Pemanfaatan Asynchronous Advantage Actor-Critic Dalam Pembuatan AI Game Bot Pada Game Arcade

Evan Kusuma Susanto, *Teknik Informatika Sekolah Tinggi Teknik Surabaya*, Yosi Kristian, S.Kom., M.Kom., *Teknik Informatika Sekolah Tinggi Teknik Surabaya*

***Abstrak*— Asynchronous Advantage Actor-Critic (A3C) adalah sebuah algoritma deep reinforcement learning yang dikembangkan oleh Google DeepMind. Algoritma ini dapat digunakan untuk menciptakan sebuah arsitektur artificial intelligence yang dapat menguasai berbagai jenis game yang berbeda melalui *trial and error* dengan mempelajari tempilan layar game dan skor yang diperoleh dari hasil tindakannya tanpa campur tangan manusia. Sebuah network A3C terdiri dari Convolutional Neural Network (CNN) di bagian depan, Long Short-Term Memory Network (LSTM) di tengah, dan sebuah Actor-Critic network di bagian belakang. CNN berguna sebagai perangkum dari citra output layar dengan mengekstrak fitur-fitur yang penting yang terdapat pada layar. LSTM berguna sebagai pengingat keadaan game sebelumnya. Actor-Critic Network berguna untuk menentukan tindakan terbaik untuk dilakukan ketika dihadapkan dengan suatu kondisi tertentu. Dari hasil percobaan yang dilakukan, metode ini cukup efektif dan dapat mengalahkan pemain pemula dalam memainkan 5 game yang digunakan sebagai bahan uji coba.**

***Kata Kunci*—Artificial Intelligence, Deep Reinforcement Learning, Machine Learning, Reinforcement Learning.**

I. PENDAHULUAN

Perkembangan game yang pesat beserta perkembangan teknologi komputer yang juga pesat mendorong peneliti untuk merancang sebuah sistem yang dapat memainkan permainan seperti layaknya manusia [1]–[4]. Pengembangan sistem cerdas ini disebut *artificial intelligence* (AI). Tetapi, biasanya artificial intelligence yang dirancang tersebut biasanya hanya dapat memainkan 1 jenis permainan saja, yaitu permainan yang telah ditentukan oleh pembuatnya. Belum ada AI yang cukup fleksibel untuk memainkan berbagai macam jenis game yang berbeda, tetapi metode pengembangan AI yang berkembang pesat tidak menutup kemungkinan untuk kemunculan AI yang fleksibel.

Metode pengenalan citra dan ekstraksi fitur yang sedang berkembang adalah metode *deep learning*. Deep learning merupakan salah satu arsitektur *neural network* yang memungkinkan input dari suatu neural network berupa gambar, bukan fitur seperti neural network umumnya. Tidak seperti *shallow learning* yang perlu mengekstraksi fitur secara manual, dan melakukan preprocessing yang banyak. Deep learning mampu mencari dan melatih fitur sendiri, dengan input *raw image*. Karena fitur juga ikut dilatih, maka memungkinkan didapatkannya fitur yang mungkin terlewat jika didesain manual [5].

Metode yang digunakan agar AI dapat belajar sendiri melalui trial and error adalah metode *reinforcement learning* [6]. Dalam penggunaannya, teknik dari reinforcement learning yang akan digunakan adalah *Actor-Critic* [7]. Actor-Critic menyediakan model matematika yang dapat digunakan untuk membantu menentukan langkah apa yang terbaik yang harus diambil oleh AI pada kondisi yang dihadapi.

Dengan adanya kedua metode tersebut, secara teori dengan waktu yang cukup lama dan data yang cukup besar maka dapat didesain sebuah arsitektur neural network yang dapat memainkan berbagai macam jenis game yang berbeda dan dapat belajar dengan sendirinya melalui trial and error.

II. REINFORCEMENT LEARNING

Reinforcement learning adalah mempelajari apa yang harus dilakukan (memetakan situasi ke aksi) untuk memaksimalkan reward yang diperoleh [6]. Pada reinforcement learning, komputer tidak diberi "supervisi" atau diberi panduan oleh user secara langsung. Komputer akan dihadapkan dengan sebuah environment dan belajar dari hasil interaksi antara komputer dengan environment tersebut. User tidak memberi panduan berupa jawaban yang diharapkan dari komputer secara langsung, tetapi user membiarkan komputer berinteraksi dengan environment dan akan memberi reward (imbalan) kepada komputer ketika komputer berhasil melakukan sesuatu hal yang baik dan memberi punishment (hukuman) ketika komputer melakukan suatu hal yang buruk.

Dasar dari algoritma-algoritma reinforcement learning berasal dari Markov Decision Process (MDP) [8]. Markov Decision Process merupakan sebuah formalisasi dari sequential decision making (pengambilan keputusan berurutan), di mana keputusan yang diambil akan mempengaruhi situasi dan reward berikutnya. Dalam MDP dibentuk sebuah model matematika ideal dari sebuah permasalahan reinforcement learning sehingga dapat dibuat pernyataan teoritis terhadap permasalahan terebut. Dalam MDP terdapat beberapa konsep yang digunakan dalam reinforcement learning yaitu state, transition, action, reward, dan policy. Selain itu digunakan pula formula untuk

3 Oktober 2018 Sekolah Tinggi Teknik Surabaya

Evan Kusuma S, Departemen Teknik Informatika, Sekolah Tinggi Teknik Surabaya, Surabaya, Jawa Timur, Indonesia (e-mail: evanks@rocketmail.com)

Yosi Kristian, Dosen Teknik Informatika, Sekolah Tinggi Teknik Surabaya, Surabaya, Jawa Timur, Indonesia (e-mail: yosi@stts.edu)

mengkalkulasi ekspektasi reward maksimum yang dapat diperoleh. Formula ini adalah Persamaan Bellman (Bellman Equation) [8] yang dirumuskan sebagai berikut:

$$V(s) = \max_a (R(s,a) + \gamma \Sigma_{s'} T(s,a,s') V(s')) \quad (1)$$

Dalam reinforcement learning terdapat 3 jenis metode dalam menentukan policy terbaik. Metode pertama adalah value based learning. Dalam metode yang bersifat value based lebih diutamakan kalkulasi atau estimasi dari value function setiap state dalam environment. Kalkulasi value function dilakukan dengan menggunakan Persamaan Bellman. Dari value function ini akan dibentuk sebuah policy yang berorientasi untuk memaksimalkan value function yang diperoleh. Salah satu penerapan algoritma value based learning adalah TD-gammon [1], yaitu penggunaan TD learning untuk memainkan permainan backgammon. TD-gammon menjadi dasar bagi banyak penelitian reinforcement learning berikutnya.

Metode berikutnya adalah policy based learning. Pada policy based learning, policy dicari dan ditentukan secara langsung. Policy direpresentasikan sebagai kumpulan parameter (parameterized policy) yang akan dikalkulasi dan dimodifikasi sesuai dengan performa agen (total skor yang diterima) ketika berinteraksi dengan environment dengan policy yang diberikan. Akan dikalkulasi kombinasi parameter terbaik agar skor tertinggi dapat diperoleh. Beberapa cara yang dapat digunakan adalah dengan policy search, algoritma evolutionary [9], atau dengan policy gradient [10].

Metode policy gradient adalah salah satu teknik reinforcement learning di mana dilakukan optimasi terhadap parameterized policy terhadap ekspektasi reward dengan menggunakan gradien. Gradien digunakan untuk mengubah nilai parameter untuk memaksimalkan nilai ekspektasi skor maksimum dan dikalkulasi dengan cara mencari turunan dari cost function (ekspektasi reward maksimum) terhadap masing-masing parameter policy. Contoh dari pemanfaatan metode policy gradient adalah algoritma REINFORCE [11] dan penentuan policy untuk dapat membuat sebuah robot berjalan cepat [12].

III. Actor-Critic

Actor-critic merupakan jenis algoritma reinforcement learning yang merupakan gabungan dari metode policy based learning dan value based learning. Actor-critic terdiri dari 2 komponen, yaitu actor dan critic [7]. Actor adalah komponen yang membentuk policy. sedangkan critic adalah komponen yang memberikan evaluasi terhadap policy yang dibuat oleh actor.

Dalam metode yang bersifat value based lebih diutamakan kalkulasi atau estimasi dari value function (atau Q function) setiap state dalam environment. Kalkulasi value function dilakukan dengan menggunakan Persamaan Bellman. Dari value function ini akan dibentuk sebuah policy yang berorientasi untuk memaksimalkan value function yang diperoleh.

Pada policy based learning policy dicari dan ditentukan secara langsung. Policy akan dikalkulasi dan dimodifikasi sesuai dengan performa agen (total skor yang diterima) ketika berinteraksi dengan environment dengan policy yang diberikan.

Metode policy based learning memiliki kelemahan berupa tidak adanya proses belajar dari pengalaman. Hal ini disebabkan karena pada setiap iterasi dilakukan perubahan nilai parameter berdasarkan nilai gradien pada iterasi tersebut, tanpa adanya pengaruh akumulasi dan konsolidasi dari pengalaman masa lalu. Sedangkan metode value based learning yang diutamakan adalah didapatkannya estimasi value function yang akurat dengan harapan policy yang baik dapat ditentukan dari hasil kalkulasi value function. Hal ini menyebabkan tidak adanya jaminan bahwa policy yang dihasilkan adalah policy yang paling optimal [7].

Metode actor-critic dirancang untuk menggabungkan kelebihan dari metode policy based learning dan value based learning [7]. Pada critic digunakan sebuah function approximation untuk estimasi nilai value function sebuah state. Hasil estimasi ini digunakan untuk update parameter policy pada actor menuju arah yang lebih baik. Dengan metode ini dapat dijamin tercapainya local optimum pada hasil policy (karena digunakannya metode gradien). Selain itu, kondisi optimal dapat tercapai lebih cepat karena digunakannya value function.

Terdapat berbagai macam jenis actor critic yang dapat dibentuk, mengingat banyaknya jenis metode estimasi value function yang dapat dilakukan, seperti metode TD learning [13] atau Q learning [14]. Salah satu varian dari metode actor-critic adalah advantage actor critic. Pada metode ini digunakan nilai advantage function dari sebuah aksi yang dilakukan pada sebuah state.

Advantage menyatakan seberapa baik sebuah tindakan apabila dilakukan pada suatu keadaan dibandingkan dengan aksi lain yang dapat dilakukan. Salah satu metode yang dapat digunakan untuk estimasi nilai advantage function adalah metode Generalized Advantage Estimation [15]. Dari metode ini dihasilkan estimasi advantage function dengan pengaturan bias dan variance yang diinginkan.

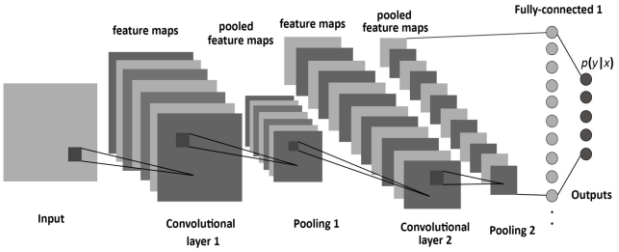
IV. Convolutional Neural Network

Neural Network adalah salah satu algoritma dalam Machine Learning yang terinspirasi dari cara kerja otak manusia untuk menyelesaikan suatu masalah. Neural network tersusun dari beberapa layer artificial neuron yang saling terhubung satu dengan yang lain (fully connected) [16]. Neural network menjadi dasar pembentukan arsitektur lain yang lebih kompleks dan didesain secara khusus untuk dapat mengatasi permasalahan tertentu. Neural network biasa digunakan untuk menyelesaikan masalah yang menggunakan input berupa feature vector (matriks 1 dimensi), tetapi kurang baik untuk menerima dan mengolah input berbentuk gambar (matriks 2 atau 3 dimensi) dan input yang bersifat sequential (hasil saat ini dipengaruhi hasil di waktu lampau).

Deep Convolutional Neural Network (DCNN) adalah pengembangan dari neural network untuk permasalahan yang berhubungan dengan citra. Arsitektur dari DCNN ini sendiri didesain sedemikian sehingga menyerupai susunan korteks visual pada kucing dan monyet untuk dapat

mengolah input gambar berupa apa yang dilihat.

Misalnya sebuah neural network perlu memproses gambar dengan ukuran 80 x 80 pixel dengan 3 channel warna (RGB). Pada hidden layer pertama dalam neural network biasa setidaknya terdapat 80 * 80 * 3 = 19200 bobot untuk sebuah neuron pada layer tersebut. Padahal untuk membuat network yang dapat menyelesaikan permasalahan kompleks dibutuhkan lebih dari sebuah neuron dan sebuah hidden layer. Konsep fully connectivity dalam kasus ini menggunakan terlalu banyak bobot yang dapat menyebabkan terjadinya overfitting, yaitu sebuah keadaan dimana sebuah neural network terlalu menyesuaikan diri dengan data yang dijumpai saat training, sehingga apabila diuji dengan data baru maka neural network tidak akan bekerja dengan baik. Selain itu, informasi struktur spatial dari sebuah gambar akan hilang karena setiap neuron tidak memiliki hubungan antara satu dengan yang lain.



Gambar. 1. Arsitektur Convolutional Neural Network Sederhana

Ide dari CNN sudah muncul pada tahun 1970, namun ide tersebut baru diterbitkan pada sebuah paper pada tahun 1998. Paper tersebut dibuat oleh Yann Lecun, Leon Bottou, Yoshua Bengio, dan Patrick Haffne dengan judul “Gradient-based learning applied to document recognition” yang kemudian menjadi dasar bagi perkembangan CNN modern [17].

Operasi convolution adalah operasi utama dalam DCNN. Pada operasi ini dilakukan ekstraksi fitur laten dari kumpulan pixel input 2 dimensi. Tidak seperti fully connected layer, struktur spatial pada gambar akan tetap dipertahankan oleh convolution layer. Selain itu, jumlah weight yang digunakan juga jauh lebih sedikit dengan adanya weight sharing. Hasil dari proses convolution disebut sebagai feature map atau activation map. Sebuah filter pada operasi convolution dapat mendeteksi sebuah fitur yang terdapat pada sebuah gambar, misal sebuah garis lurus, garis miring, atau bentuk sederhana lain [18].

V. Long Short Term Memory Network

LSTM merupakan salah satu arsitektur Recurrent Neural Network (RNN) [19], yaitu salah satu pengembangan dari neural network biasa yang didesain untuk menangani input yang bersifat bersambung. Yang dimaksud dengan input bersambung adalah output tidak hanya ditentukan oleh input yang baru saja diberikan tetapi juga bergantung pada input yang telah diberikan sebelumnya. Contoh dari input sequential adalah penerjemahan bahasa, di mana arti dari sebuah kata dapat dipengaruhi oleh konteks dari kata sebelumnya. Oleh sebab itu, RNN didesain agar output dari hasil terakhir dipertimbangkan menjadi input untuk hasil selanjutnya.

Terdapat 2 masalah yang dimiliki oleh RNN, yaitu exploding gradient dan vanishing gradient. Kedua permasalahan ini ditemui tidak hanya pada RNN tetapi pada neural network yang dalam (memiliki banyak layer). RNN dapat dianggap memiliki kedalaman yang sangat besar karena kedalaman RNN ditentukan oleh banyaknya data dalam sequence yang hendak diproses. Dalam kasus seperti penerjemahan sebuah kalimat, input yang diberikan adalah setiap kata dalam kalimat tersebut yang jumlahnya cukup banyak.

Exploding gradient terjadi ketika nilai weight dari network bernilai lebih besar dari 1 dan ukuran network sangat dalam. Seperti yang dapat dilihat pada persamaan backpropagation, nilai dari error akan dipropagasikan ke belakang. Salah satu operasi yang dilakukan adalah nilai error dikalikan dengan weight pada layer sebelumnya. Hal ini menyebabkan dalam proses backpropagation untuk setiap timestep nilai error selalu diperbesar sehingga gradien hasil akhir sangat besar. Nilai gradien yang terlalu besar menyebabkan nilai weight yang diupdate juga besar sehingga network tidak dapat konvergen.

Vanishing gradient adalah kebalikan dari exploding gradient, dimana nilai dari weight network kurang dari 1 dan ukuran network sangat dalam. Hal ini menyebabkan nilai error diperkecil setiap timestep sehingga gradien lama kelamaan akan menghilang. Karena nilai gradien hilang, nilai update bobot untuk layer di depan akan menjadi kecil sehingga network tidak dapat dilatih dengan baik [20].

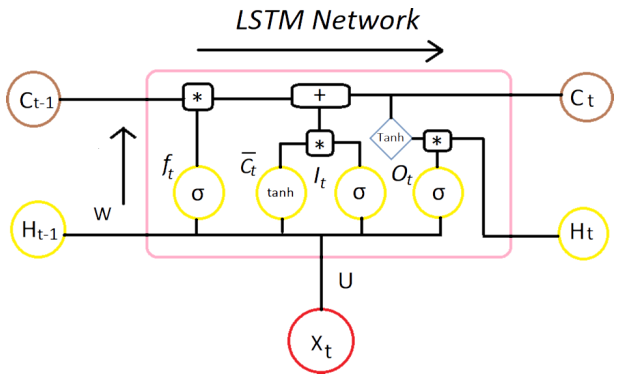
Untuk mengatasi masalah ini, dirancanglah pengembangan dari RNN yaitu Long Short Term Memory network (LSTM) [21]. LSTM pertama kali ditemukan oleh Sepp Hochreitern Jürgen Schmidhuber pada tahun 1997. LSTM juga dan disempurnakan oleh banyak orang seperti Felix Gers, Fred Cummins, Santiago Fernandez, Justin Bayer, Daan Wierstra, Julian Togelius, Faustian Gomez, Matteo Gagliolo, dan Alex Graves.

LSTM tersusun dari sebuah cell state yang bersifat sebagai memori untuk LSTM. Dengan adanya memori ini, permasalahan long term dependency (hilangnya efek input lama terhadap training) yang dimiliki oleh RNN dapat diatasi. Nilai dari memory cell pada LSTM diatur oleh 3 gating unit berbeda, yaitu forget gate, input gate, dan output gate.

VI. Deep Q Network

Deep reinforcement learning merupakan penggabungan dari algoritma reinforcement learning dan algoritma deep learning. Salah satu kegunaan dari deep learning adalah dapat dibentuk menjadi sebuah arsitektur general artificial intelligence [3]. Implementasi dari algoritma reinforcement learning terbatas pada permasalahan-permasalahan yang sederhana. Untuk permasalahan yang kompleks digunakan kumpulan fitur-fitur penting yang ditentukan oleh pembuat program sehingga permasalahan kompleks tersebut dapat disederhanakan. Algoritma deep learning dapat digunakan untuk memproses input yang bersifat kompleks, seperti citra atau data berurutan [18], [21]. Dengan deep learning dapat

dilakukan ekstraksi dan deteksi fitur-fitur penting dari sebuah input dengan dimensionalitas tinggi [3].



Gambar. 2. Struktur LSTM Sederhana

Terdapat beberapa tantangan dalam proses penggabungan antara deep learning dan reinforcement learning. Tantangan pertama adalah perbedaan metode training yang dilakukan, di mana pada deep learning digunakan data dengan jumlah yang sangat banyak dan telah diberi label oleh manusia. Pada reinforcement learning training dilakukan berdasarkan reward yang diterima, di mana reward ini bersifat jarang, terdapat noise, serta terdapat delay antar reward. Permasalahan lainnya adalah pada deep learning data yang digunakan tidak berhubungan antara satu dengan yang lain, sedangkan pada reinforcement learning data yang masuk bersifat berurutan dan memiliki hubungan yang erat antar data. Selain itu data yang diterima pada reinforcement learning akan berubah seiring perubahan policy, di mana pada deep learning distribusi data yang diterima selalu tetap [3].

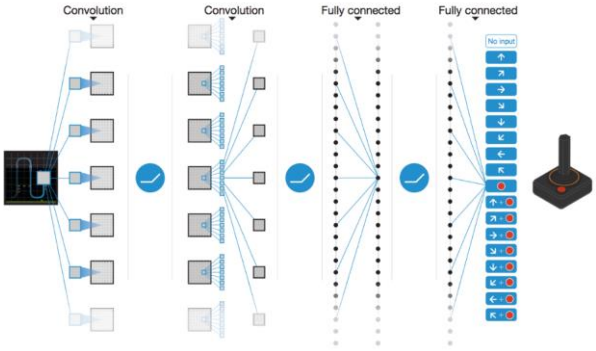
Deep Q Network adalah sebuah arsitektur deep reinforcement learning yang pertama kali dikembangkan. Deep Q learning dikembangkan oleh DeepMind pada tahun 2013. Dari arsitektur ini dihasilkan sebuah AI yang berhasil mengalahkan nilai AI lain dalam 43 game Atari 2600 [22].

Secara garis besar, Deep Q Network adalah sebuah convolutional neural network yang dilatih menggunakan metode Q learning agar ketika diberi sebuah state (citra yang menggambarkan keadaan game sekarang), network akan menghasilkan estimasi Q function untuk masing-masing aksi yang dapat dilakukan. Aksi yang akan dipilih adalah aksi dengan nilai Q function yang paling tinggi [3]. Metode Q Learning sendiri dapat diformulasikan sebagai berikut:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(R + \gamma \max_{a'} Q(s',a') - Q(s,a)) \quad (2)$$

Untuk menghilangkan korelasi antar data training pada DQN digunakan metode experience replay [23]. Dengan experience replay, pengalaman dari agen tidak langsung dijadikan data training tetapi akan disimpan dalam sebuah replay memory. Pada saat training akan diambil sebuah batch dari experience secara random dari replay memory sebagai data training.

Untuk mengatasi nilai target Q function yang berubah-ubah setiap perubahan policy, digunakan separate target network [22]. Digunakan 2 set parameter berbeda untuk nilai target dan untuk proses training. Parameter dari target network diubah setelah beberapa step tertentu, dimana jumlah step tersebut adalah angka yang cukup besar, sedangkan network kedua selalu diubah tiap epoch pada training. Dengan target network, perubahan nilai target Q function dapat dikurangi.



Gambar. 3. Arsitektur DQN

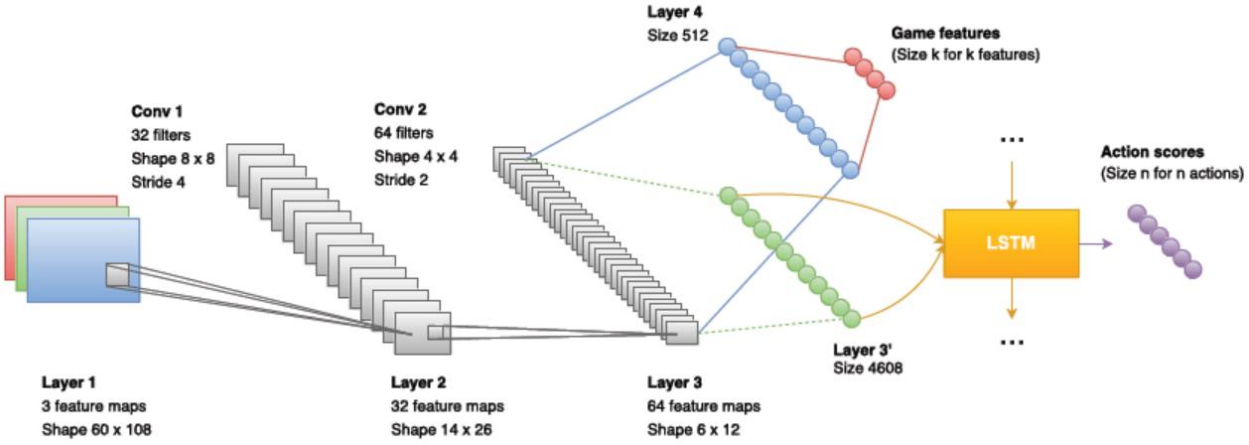
Untuk memperoleh data yang lebih bervariasi, digunakan ϵ-greedy policy pada proses training [3]. Pada fase testing, aksi yang diambil agen adalah aksi dengan nilai Q function tertinggi, sedangkan pada fase training, terdapat peluang sebesar ϵ untuk mengambil aksi random. Hal ini dilakukan untuk mengeksplorasi kemungkinan-kemungkinan yang belum pernah dicoba dan memperbaiki kemungkinan nilai Q function yang salah. Nilai ϵ akan dikurangi seiring berjalannya proses training untuk mengurangi proses eksplorasi pada akhir training.

Pada pengembangan berikutnya dilakukan berbagai modifikasi pada arsitektur dan mekanisme training yang dilakukan dalam DQN sehingga diperoleh hasil yang lebih baik. Pengembangan-pengembangan tersebut antara lain penggunaan prioritized experience replay [24], penggunaan advantage function [25], implementasi DQN secara paralel dengan menggunakan beberapa komputer [26], dan penggunaan actor-critic network untuk environment yang bersifat continuous [27].

VII. Asynchronous Advantage Actor Critic

Asynchronous Advantage Actor Critic (A3C) adalah sebuah arsitektur deep reinforcement learning baru yang dikembangkan oleh DeepMind pada tahun 2016. Kelebihan dari A3C dibandingkan DQN adalah penggunaan resource yang lebih sedikit, proses training yang lebih cepat, dapat digunakan untuk environment yang bersifat continuous, memiliki kemampuan untuk mengingat state sebelumnya lebih jauh, dan dapat mencapai skor yang lebih tinggi dibanding skor yang dicapai DQN dengan rata-rata skor mencapai 4 kali rata-rata skor DQN [28].

Perbedaan pertama antara A3C dengan DQN adalah proses training dari algoritma A3C berjalan secara asynchronous. A3C memanfaatkan kemampuan multiprocessing sebuah multicore CPU untuk menjalankan beberapa agen sekaligus secara bersamaan. Tiap agen memiliki weight network masing-masing dan berinteraksi dengan environment mereka sendiri pada saat yang bersamaan secara paralel [28]. Kelebihan penggunaan metode ini adalah proses training dapat dipercepat karena tidak diperlukan komunikasi antara beberapa komputer



Gambar. 3. Contoh Arsitektur A3C

berbeda dan dapat dilakukan update parameter dengan metode Hogwild! [29] yang lebih cepat. Selain itu penggunaan memory dapat dikurangi karena experience replay dapat digantikan dengan agen yang bekerja secara paralel.

Perbedaan lainnya adalah digunakannya advantage actor critic untuk menentukan aksi terbaik untuk diambil. Penggunaan advantage function menghasilkan AI yang baik dalam waktu yang lebih cepat dibandingkan Q function [28]. Metode estimasi advantage function yang digunakan adalah generalized advantage estimation [15] untuk menghasilkan estimasi yang dapat diatur keseimbangan antara bias dan variancenya. Selain itu, penggunaan actor-critic membuat AI dapat dimodifikasi dengan mudah untuk menghadapi environment yang bersifat continuous [30].

Perbedaan antara arsitektur DQN dengan A3C adalah pada A3C terdapat sebuah layer LSTM setelah convolutional layer dan Q network digantikan sebuah actor-critic network. LSTM membuat arsitektur dapat digunakan untuk mengatasi environment yang memiliki long term dependency, misalnya untuk menyusuri labirin [31].

VIII. ENVIRONMENT

Bagian terakhir yang diperlukan untuk pembuatan Artificial Intelligence adalah sebuah environment. Istilah environment sendiri merujuk pada program yang berinteraksi dengan AI dengan memberikan observasi atau keadaan saat itu, menerima dan memproses perintah yang diberikan lalu mengembalikan umpan balik berupa skor positif apabila aksi yang dilakukan baik dan skor negatif apabila aksi yang dilakukan buruk.

Environment yang digunakan pada percobaan ini adalah OpenAI gym [32]. Library ini dibuat dengan tujuan untuk membuat sebuah standarisasi yang dapat digunakan untuk penelitian dalam bidang reinforcement learning. Dalam OpenAI gym disediakan kumpulan environment yang dapat digunakan untuk uji coba algoritma reinforcement learning. Environment yang disediakan didesain berdasarkan sebuah interface yang sama sehingga dimungkinkan untuk membuat 1 algoritma untuk menyelesaikan semua environment. Pada penelitian ini digunakan game berbasis Atari yang disediakan oleh OpenAI Gym dan game berbasis HTML5 yang dibuat interfacenya menggunakan standar OpenAI Gym.

Game Atari yang disediakan OpenAI Gym diperoleh dari library Arcade Learning Environment (ALE). Dalam library ini terdapat lebih dari 50 ROM game Atari yang dapat digunakan sebagai environment untuk reinforcement learning. Kegunaan library ALE pertama-tama adalah menyediakan fungsi-fungsi untuk mengendalikan emulator Atari. Selain itu disediakan pula fungsi-fungsi untuk mengakses data-data yang terdapat dalam emulator Atari, seperti tampilan layar, skor, dan nyawa player.

Untuk game HTML5 yang dibuat sendiri, diperlukan bantuan 2 library, yaitu Phaser dan Pyppeteer. Phaser adalah sebuah library open source untuk pengembangan game berbasis HTML5 dan javascript. Library ini digunakan untuk pengembangan game pada platform web browser baik pada desktop maupun mobile dengan WebGL dan canvas rendering. Pyppeteer merupakan implementasi dari library Puppeteer dengan menggunakan Python. Puppeteer sendiri adalah sebuah library untuk mengontrol web browser Chrome atau Chromium. Dengan Pyppeteer dapat diambil screenshot dari sebuah web page, crawling sebuah website, pengiriman form secara otomatis, otomasi testing webapp, dan sebagainya.

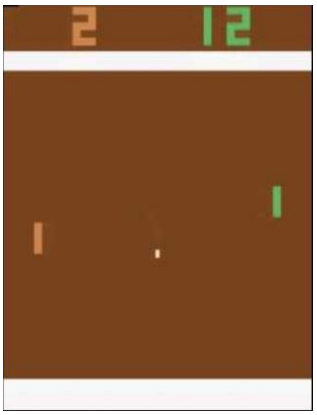
Pada environment dengan standar OpenAI Gym harus terdapat 4 komponen penting, yaitu kemampuan untuk memberikan observation atau keadaan game saat itu, menerima dan memproses perintah yang diberikan, serta memberikan reward untuk tiap aksi yang diberikan. Hasil observasi game dapat diperoleh dari screenshot game pada web browser yang dilakukan dengan bantuan library Pyppeteer. Pengiriman perintah dari AI ke game HTML5 dilakukan dengan bantuan Pyppeteer. Library Pyppeteer dapat mengakses dan mengubah isi variabel pada game sehingga perintah dapat dijalankan. Setelah perintah diberikan, perintah akan dijalankan sesuai logika game yang sudah ditetapkan menggunakan bantuan library Phaser. Reward diperoleh dengan cara mengakses isi variabel skor pada game dengan menggunakan library Pyppeteer.

IX. UJI COBA

Terdapat 2 jenis uji coba yang dilakukan, yaitu uji coba hyperparameter dan uji coba perbandingan skor. Pada uji coba pengubahan hyperparameter, terdapat 47 skenario uji coba yang akan dilakukan. Masing-masing uji coba dilakukan menggunakan game Pong berbasis konsol Atari.

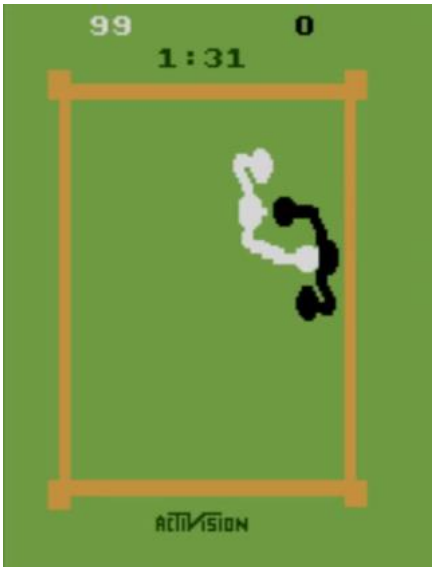
Pada uji coba perbandingan skor, terdapat 5 skenario uji coba, dimana masing-masing skenario digunakan untuk menguji tiap game yang ada. Pada tiap skenario akan diambil nilai yang dapat dicapai AI dan nilai yang dapat dicapai oleh pemain pemula.

Pada uji coba pengubahan hyperparameter, terdapat 47 skenario uji coba yang akan dilakukan. Akan dilakukan pengubahan hyperparameter yang dapat mempengaruhi kecepatan training dan score yang dapat diperoleh AI. Terdapat 9 hyperparameter yang nilainya hendak diubah, yaitu jumlah worker, jumlah frameskip, ukuran input, update step, gamma, learning rate, jumlah hidden unit LSTM, tau, dan GPU support. Pada uji coba hyperparameter akan digunakan 2 game berbeda, yaitu game Atari Pong dan game Boxing.



Gambar. 4. Game Pong

Tujuan dari game Pong adalah untuk mencegah bola masuk ke gawang sendiri dan berusaha memantulkan bola agar masuk ke gawang lawan di sebelah kiri. Pada uji coba, AI akan diberi skor +1 apabila berhasil memasukkan bola ke gawang lawan, sedangkan jika lawan berhasil memasukkan bola ke gawang AI maka AI akan diberi skor -1. Permainan akan berakhir ketika salah satu pihak mendapat skor 21. Nilai maksimal dari permainan dengan sistem penilaian ini adalah 21 (jika AI menang 21 - 0) dan nilai minimum dari sistem penilaian ini adalah -21 (jika AI kalah 0 - 21).



Gambar. 5. Game Boxing

Tujuan dari game Boxing adalah dapat mengalahkan petinju lawan dengan memukul kepala lawan dan mencegah lawan memukul kepala petinju yang dimainkan. Pukulan yang mengenai bagian tubuh lain tidak dihitung. Sistem pemberian skor pada game ini adalah, AI akan diberi skor +1 apabila berhasil meninju kepala lawan, sedangkan jika lawan berhasil meninju kepala pemain maka AI akan diberi skor -1. Permainan akan berakhir ketika salah satu pihak mendapat skor 100 atau ketika waktu permainan habis. Permainan diberi waktu 2 menit. Nilai maksimal dari permainan dengan sistem penilaian ini adalah 100 (jika AI menang 100 - 0) dan nilai minimum dari sistem penilaian ini adalah -100 (jika AI kalah 0 - 100).

Pada Tabel I ditunjukkan nilai hyperparameter yang dijadikan standar pembanding setiap percobaan pengubahan nilai hyperparameter. Nilai hyperparameter yang tidak diganti akan mengikuti nilai pada Tabel I.

TABEL I
NILAI DEFAULT HYPERPARAMETER

No	Hyperparameter	Nilai
1	Worker	14
2	Frameskip	4
3	Input size	80 × 80 pixel
4	Update step	20
5	Gamma	0.99
6	Learning rate	0.0001
7	LSTM hidden unit	512
8	Tau	1
9	GPU	No

Parameter worker adalah parameter yang menentukan jumlah worker thread yang digunakan oleh AI. Parameter kontrol menggunakan 14 worker thread yang berjalan bersama pada saat training dijalankan.

TABEL II
PENGARUH PARAMETER JUMLAH WORKER PADA GAME PONG

No	Worker	Max Rolling Average	Waktu
1	6	20.3	7:09:40
2	8	20.25	6:47:37
3	10	20.25	7:29:52
4	12	19.95	5:39:34
5	14	20	5:31:58
6	16	19.1	7:33:31

TABEL III
PENGARUH PARAMETER JUMLAH WORKER PADA GAME BOXING

No	Worker	Max Rolling Average	Waktu
1	6	71.2	11:57:06
2	8	77.45	11:47:57
3	10	32	11:46:27
4	12	48.6	11:58:51
5	14	71.4	12:00:59
6	16	6.65	9:38:05

Parameter frameskip adalah parameter yang menentukan jumlah frame yang dilompati oleh AI setiap langkah. Parameter kontrol menggunakan 4 frameskip dengan mengambil max value terhadap 2 frame terakhir.

TABEL IV
PENGARUH PARAMETER FRAMESKIP PADA GAME PONG

No	Frameskip	Max Rolling Average	Waktu
1	3	17.4	1
2	4	20	2
3	5	4.75	3

TABEL V
PENGARUH PARAMETER FRAMESKIP PADA GAME BOXING

No	Frameskip	Max Rolling Average	Waktu
1	3	1.45	10:59:25
2	4	71.4	12:00:59
3	5	87.8	11:49:13

Pada uji coba berikutnya dilakukan pengubahan ukuran input. Parameter input size adalah parameter yang menentukan ukuran input yang akan diproses network. Parameter kontrol menggunakan citra berukuran 80 x 80 pixel.

TABEL VI
PENGARUH UKURAN INPUT PADA GAME PONG

No	Input Size	Max Rolling Average	Waktu
1	10x10	15.05	7:30:32
2	20x20	20.15	4:59:13
3	40x40	21	5:46:03
4	60x60	20.05	6:08:27
5	80x80	20	5:31:58

TABEL VII
PENGARUH UKURAN INPUT PADA GAME BOXING

No	Input Size	Max Rolling Average	Waktu
1	10x10	36.1	11:53:20
2	20x20	90.65	10:22:15
3	40x40	99.85	11:20:57
4	60x60	96.7	11:51:41
5	80x80	71.4	12:00:59

Parameter update step adalah parameter yang menentukan jumlah frame dan reward yang disimpan oleh worker sebelum melakukan perhitungan gradien dan melakukan update bobot model. Parameter kontrol menggunakan 20 step sebelum update.

TABEL VIII
PENGARUH PARAMETER UPDATE STEP PADA GAME PONG

No	Update Step	Max Rolling Average	Waktu
1	5	7.85	4:34:40
2	10	19.25	5:09:32
3	15	19.8	5:41:31
4	20	20	5:31:58

TABEL IX
PENGARUH PARAMETER UPDATE STEP PADA GAME BOXING

No	Update Step	Max Rolling Average	Waktu
1	5	95.7	11:32:18
2	10	100	11:24:04
3	15	57.55	11:46:37
4	20	71.4	12:00:59

Parameter gamma adalah parameter yang menentukan jumlah discount factor yang digunakan oleh AI. Parameter kontrol menggunakan nilai gamma sebesar 0.99.

TABEL X
PENGARUH PARAMETER GAMMA PADA GAME PONG

No	Gamma	Max Rolling Average	Waktu
1	0.93	18.8	7:47:09
2	0.95	19.65	7:13:55
3	0.98	20.25	7:04:39
4	0.99	20	5:31:58
5	0.991	20.3	7:00:55
6	0.995	19.8	7:18:25

TABEL XI
PENGARUH PARAMETER GAMMA PADA GAME BOXING

No	Gamma	Max Rolling Average	Waktu
1	0.93	39	11:13:43
2	0.95	49.5	11:51:31
3	0.98	32.75	11:15:42
4	0.99	71.4	12:00:59
5	0.991	75.85	12:00:11
6	0.995	18.85	10:48:38

Parameter learning rate adalah parameter yang menentukan learning rate yang digunakan untuk update bobot model. Parameter kontrol menggunakan learning rate sebesar 0.0001.

TABEL XII
PENGARUH PARAMETER LEARNING RATE PADA GAME PONG

No	Learning Rate	Max Rolling Average	Waktu
1	0.00003	-0.85	7:25:57
2	0.0001	20	5:31:58
3	0.0003	20.03	7:51:39
4	0.001	-1.25	5:44:29

TABEL XIII
PENGARUH PARAMETER LEARNING RATE PADA GAME BOXING

No	Learning Rate	Max Rolling Average	Waktu
1	0.00003	-3.75	9:25:45
2	0.0001	71.4	12:00:59
3	0.0003	-16.25	7:25:19
4	0.001	-25.25	3:28:53

Parameter LSTM hidden unit menyatakan banyaknya hidden unit yang digunakan oleh model. Parameter kontrol menggunakan 512 hidden unit untuk LSTM.

TABEL XIV
PENGARUH UKURAN LSTM PADA GAME PONG

No	LSTM Hidden Unit	Max Rolling Average	Waktu
1	32	19.9	6:14:07
2	64	20	5:05:18
3	128	20.15	7:07:25
4	256	17.8	7:13:15
5	384	19.1	4:39:10
6	512	20	5:31:58

Parameter Tau pada program ini adalah parameter lambda yang digunakan dalam perhitungan Generalized Advantage Estimation. Parameter ini tidak dinamai lambda pada program karena kata "lambda" merupakan sebuah keyword pada bahasa pemrograman Python. Parameter kontrol menggunakan nilai Tau 1.

TABEL XV
PENGARUH UKURAN LSTM PADA GAME BOXING

No	LSTM Hidden Unit	Max Rolling Average	Waktu
1	32	-6.1	10:34:35
2	64	17.6	9:43:10
3	128	6.25	8:56:17
4	256	78.9	11:10:12
5	384	79.35	12:00:40
6	512	71.4	12:00:59

TABEL XVI
PENGARUH PARAMETER TAU PADA GAME PONG

No	Tau	Max Rolling Average	Waktu
1	0.3	18.9	7:58:59
2	0.5	17.7	6:48:25
3	0.7	20.55	7:57:33
4	0.9	19.9	7:11:39
5	1	20	5:31:58
6	1.1	20.4	5:41:19

TABEL XVII
PENGARUH PARAMETER TAU PADA GAME BOXING

No	Tau	Max Rolling Average	Waktu
1	0.3	21.25	11:36:44
2	0.5	38.1	12:00:29
3	0.7	80.2	11:51:47
4	0.9	84.05	11:29:29
5	1	71.4	12:00:59
6	1.1	-19.65	9:38:48

Parameter gpu_id mengatur GPU mana saja yang dapat digunakan oleh program untuk melakukan training. Untuk uji coba ini, akan digunakan sebuah GPU dan akan dilakukan pengubahan jumlah worker. Parameter kontrol tidak menggunakan GPU dan menggunakan 14 worker. Khusus pada uji coba ini, proses training dilakukan hanya selama 1 jam untuk masing-masing skenario uji coba.

TABEL XVIII
PENGARUH PENGGUNAAN GPU PADA GAME PONG

No	Worker	Max Rolling Average	Waktu
1	4	10.7	0:56:54
2	6	20.25	0:59:05
3	8	20.45	0:56:08
4	10	20	0:58:06
5	12	20	0:59:05

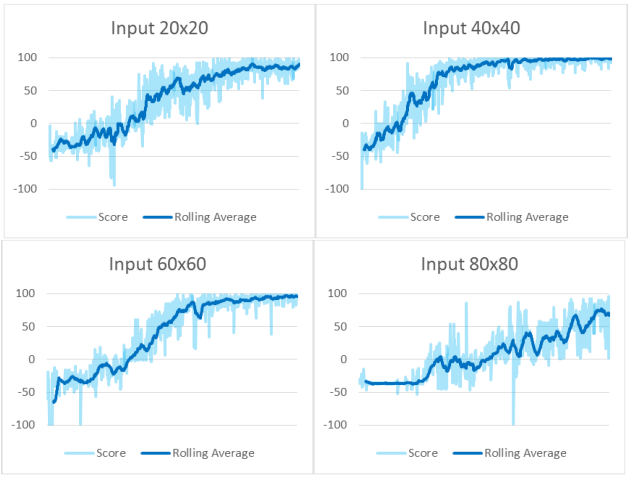
TABEL XIX
PENGARUH PENGGUNAAN GPU PADA GAME BOXING

No	Worker	Max Rolling Average	Waktu
1	4	-20.25	0:59:30
2	6	19	0:59:24
3	8	14.3	1:00:01
4	10	19.65	1:00:05
5	12	47.4	0:59:10

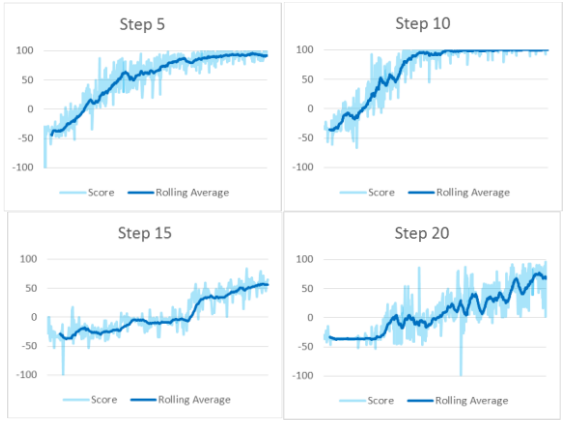
Pada uji coba skor akan dilakukan pembandingan antara skor yang berhasil diperoleh oleh AI dengan skor yang diperoleh pemain pemula. Terdapat 5 game berbeda yang akan digunakan dalam uji coba ini, 3 diantaranya adalah game Atari sedangkan 2 lainnya adalah game berbasis HTML5 yang merupakan game buatan sendiri yang telah diintegrasikan dengan library OpenAI gym. Tiga game Atari tersebut antara lain: Pong, Boxing, dan Space Invaders. Sedangkan untuk game HTML5 adalah game balapan sederhana dan game Pinball. Skor yang diperoleh pemain pemula akan digunakan sebagai pembading untuk skor yang diperoleh AI. Akan ditunjukkan perbandingan skor rata-rata, skor tertinggi, dan skor terendah dari pemain pemula beserta AI.

Tujuan dari game Space Invaders adalah untuk menghalau agar alien tidak sampai ke bumi. Sistem pemberian skor pada game ini adalah, alien pada baris 1 dari bawah memberi skor +5 jika ditembak, baris 2 +10, baris 3 +15, dan seterusnya. Kapal induk alien memberi skor +200. Jika player tertembak maka pemain akan kehilangan 1 nyawa (total nyawa pemain 3). Permainan akan berakhir ketika alien berhasil mencapai bumi atau pemain kehabisan nyawa.

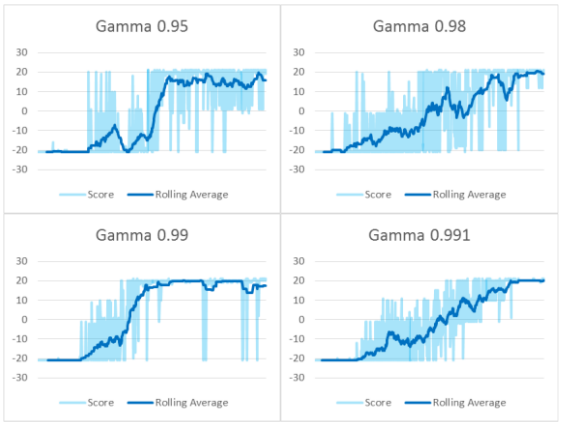
Gambar. 6. Grafik Skor dan Rolling Average dari Pengubahan Jumlah Worker pada Game Boxing



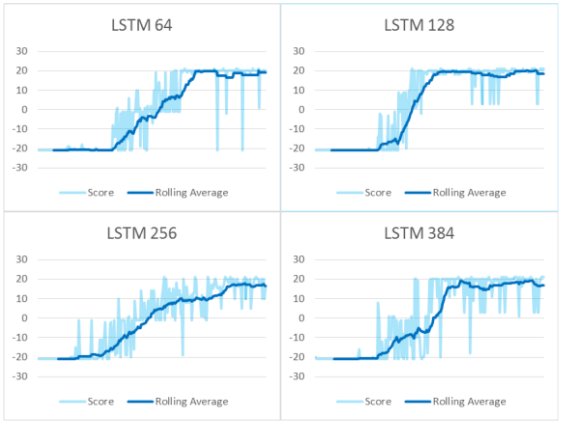
Gambar. 7. Grafik Skor dan Rolling Average dari Pengubahan Ukuran Input pada Game Boxing



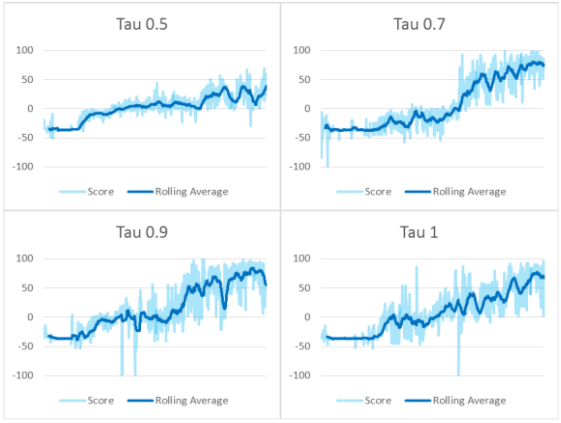
Gambar. 8. Grafik Skor dan Rolling Average dari Pengubahan Jumlah Step pada Game Boxing



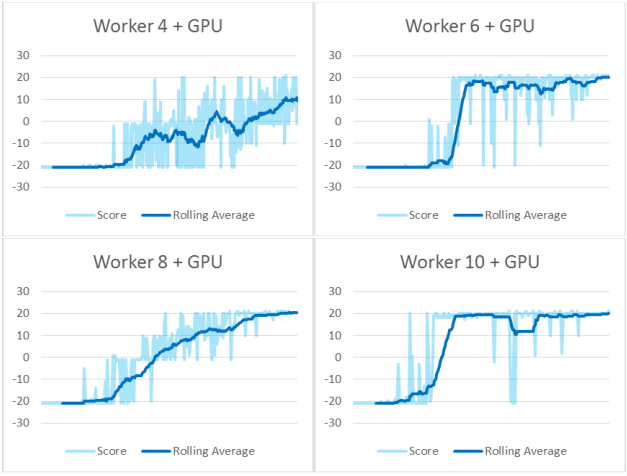
Gambar. 9. Grafik Skor dan Rolling Average dari Pengubahan Parameter Gamma pada Game Pong



Gambar. 10. Grafik Skor dan Rolling Average dari Pengubahan Ukuran LSTM pada Game Pong



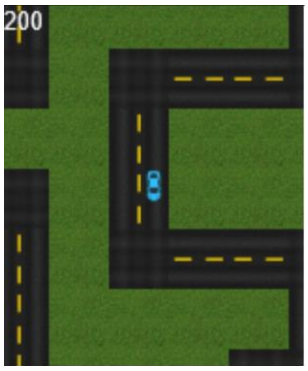
Gambar. 11. Grafik Skor dan Rolling Average dari Pengubahan Parameter Tau pada Game Boxing



Gambar. 12. Grafik Skor dan Rolling Average dari Penggunaan GPU pada Game Pong

Gambar. 13. Game Space Invaders

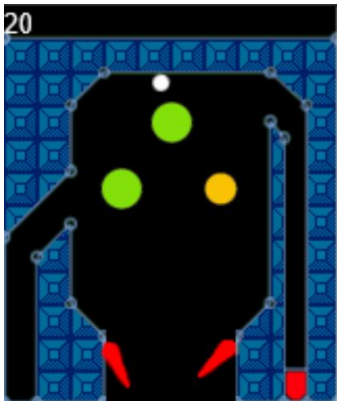
Tujuan dari game Racing adalah mengendarai mobil secepat mungkin untuk mencapai garis finish. Sistem penilaian game ini adalah, AI akan diberi skor apabila berhasil mencapai petak jalan berikutnya. Skor yang diperoleh adalah +20 pada awalnya, namun akan dikurangi terus menerus setiap waktu sehingga untuk mendapatkan skor penuh mobil harus berjalan dengan cepat. Jika berhasil mencapai garis finish, AI akan diberi skor bonus berupa 50% dari skor sebelum mencapai finish.



Gambar. 14. Game Racing

Tujuan dari permainan ini adalah mencegah bola jatuh ke lubang sambil berusaha mendapatkan skor dengan cara mengarahkan agar bola menumbuk bantalan di tengah. Pada game ini, AI akan diberi skor +5 apabila bola berhasil mengenai salah satu dari 3 bantalan di tengah area

permainan. AI akan kalah apabila bola jatuh ke dalam lubang dan tidak dapat dipukul kembali ke atas oleh pemukul.



Gambar.15. Game Pinball

.

TABEL XX
PERBANDINGAN SKOR PEMAIN PEMULA DENGAN AI PADA GAME ATARI

	Pong		Boxing		Space Invaders	
	Human	**AI**	**Human**	**AI**	**Human**	**AI**
Average	-10.1	20.6	7.7	98.7	972.5	2896.5
Max	13	21	20	100	2150	3225
Min	-19	20	3	96	575	2660

TABEL XXI
PERBANDINGAN SKOR PEMAIN PEMULA DENGAN AI PADA GAME HTML5

	Racing		Pinball	
	Human	**AI**	**Human**	**AI**
Average	555	966.5	95	124.5
Max	683	1041	220	350
Min	415	840	30	15

Hasil dari seluruh uji coba AI telah didokumentasikan dalam bentuk video yang dapat dilihat pada website Youtube dengan link https://tinyurl.com/y8swcxl6.

X. KESIMPULAN

Berdasarkan hasil tersebut dapat disimpulkan beberapa poin yang akan disebutkan dibawah ini.

1. Reinforcement Learning adalah salah satu jenis machine learning yang dapat digunakan untuk membuat sebuah Artificial Intelligence dengan trial and error, tanpa diperlukan handcrafted feature.
2. Algoritma deep learning dapat digabungkan dengan algoritma reinforcement learning untuk mengatasi masalah yang lebih kompleks dan bervariasi, seperti memainkan berbagai macam jenis game berbeda hanya dari mempelajari citra output dari layar.
3. Asynchronous Advantage Actor-Critic (A3C) adalah algoritma deep reinforcement learning yang dapat belajar memainkan beberapa jenis game yang berbeda dengan 1 arsitektur. A3C terdiri dari 3 komponen utama, yaitu Convolutional Neural Network (CNN), Long Short-Term Memory Network (LSTM), dan Actor-Critic Network.
4. Pada algoritma A3C, Convolutional Neural Network bertugas merangkum citra input dari layar konsol game dengan cara mengekstraksi fitur penting yang diperoleh dari citra input. Hasil dari Convolutional Neural Network ini menjadi representasi state saat ini dan akan diolah lebih lanjut.
5. Dalam algoritma A3C, Long Short-Term Memory Network berfungsi sebagai pengolah data state yang bersifat sequential untuk merangkum perubahan yang terjadi pada environment. Data representasi state dari Convolutional Neural Network diolah menjadi representasi state dan perubahannya oleh Long Short-Term Memory Network.
6. Actor-Critic Network dalam A3C berfungsi menentukan tindakan terbaik untuk dilakukan pada situasi yang diberikan.
7. Pada paper acuan, algoritma A3C diimplementasikan menggunakan CPU sedangkan pada program acuan dilakukan modifikasi kepada program agar dapat memanfaatkan GPU. Hasilnya adalah proses training dapat dilakukan dengan lebih cepat.
8. Dari hasil percobaan, AI yang dibuat menggunakan A3C memperoleh nilai yang lebih tinggi dibandingkan nilai pemain pemula dalam 5 jenis game yang berbeda.

DAFTAR PUSTAKA

[1] G. Tesauro, “Temporal difference learning and TD-Gammon,” *Commun. ACM*, 1995, doi: 10.1145/203330.203343.

[2] M. Campbell, a. J. Hoane Jr., and F. Hsu, “Deep Blue,” *Artif. Intell.*, vol. 134, no. 1–2, pp. 57–83, 2002, doi: 10.1016/S0004-3702(01)00129-1.

[3] V. Mnih *et al.*, “Playing atari with deep reinforcement learning,” *arXiv Prepr. arXiv1312.5602*, 2013.

[4] D. Silver *et al.*, “Mastering the game of Go without human knowledge,” *Nature*, 2017, doi: 10.1038/nature24270.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *Adv. Neural Inf. Process. Syst.*, 2012, doi: http://dx.doi.org/10.1016/j.protcy.2014.09.007.

[6] R. S. Sutton and A. G. Barto, *Introduction to reinforcement learning*, vol. 135. MIT press Cambridge, 1998.

[7] V. R. Konda and J. N. Tsitsiklis, “Actor-critic algorithms,” in *Advances in neural information processing systems*, 2000, pp. 1008–1014.

[8] R. Bellman, “A Markovian decision process,” *J. Math. Mech.*, pp. 679–684, 1957.

[9] G. Brockman *et al.*, “Evolutionary algorithms for reinforcement learning,” *J. Artif. Intell. Res.*, vol. 47, pp. 253–279, Jun. 2013.

[10] R. S. Sutton, D. Mcallester, S. Singh, and Y. Mansour, “Policy Gradient Methods for Reinforcement Learning with Function Approximation,” *Adv. Neural Inf. Process. Syst. 12*, 1999, doi: 10.1.1.37.9714.

[11] R. J. Willia, “Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning,” *Mach. Learn.*, 1992, doi: 10.1023/A:1022672621406.

[12] N. Kohl and P. Stone, “Policy gradient reinforcement learning for fast quadrupedal locomotion,” in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA ’04. 2004*, 2004, doi: 10.1109/ROBOT.2004.1307456.

[13] R. S. Sutton, “Learning to Predict by the Methods of Temporal Differences,” *Mach. Learn.*, 1988, doi: 10.1023/A:1022633531479.

[14] C. J. C. H. Watkins and P. Dayan, “Q-learning,” *Mach. Learn.*, 1992, doi: 10.1007/BF00992698.

[15] G. Brockman *et al.*, “High-dimensional continuous control using generalized advantage estimation,” *Adv. Neural Inf. Process. Syst.*, vol. 47, no. 7540, pp. 693–701, Jun. 2015.

[16] V. Sharma, S. Rai, and A. Dev, “A Comprehensive Study of Artificial Neural Networks,” *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, 2012.

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, 1998, doi: 10.1109/5.726791.

[18] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," *CoRR*, vol. abs/1511.0, 2015.
[19] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur, "Recurrent Neural Network based Language Model," *Interspeech*, 2010.
[20] S. Hochreiter, "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions," *Int. J. Uncertainty, Fuzziness Knowledge-Based Syst.*, 1998, doi: 10.1142/S0218488598000094.
[21] S. Hochreiter and J. Urgen Schmidhuber, "LONG SHORT-TERM MEMORY," *Neural Comput.*, 1997, doi: 10.1162/neco.1997.9.8.1735.
[22] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.
[23] L. Lin, "Reinforcement Learning for Robots Using Neural Networks," *Report, C.*, 1993.
[24] V. Mnih *et al.*, "Prioritized Experience Replay," *Int. Conf. Mach. Learn.*, 2015, doi: 10.1038/nature14236.
[25] Z. Wang, N. de Freitas, and M. Lanctot, "Dueling Network Architectures for Deep Reinforcement Learning," *arXiv Prepr. arXiv1511.06581*, 2015.
[26] A. Nair *et al.*, "Massively Parallel Methods for Deep Reinforcement Learning," *arXiv:1507.04296*, 2015, doi: 10.1109/IJCNN.2010.5596468.
[27] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," *arXiv Prepr. arXiv1509.02971*, 2015.
[28] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.
[29] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," in *Advances in neural information processing systems*, 2011, pp. 693–701.
[30] DeepMind, "Asynchronous Methods for Deep Reinforcement Learning: MuJoCo - YouTube." [Online]. Available: https://www.youtube.com/watch?v=Ajjc08-iPx8&feature=youtu.be. [Accessed: 01-May-2018].
[31] DeepMind, "Asynchronous Methods for Deep Reinforcement Learning: Labyrinth - YouTube." [Online]. Available: https://www.youtube.com/watch?v=nMR5mjCFZCw&feature=youtu.be. [Accessed: 01-May-2018].
[32] G. Brockman *et al.*, "OpenAI Gym." 2016.

Evan Kusuma Susanto Lahir di Kota Surabaya, Jawa Timur, Indonesia pada tahun 1996. Saat ini sebagai mahasiswa tingkat akhir dengan jurusan S1 Teknik Informatika di STTS Surabaya. Risetnya berfokus pada machine learning dan deep reinforcement learning.

Yosi Kristian lahir di Tuban, Jawa Timur, Indonesia pada tahun 1981. Beliau menyelesaikan studi S1 di program studi Informatika STTS pada tahun 2004 dan menyelesaikan S2 di program studi Informatika STTS pada tahun 2008. Beliau telah menyelesaikan studi masternya di jurusan Informatika STTS. Saat ini beliau sedang melanjutkan Ph.D di Institut Teknologi Sepuluh November (ITS), Surabaya, Indonesia. Penelitiannya terdiri dari Machine Learning, Intellegence System, dan Computer Vision. Beliau merupakan member dari IAENG.