untuk dapat berintegrasi dengan Ludii dan sistem pelatihan yang digunakan.
- **Bab 4 Analisis dan Perancangan Perangkat Lunak**
  Pada Bab 4 diberikan rancangan dan analisis dalam bentuk *use case diagram*, diagram sekuens, diagram kelas, dan *flowchart*. Use case diagram digunakan untuk membahas gambaran keseluruhan sistem yang dibuat, beserta entitas-entitas dan *library* yang berperan dalam pengujian dan pelatihan agen. Sedangkan, diagram sekuens dan diagram kelas digunakan untuk meninjau terlebih secara *hi level* kelas-kelas yang digunakan oleh sistem beserta interaksinya ketika digunakan dalam pelatihan dan pengujian. *Flowchart* digunakan untuk memeberi detail pada proses *MCTS* dan pelatihan *self-play*. Selain itu, dibahas juga rancangan data struktur yang digunakan *MCTS*.
- **Bab 5 Implementasi**
  Pada Bab 5 diberikan penjelasan secara lebih detail tentang modifikasi yang perlu dilakukan di *library*. Setelah itu juga diberikan secara lebih detail tentang kode-kode implementasi *neural network* menggunakan DJL, kode-kode yang digunakan pada pelatihan, dan beberapa modifikasi yang perlu ditangani saat mengimplementasikan *MCTS*.

- **Bab 6 Eksperimen**
  Pada Bab 6 dijabarkan hasil eksperimen pada gim Tic-Tac-Toe, Hex dengan ukuran papan 4×4 sampai 7×7, dan Reversi 6×6 . Bab ini diawali dengan penjelasan secara detail lingkungan pelatihan, lingkungan pengujian, *wrapper*, serta model *neural network* yang digunakan. Setelah itu dicantumkan data hasil eksperimen beserta analisis dari hasil yang didapatkan.
- **Bab 7 Kesimpulan dan saran**
  Pada Bab 7 diberikan kesimpulan dan saran yang didapatkan dari penelitian yang dilakukan di skripsi ini. Di sini diberikan juga beberapa saran yang dapat dipertimbangkan sebagai panduan untuk meningkatkan performa di penelitian selanjutnya.