TRADING CRYPTOCURRENCY AGENT MENGGUNAKAN DEEP REINFORCEMENT LEARNING DENGAN MODEL YANG BERBASIS Q-LEARNING DENGAN MENGGUNAKAN DQN, DOUBLE DQN, DAN DUELING DQN PADA STUDI KASUS BITCOIN

Ida Bagus Dwi Putra Purnawa[1)] **Eddy Muntina Dharma**[2)] **Ida Bagus Kresna Sudiatmika**[3)]
Program Studi Teknik Informatika[1) 2) 3)]
STMIK Primakara, Denpasar Selatan, Bali[1) 2) 3)]
putrapurnawa@gmail.com[(1)] aguseddy@gmail.com[2)] kresna@primakara.ac.id[3)]

ABSTRACT

This study aims to test the Q-Learning-based Deep Reinforcement Learning algorithm models such as DQN, Double DQN, and Dueling DQN in their implementation as trading agents on trading bots. Case studies taken on a, Bitcoin. In testing this dataset was taken from one of the exchange platforms, to be precise from Gemini Exchange, a cryptocurrency trading platform. The data is used to simulate a trading environment to train trading agents. The test method is carried out by measuring the average reward (profit/loss) and the percentage increase and decrease in the average reward value and compare the architecture used in this study with the VGG16 architecture.

Keywords*: Trading Cryptocurrency, DQN, Double DQN, Dueling DQN, Deep Reinforcement Learning.*

ABSTRAK

Penelitian ini bertujuan untuk menguji model algoritma Deep Reinforcement Learning yang berbasis Q-Learning seperti DQN, Double DQN, dan Dueling DQN dalam pengimplementasiannya sebagai trading agent pada trading bot. Studi kasus yang diambil pada platform trading cryptocurrency yaitu Bitcoin. Pada pengujian ini dataset diambil dari salah satu platform exchange, tepat nya dari Gemini Exchange, salah satu platform trading cryptocurrency. Data digunakan untuk mensimulasikan environment trading untuk melatih trading agent. Metode pengujian dilakukan dengan mengukur nilai rata-rata reward (profit/loss) dan persentase kenaikan dan penurunan dari nilai rata-rata reward dan juga membandingkan arsitektur yang digunakan dalam penelitian ini dengan arsitektur VGG16.

Kata Kunci : Trading Cryptocurrency, DQN, Double DQN, Dueling DQN, Deep Reinforcement Learning.

PENDAHULUAN

Cryptocurrency atau biasa disebut juga dengan aset kripto merupakan investasi yang menawarkan keuntungan cukup tinggi sehingga menarik masyarakat luas untuk investasi pada *cryptocurrency*, khususnya pada generasi milenial di negara Indonesia (Widya, 2020)[1]. Popularitas *cryptocurrency* dibuktikan dengan harga *Bitcoin* yang naik sebesar 450% sepanjang tahun 2020, *Ethereum* naik lebih dari 1000% dalam satu tahun terakhir, dan *Yearn Finance (YFI)* yang mencapai rekor naik sampai Rp 650 juta dalam waktu 3 bulan. Ditambah berita *public figure* yang terkenal seperti *Elon Musk*, pemilik *Tesla*, yang memiliki aset kripto di *Doge* sehingga valuasi *cryptocurrency* ini melambung tinggi jauh dibandingkan kenaikan saham pada umumnya (Widya, 2020)[1].

Investasi yang banyak dilakukan pada *cryptocurrency* dalam jangka pendek adalah *trading*. *Trading* merupakan suatu konsep dari ekonomi dasar yang di dalamnya terdapat aktivitas jual beli produk barang atau jasa[2]. Dalam konsep finansial, kegiatan *trading* ini lebih mengacu pada aktivitas jual beli sekuritas seperti saham. Selain itu, *trading* juga kerap kali dilakukan di pasar berjangka panjang dan juga pasar valuta asing atau yang saat ini sering

disebut dengan sebagai *forex (foreign exchange)* dan orang yang melakukan *trading* disebut sebagai *trader*. Sejak kehadiran *Bitcoin* beberapa tahun lalu, *trading bitcoin* menjadi salah satu opsi *trading* yang baru, yang menyebabkan tren *trading* pada pada *cryptocurrency* naik daun dan kian digemari.

Banyak orang yang memiliki kesibukan pada hal lain di dunia nyata yang menyebabkan mereka tidak bisa fokus melakukan *trading* atau tidak mempunyai waktu dalam menganalisa tren sehingga mereka menggunakan strategi *trading* yang kurang tepat, maka dari itu penggunaan *trading bot* merupakan solusi dalam mengatasi hal tersebut (Nugroho, 2021)[3]. *Trading bot* merupakan sebuah program yang dapat mengeksekusi dan pemosisian *trading online* dengan cepat, jadi *trading bot* berfungsi mengotomatisasikan *trading* dan memungkinkan untuk mengatur strategi *trading* berdasarkan parameter yang diinginkan. Banyak pihak yang ada sudah menyediakan layanan *trading bot* untuk para *trader*, seperti *Royal Q Trading*, *Cryptohopper*, *Tokocrypto* dan lain-lainnya. Selain menggunakan layanan tersebut, opsi lainnya dengan cara membuat program *trading bot* sendiri, untuk membuat *trading bot* sendiri pun yang pastinya membutuhkan pengetahuan dibidangnya. Pada penelitian ini penulis lebih berfokus pada pembuatan *trading agent-nya*, yang dimana berfokus pada model yang dibuat sehingga nantinya model tersebut dapat digunakan ketika membuat atau membangun *trading bot*. Selain membuat *trading bot*, salah satu metode yang bisa digunakan untuk membantu melakukan trading yaitu dengan metode *forecasting*. Metode ini biasanya dilakukan dengan cara memprediksi harga stock maupun tren untuk bisa membantu para *trader* dalam melakukan *trading*. *Forcecasting* dapat dilakukan dengan menerapkan algoritma seperti *Autoregressive (AR)*, *XGBoost*, *LSTM* maupun *GRU*. *Forecasting* biasanya dilakukan untuk memprediksi *stock price* ataupun *stock market*, yang dimana biasanya hasil prediksinya selalu tidak tepat, hal itu dikarenakan *share price* sebagian besar bergantung pada opini *trader* terhadap masa depan perusahaan dan tidak hanya pada harga sebelumnya, jadi tidak masuk akal untuk memprediksi masa depan *stock price* menggunakan harga sebelumnya (Ivan, 2020)[4]

Menggunakan *forecasting* hanya dapat memberikan informasi tren yang terjadi sehingga *trader* perlu melakukan analisa terhadap infomasi tren tesebut untuk diaplikasikan pada strategi *trading*, jadi menggunakan *forecasting* tidak melakukan otomatisasi pada aktivitas *trading*. *Reinforcement Learning (RL)* merupakan salah satu bagian dari *machine learning* yang dimana algoritma *RL* belajar untuk memaksimalkan *reward* pada sebuah *environment*, dan sangat berguna ketika permasalahan melibatkan pengambilan keputusan atau mengambil tindakan[11]. Dengan menggunakan *Reinforcement Learning*, yaitu dengan melatih *agent* tidak hanya untuk menganalisa tren tetapi juga untuk melakukan otomatisasi pada aktivitas *trading*. Salah satu model *RL* yang umum digunakan yaitu *Q-Learning*. *Q-Learning* merupakan metode tertentu untuk belajar mengoptimalkan nilai tindakan *(action)*. Ide utama dari *Q-Learning* adalah algoritma memprediksi nilai dari pasangan *state-action*, dan setelah itu membandingkan prediksi tersebut untuk diamati nilai *reward-nya* dan kemudian memperbaharui parameter pada algoritma, yang membuat prediksi selanjutnya menjadi lebih baik. Untuk mendapatkan nilai dari *state-action*, *Q-Learning* menggunakan *Q-Table* yang sudah dibuat untuk menentukan nilai dari *state-action* atau yang biasa di sebut *Q-Value*, yang dimana membuat baik dari *action space* dan *state space-nya* bersifat diskrit. Menggunakan *Q-Table* membuat *Q-Learning* memiliki keterbatasan, maka dari itu *Q-Table* bisa diganti dengan menggunakan *neural network* yang dimana tidak perlu menentukan nilai *Q-Value* dari pasangan *state-action* melainkan menggunakan *state* sebagai *input* dari *neural network* hingga menghasilkan pasangan nilai *Q-Value* dan *action*. *Q-Learning* dengan menggunakan *neural network* disebut dengan *Deep Q-Learning* atau *Deep Q-Network (DQN)*. *Q-learning* merupakan model yang paling sederhana untuk diaplikasikan tetapi *Q-learning* memiliki banyak limitasi dalam pengimplementasiannya pada dunia nyata. Maka dari itu penggunaan *neural*

network pada *Q-learning* dapat mengatasi limitasi tersebut, sehingga membuat *DQN* menjadi salah satu pemilihan model yang baik. Selain *DQN* terdapat beberapa model yang berbasis *Q-Learning* seperti *Double DQN*, dan *Dueling DQN*. Model tersebut merupakan variasi model dari *DQN*, yang dimana *Double DQN* menggunakan dua *neural network* model yang identik sama halnya seperti *DQN*. Satu model belajar ketika saat *experience replay*, sama seperti yang dilakukan *DQN*, dan yang lainnya merupakan *copy* dari episode terakhir dari model pertama. Pada *Double DQN*, *Q-value* dikomputasi pada model kedua. Sedangkan *Dueling DQN* memisahkan *network* menjadi dua *streams*, satu untuk mengestimasi *state-value* dan yang lain untuk mengestimasi *state-dependent action advantages*. Setelah dua *streams* tersebut, modul terakhir dari *network* adalah menggabungkan *output* dari *state-value* dan *advantage*. Pada penelitian ini penulis menguji model yang berbasis *Q-Learning* seperti *DQN*, *Double DQN*, dan *Dueling DQN* dalam pengimplementasiannya sebagai *trading agent*, dengan kata lain untuk menguji model yang mana yang memiliki performa yang lebih baik jika di implementasikan pada *environment trading cryptocurrency*. Model tersebut dipilih untuk diuji karena memiliki arsitektur yang hampir sama sehingga penentuan parameter modelnya tidak berbeda jauh. Selain hal itu *DQN* mempunyai prinsip yang simple jadi lebih mudah dipelajari dari model yang lain. Pada penelitian ini menggunakan arsitektur *neural network* yang dibuat secara *custom* dan juga akan dibandingkan dengan arsitektur *VGG16* dari segi performa. *Bitcoin* dipilih sebagai studi kasus karena selain memiliki *volume trading* yang tinggi, dan juga salah satu mata uang *crypto* yang populer selain pesaingnya yaitu *Ethereum*. Pada penelitian ini *output-nya* berupa model yang sudah dilatih dan diuji, serta hasil dari pengujian model.

TINJAUAN PUSTAKA

Cryptocurrency

Menurut *Trautman* (2014), *cryptoccurency* merupakan bagian dari uang digital yang mungkin memiliki institusi terpusat atau berbasis pada jaringan terdesentralisasi[6].

Trading

Menurut *Adam* (2021), *trading* merupakan suatu konsep dari ekonomi dasar yang di dalamnya terdapat aktivitas jual beli produk barang atau jasa. Dalam konsep finansial, kegiatan trading ini lebih mengacu pada aktivitas jual beli sekuritas seperti saham. Selain itu, trading juga kerap kali dilakukan di pasar berjangka panjang dan juga pasar valuta asing atau yang saat ini sering disebut dengan sebagai *forex (foreign exchange)*[10]..

Reinforcement Learning

Reinforcement Learning (RL) merupakan salah satu bagian dari *machine learning* yang dimana algoritma *RL* belajar untuk memaksimalkan *reward* pada sebuah *environment*, dan sangat berguna ketika permasalahan melibatkan pengambilan keputusan atau mengambil tindakan[11]..

Q-Learning

Q-learning merupakan pengembangan dari *temporal-difference (TD)* yang juga disebut dengan istilah *off-policy TD control*, dan cara memperbarui *value function (V)* dari *Q-learning* berdasarkan nilai *action-value function* yang paling maksimum di-*state* selanjutnya, di mana α merupakan learning rate $(0 < \alpha \leq 1)$ yang nantinya nilai tersebut akan selalu diperbarui dengan nilai baru dan γ sebagai *discount rate* $(0 \leq \gamma \leq 1)$, umunnya sebagai penentu nilai *reward* dalam jangka panjang, jika nilai γ kecil, maka *agent* akan lebih mementingkan *reward* jangka pendek, dan sebaliknya. Sedangkan *TD-learning* memperbarui nilai *value function*-nya berdasarkan *state* selanjutnya.

Deep Q-Learning (DQN)

Deep Q-Learning merupakan *algoritma Q-Learning* yang menggunakan *neural network* untuk memperkirakan nilai dari fungsi *Q-value* , berbeda dengan *Q-Learning* yang menggunakan *Q-Table[7]*. *Q-Learning* menghasilkan nilai *Q-value* berdasarkan masukan *action-state* melalui *Q-Table*, dan Pada *Deep Q-Learning* menerima masukan dari *state* dan menghasilkan *Q-value* dari setiap *action* melalui *neural network*, *action* dipilih berdasarkan *Q-value* probabilitas tertinggi[7].

Pada penelitian ini menggunakan *Convolutional Neural Network (CNN)* karena masukan *observation* berupa gambar.

Double Deep Q-Network

DQN terdiri dari dua salinan bobot *Q-network*, satunya *online* yang dimana di *update* dalam setiap *step*, dan yang satunya *offline*, juga dipanggil dengan target *network*, yang dimana di *update* setelah beberapa (terkadang ribuan) *step*. Semenjak di dalam *DQN* ditentukan oleh target *Q-network*, oleh karena itu *action* akan ketinggalan *update* di dalam *Q-value*. Hal tersebut membuat penentuan *action* berbasis pada nilai maximum secara berulang-berulang, yang dimana membuatnya *stagnant* sampai target *network* di *update*. Hal tersebut memiliki efek membuat *agent* cenderung *over-exploit* dan *under-explore*, hal ini bisa menjadi kurang optimal karena *agent* itu akan dibatasi untuk beberapa *states* yang berdekatan dan mungkin tidak akan pernah mencapai *global optima*. Oleh karena itu *algoritma Double DQN [23]* ada untuk mengatasi permasalahan tersebut, yang dimana dari pada mempunyai dua salinan dari satu *Q-network* dan meng-*update* bobot setelah beberapa putaran yang ditentukan untuk menyinkronkan dua salinan bobot, *action* ditentukan berdasarkan *Q-network* yang bebeda *(online)*, dan *update* dilakukan sesuai nilai yang disarankan oleh target *Q-network (offline)*. Hal ini dapat menyelesaikan permasalahan *over-exploitation/under-exploration* pada *DQN*.

Dueling Deep Q-Network

Di dalam *DQN* dan *Double DQN* mempunyai dua *Q-network* yang bebeda, satu untuk mengambil *action*, dan yang lain untuk mengestimasi *Q-value*. Tetapi kedua *Q-network* itu *sequential*, namun *Dueling DQN* hanya mempunyai satu *CNN network*. Karena itu membuatnya tidak sepenuhnya *sequential*. Di dalam arsitektur *CNN* yang biasa digunakan di dalam *Dueling DQN*, *fully-connected layer* dibagi menjadi dua *network* yang memiliki *output node* dan *activation function* masing-masing. Dua *sub-network* dari arsitektur *CNN* disini mewakilkan *value* dan *advantage network*. *Value* disini itu bukan *Q-value (Qs,a)*, tapi merupakan *state (Vs) value*, yang dimana idealnya sama untuk *state* tertentu. Untuk mewakilkan ini, hanya satu *output node* dengan *linear activation* yang diperlukan. *Sub-network* lainnya, juga dipanggil *Advantage*, mewakilkan tambahan keuntungan dari pengambilan *action* diatas estimasi *state value* dari *network* pertama. *Network* ini mempunyai banyak *linear output node* sebanyak jumlah *action*, yang dimana mencerminkan *"advantage"* dari *action* tertentu, dan gabungan nilai *Qs,a*.

Deep Reinforcement Learning

Deep Reinforcement Learning merupakan sub-bidang dari *machine learning* yang menggabungkan *reinforcement learning (RL)* dengan *deep learning (DL)*. *RL* mempertimbangkan *agent* untuk belajar melalui *trial* dan *error*, dalam *Deep Reinforcement Learning* membuat *agent* dapat membuat keputusan dari data yang tidak terstruktur tanpa melakukan manual *engineering* pada *state space*. *Algoritma Deep Reinforcement Learning* dapat menerima masukan yang besar dan menentukan *action* untuk melakukan mengoptimasi pada *objective*.

Convolutional Neural Network

Convolutional Neural Networks (CNN) merupakan salah satu arsitektur *deep feed-forward artificial neural network* yang banyak diaplikasikan pada analisis citra *(Alom et al. 2018)*. *CNN* dibuat berdasarkan oleh proses biologi di mana pola konektivitas antar *neurons* menyerupai visual *cortex* pada binatang. *CNN* terdiri atas satu lapis masukan *(input layer)*, satu lapis keluaran *(output layer)*, dan sejumlah lapis *tersembunyi (hidden layers)*. *Hidden layers* biasanya berisi *convolutional layers*, *pooling layers*, *normalization layers*, *ReLU layer*, *fully connected layer*, dan *loss layer (Alom et al. 2018)*. *CNN* memiliki arsitektur yang disusun secara 3 dimensi: lebar *(width)*, tinggi *(height)*, dan kedalaman *(depth)* yang dimana berbeda dengan arsitektur *MLP* yang disusun secara dua dimensi. Setiap lapisan *CNN* mentransformasikan *volume* masukan tiga dimensi kedalam keluaran tiga dimensi.

Activation Function

Merupakan sebuah fungsi yang bagaimana jumlah tertimbang dari masukan *(weighted sum of the input)* bertransformasi menjadi *output*

dari *node* ke *node* di dalam lapisan jaringan. *activation function* dapat dibagi menjadi 2 tipe yaitu *Linear Activation Function* dan *Non-Linear Activation Function*

Overfitting

Overfitting mengacu pada model yang dimana model tersebut melatih *training data* terlalu baik. *Overfitting* terjadi ketika model mempelajari detail dan *noise* pada *training data* hingga dimana berdampak negatif terhadap performa model pada pada data baru. Yang berarti *noise* dan *random fluctuations* di *training data* diambil dan dipelajari sebagai konsep oleh model. Yang menjadi permasalahannya adalah dimana konsep tersebut tidak berlaku pada data baru yang berdampak pada kemampuan generalisasi data pada model. Salah satu cara untuk mengetahui sebuah model mengalami *overfitting* adalah dengan membandingkan nilai *loss* atau *error* ketika pelatihan dengan pengujian, jika nilai *loss* lebih kecil pada saat pelatihan dibandingkan pada saat pengujian, maka model tersebut mengalami *overfitting.*

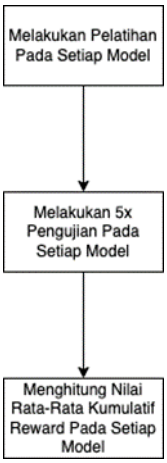
METODE PENELITIAN

Metode Penelitian

Pada penelitian ini penulis menggunakan metode penelitian eksperimen dengan pendekatan dan jenis penelitian kuantitatif. Menurut V. Wiratna Sujarweni (2014:39) penelitian kuantitatif adalah jenis penelitian yang menghasilkan penemuan-penemuan yang dapat dicapai (diperoleh) dengan menggunakan prosedur-prosedur statistik atau cara lain dari kuantifikasi (pengukuran)[14]. Penelitian ini mengukur nilai kumulatif *reward* dan *persentase* kenaikan dan penurunan pada nilai rata-rata *reward* pada setiap model *algoritma RL*. Dan penelitian eksperimen menurut Sugiyono (2011: 72) penelitian eksperimen adalah metode penelitian yang digunakan untuk mencari pengaruh perlakuan tertentu terhadap yang lain dalam kondisi yang terkendali[15]. Penelitian ini mengukur dan menguji performa model *deep reinforcement learning* yang berbasis *Q-Learning* seperti *DQN*, *Double DQN*, dan *Dueling DQN* dalam pengimplementasiannya pada *trading cryptocurrency*.

Metode Pengukuran Performa Model

Dalam *reinforcement learning*, pengukuran *performa algoritma* dapat diukur dengan cara seberapa baik *policy* yang ditemukan atau seberapa besar *reward* yang diterima ketika *algoritma* bertindak dan belajar. Salah satu cara menunjukan performa dari *algoritma reinforcement learning* adalah dengan menampilkan grafik nilai kumulatif *reward* terhadap jumlah dari total *step*. Dalam studi kasus penelitian ini *reward* merupakan nilai *profit* atau *loss* yang didapatkan oleh *agent*, dengan kata lain *reward* merupakan seberapa besar *agent* dapat membuat *profit* ketika melakukan aktivitas *trading*. Dalam penelitian ini pengukuran dilakukan dengan cara mengukur rata-rata nilai kumulatif *reward* pada setiap model, dengan kata lain mengukur *profit* yang didapatkan oleh *agent*. Semakin besar nilai *profit* yang didapatkan oleh *agent*, semakin baik performa model dalam melakukan aktivitas *trading*. Tahapan pengukuran model dapat dilihat pada Gambar 1.



Gambar 1. Tahapan Pengukuran Performa Model

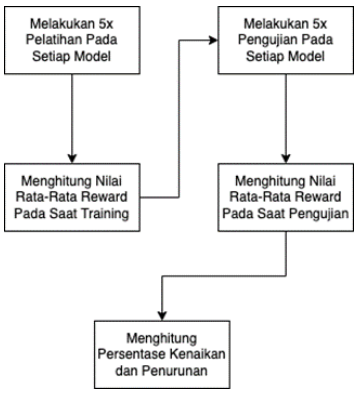
Metode Pengukuran Persentase Kenaikan dan Penurunan Performa Model

Persentase kenaikan atau penurunan merupakan perubahan secara relatif yang menunjukan besarnya kenaikan atau penurunan angka terakhir terhadap angka awal. Dengan mengukur persentase kenaikan dan penurunan pada model dapat mengetahui kenaikan dan

penurunan performa model ketika pada saat di *training* maupun *testing*. Persentase kenaikan atau penurunan dapat dihitung dengan menggunakan rumus :

(Nilai Awal – Nilai Akhir) / Nilai Awal x 100%

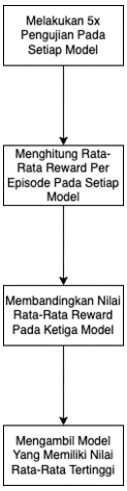
jika nilai awal lebih rendah, maka yang dihitung adalah persentase kenaikan. Persentase kenaikan dan penurunan dihitung pada rata-rata nilai *reward* pada setiap model. Tahapan pengukuran persentase kenaikan dan penurunan performa model dapat dilihat pada Gambar 2.



Gambar 2. Tahapan Mengukur Persentase Kenaikan dan Penurunan

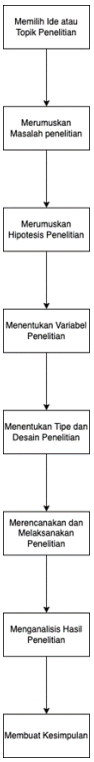
Metode Perbandingan Performa Model

Dengan membandingkan performa dari setiap model dapat menunjukan model yang mana memiliki performa yang terbaik diantara ketiga model. Untuk membandingkan performa setiap model dapat membandingkan rata-rata nilai reward pada setiap model ketika pengujian. Karena tujuan dalam melakukan *trading* adalah mendapatkan profit sebesar-besarnya, maka dari itu untuk menentukan model yang mana memiliki performa yang jauh lebih baik dari ketiga model adalah dengan mencari model yang memiliki nilai rata-rata *reward (profit)* tertinggi. Tahapan membandingkan performa model dapat dilihat pada Gambar 3.



Gambar 3. Tahapan Membandingkan Performa Model

Tahapan-Tahapan Metode Penelitian Eksperimen



Gambar 4. Tahapan Metode Penelitian Eksperimen

Pada penelitian ini alat dan bahan penelitian yang digunakan yaitu berupa perangkat keras dan perangkat lunak, pada perangkat keras yaitu komputer yang digunakan untuk melatih agent dengan spesifikasi :

Tabel 1. Spesifikasi Perangkat Keras

Processor	RAM	GPU	OS
Intel Core i7-8750H	24 GB	GTX 1060	Windows 11 Home

Pelatihan dan pengujian pada penelitian ini dilakukan dengan menggunakan *GPU Diskrit* yaitu *GTX 1060* dengan *6GB VRAM* dan *Compute Capability 6.1*. Dimana *GTX 1060* itu merupakan *sweet spot* dalam memilih *GPU* untuk *Deep Learning* sebelum *series RTX release* di pasaran. Perangkat keras yang digunakan bukanlah perangkat keras yang *high-end* dimana dapat melakukan komputasi pada *deep learning* dengan cepat. Walaupun begitu dengan spesifikasi seperti diatas sudah memenuhi kriteria minimal untuk melakukan komputasi pada *deep learning* maupun *deep reinforcement learning* dengan parameter yang cukup besar. Penggunaan *GPU Cloud* dapat menjadi pilihan, tetapi dengan harganya yang relatif mahal, membuat pilihan penggunaan *GPU Cloud* menjadi opsi yang kedua.

Jenis Data

Pada penelitian ini data yang diambil bersifat kuantitatif, dan merupakan *data internal* yang berasal dari *platform exchange* itu sendiri. Karena data ini penulis ambil dari *cryptodatadownload.com* menjadikan data ini merupakan data sekunder. Data primer dalam penelitian ini adalah data yang didapatkan dari hasil pelatihan dan pengujian pada setiap model. Selain itu data ini merupakan *historical dataset* yang dimana merupakan *time series* yang diambil dari tahun 2015 sampai 2021. *Dataset* ini memuat *date*, *open price*, *high price*, *low price*, *close price*, *volume* dll. Bentuk dataset dapat dilihat pada Gambar 5 berikut.

	Unix Timestamp	Date	Symbol	Open	High	Low	Close	Volume
0	1.637120e+12	11/17/2021 4:00	BTCUSD	59050.45	60831.30	58434.80	60352.88	1497.442949
1	1.637040e+12	11/16/2021 4:00	BTCUSD	60944.13	61558.53	58573.00	59050.45	2725.200365
2	1.636950e+12	11/15/2021 4:00	BTCUSD	65744.18	66340.74	60503.00	60944.13	1790.366778
3	1.636860e+12	11/14/2021 4:00	BTCUSD	64673.28	66200.00	63602.20	65744.18	528.499725
4	1.636780e+12	11/13/2021 4:00	BTCUSD	63805.12	65338.87	63409.49	64673.28	310.006539
...	...	...	...	...	...	...	...	...
2228	1.444622e+09	10/12/2015 4:00	BTCUSD	248.98	248.98	245.75	245.75	71.047743
2229	1.444536e+09	10/11/2015 4:00	BTCUSD	246.30	249.50	245.96	248.98	22.747091
2230	1.444450e+09	10/10/2015 4:00	BTCUSD	245.51	246.30	244.60	246.30	30.870549
2231	1.444363e+09	10/9/2015 4:00	BTCUSD	243.60	249.97	243.60	245.51	61.587068
2232	1.444277e+09	10/8/2015 4:00	BTCUSD	0.00	245.00	0.00	243.60	34.754703

Gambar 5. Bentuk Dataset

Dataset ini digunakan untuk mensimulasikan *environment trading crypto* yang dimana *environment* digunakan untuk melatih *trading agent* yang sudah dibuat

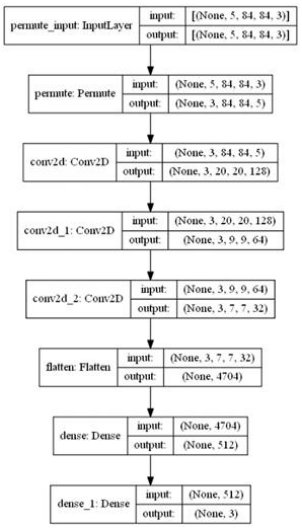
Sumber Data

Data dari penelitian ini diambil dari *cryptodatadownload.com* yang dimana merupakan situs untuk mengunduh data *trading cryptocurrency* pada berbagai *platform exchange* seperti *Gemini*, *coinbase*, *Bitsamp*, *Binance dll*.

Alur Penelitian

Gambar 6. Alur Penelitian

Tahap pertama pada penelitian ini adalah pengambilan data, data yang digunakan untuk melatih *agent* merupakan *historical data* yang dimana data ini diambil dari *cryptodatadownload.com* khususnya dari *Gemini Exchange*. *Data* yang diambil merupakan *real-world data*. Tahap selanjutnya adalah merancang dan membangun *trading environment* dengan menggunakan *framework OpenAI Gym*. Tahap ke tiga adalah merancang dan membangun *trading agent* menggunakan *framework tensorflow* dan *keras-rl2*. *Arsitektur CNN* yang digunakan pada penelitian ini dapat dilihat pada Gambar 7.



Gambar 7. Arsitektur CNN

Tahap berikutnya adalah melatih setiap *trading agent* dengan parmeter jumlah *step* **100.000** dengan **54** *episode*. Tahap terakhir yaitu menguji setiap *trading agent* dengan ketiga metode pengukuran yang sudah dijelaskan sebelumnya.

HASIL DAN PEMBAHASAN

Hasil Pengujian Pengukuran Performa Model

Pada proses pengujian dilakukan dengan total jumlah 10 episode pada masing-masing model.

Tabel 2. Profit Model DQN, Double DQN, dan Dueling DQN

Model	Opening Account Balance	Rata-Rata Profit	Percentage Increase %
DQN	100000	217764892,9	217665
Double DQN	100000	789788396,4	789688
Dueling DQN	100000	68604828,21	68505

Berdasarkan Tabel 2 dan pada hasil pengujian *Double DQN* memiliki performa yang jauh lebih baik dari kedua model lainnya dengan rata-rata nilai *reward* 789788396,4, hal itu disebabkan *Double DQN* memiliki dua *Q-network* yang dimana ketika satu *Q-network* melampaui estimasi *(overestimate)* pada *Q-value* maka dengan menggunakan *Q-network* kedua dapat mengkontrol bias tersebut ketika mengambil nilai maksimum. Sehingga dapat terhindar dari permasalahan *maximization bias*, Berbeda halnya dengan *DQN* yang memiliki rata-rata nilai *reward* **217764892,9** yang dimana hanya memiliki satu *Q-network* yang rawan akan permasalahan *maximization bias*. Pada urutan terakhir terdapat *Dueling DQN* yang memiliki nilai rata-rata *reward* **68604828,21**, yang jauh lebih kecil dari kedua model lainnya. Secara teori *Dueling DQN* mempunyai banyak keunggulan dari kedua model lainnya, salah satunya adalah *Dueling DQN* dapat menyelesaikan permasalahan *overoptimistic* pada *Q-value* seperti Double DQN, tetapi pada *Double DQN* karena menggunakan dua *Q-network*, maka dari itu harus melatih kedua *Q-network* tersebut secara terpisah sehingga memiliki dua kali waktu konvergensi. Jika dilihat dari data *reward* per *episode* nya, nilai nya cukup konsisten di angka **68000000**-an yang dimana artinya konsistensi ini membuat model dekat untuk mencapai konvergensi. Untuk mendapatkan hasil yang lebih baik disarankan untuk melakukan *hypertunning* pada parameter seperti menggunakan *Boltzmann Q Policy* yang pada beberapa kasus lebih baik dari pada menggunakan *Epsilon Greedy*, atau menambahkan nilai *target network update* menjadi **1000** *(OpenAI Baseline DQN)* atau **10000** berdasarkan referensi dari paper *Deepmind Nature[25]*.

Hasil Pengukuran Persentase Kenaikan dan Penurunan Nilai Rata-Rata Reward Model Ketika Training dan Testing

Pada proses *training*, dilakukan pelatihan model dengan parameter *number of step* **100.000** dengan jumlah **54** *episode* pada ketiga model *DQN*, *Double DQN*, dan *Dueling DQN***.**

Tabel 3. Hasil Pengukuran Persentase Kenaikan dan Penurunan

Model	Training (Mean Reward)	Testing (Mean Reward)	Percentage Increase or Decrease (%)
DQN	676360702,7	217764892,9	-68
Double DQN	618302508,7	789788396,4	28
Dueling DQN	505355050	68604828,21	-86

Pada Tabel 3 hasil pengukuran persentase kenaikan atau penurunan pada performa model pada saat *training* dan *testing*, hanya *Double DQN* yang memiliki persentase kenaikan yaitu sebesar **28%**. Setiap model mengalami kenaikan loss yang tinggi, tetapi pada *Double DQN* menunjukan nilai *loss* yang mulai turun, berbeda dengan *DQN* yang menunjukan tanda-tanda nilai *loss* yang terus naik. Nilai *loss* yang naik pada ketiga model disebabkan oleh nilai *target network update* yang kecil sehingga menyebabkan *training* dari *DQN* tidak stabil. Maka dari itu untuk mengatasi masalah tersebut dapat dengan menambahkan nilai *target network update* menjadi **1000** *(OpenAI Baseline DQN)* atau **10000** menurut referensi dari *paper Deepmind Nature[25]*. Penurunan nilai rata-rata *reward* pada *DQN* dan *Dueling DQN* disebabkan oleh model yang mengalami *overfitting*, salah satu penyebab *overfitting* adalah karena melakukan *training model* yang terlalu lama. Dalam *reinforcement learning* hal tersebut disebut dengan *exploration-exploitation trade-off*, dimana agent yang di *training* belajar untuk menyelesaikan tugas yang spesifik dan *policy* nya *overfitting*, untuk mengatasi masalah tersebut dapat dengan meningkatkan *exploration* seperti menggunakan distribusi pada *Q-value* seperti *Softmax Activation Function* atau menggunakan *Boltzmann-style exploration* yaitu *Boltzmann Q Policy*.

Hasil Perbandingan Performa Masing-Masing Arsitektur Pada Setiap Model

Tabel 4. Hasil Perbandingan Performa Model

Model	Mean Reward	Number of Test
DQN	217764892,9	5
Double DQN	789788396,4	5
Dueling DQN	68604828,21	5
VGG16 + DQN	1599,56854	5
VGG16 + Double DQN	1434068,654	5
VGG16 + Dueling DQN	814,30104	5

Pada penelitian ini performa model ditentukan dengan seberapa besar nilai rata-rata *reward* yang diterima oleh model, semakin besar nilai rata-rata *reward* pada suatu model semakin baik performa dari model tersebut, dikarenakan semakin besar nilai *reward* yang didapat semakin besar juga *profit* yang didapat ketika melakukan *trading*. Berdasarkan hasil pengujian dan hasil perbandingan pada Tabel 4, *Double DQN* mempunyai performa yang jauh lebih baik dari kedua model, seperti yang sudah dijelaskan sebelumnya hal tersebut dikarenakan *Double DQN* dapat mengatasi permasalahan *maximization bias* dengan memiliki dua *Q-network*, yang dimana *DQN* sangat rawan dengan permasalahan tersebut. Walaupun dalam teori model *Dueling DQN* mempunyai banyak keunggulan dari model *Double DQN*, yang dimana hal tersebut disebabkan oleh *overfitting model* atau dalam *reinforcement learning* disebut dengan *exploration-exploitation trade-off* sehingga menyebabkan generalisasi dan performa model jauh lebih turun dari pada model lainnya. Hal tersebut bisa diatasi dengan menggunakan *Softmax Activation Function* pada *Q-value*, menggunakan *Boltzmann-style exploration*, dan juga melakukan *hypertunning* pada parameter. Sebagai tambahan, pengujian juga dilakukan menggunakan arsitektur *VGG16* dengan kombinasi dengan model *DQN*, *Double DQN* dan *Dueling DQN*. Arsitektur *VGG16* yang digunakan pada pengujian ini memiliki

perubahan atau modifikasi pada *pooling layer*, yang dimana *pooling layer* digantikan dengan *strides* pada *convolutional layer* yang dimana sama-sama berfungsi untuk melakukan *down-sampling* pada masukan. Penggunaan *strides* pada pada *VGG16* bertujuan untuk menyesuaikan dengan ukuran dari masukan model, selain hal itu menggunakan *strides* pada *convolutional layer* dapat mengurangi *computational cost[24]*. Berdasarkan hasil pengujian yang dapat dilihat pada Tabel 4, jika dibandingkan dengan arsitektur yang dibuat pada penelitian ini hasil dari *VGG16* sangatlah buruk. Sama seperti sebelumnya *VGG16 + Double DQN* memiliki performa yang lebih baik dari kombinasi *VGG16* dengan model yang lainnya, dengan nilai rata-rata *reward* **1434068,654**, walaupun tidak sebanding dengan *Double DQN* dengan *arsitektur kustom* yang memiliki nilai rata-rata *reward* sebesar **789788396,4**. Hal tersebut terjadi karena beberapa hal seperti arsitektur *VGG16* tidak didesain untuk di implementasikan pada *reinforcement learning*, dan arsitektur *VGG16* terlalu kompleks sehingga model mengalami *overfitting*. Selain hal itu *VGG16* membutuhkan komputasi yang berat untuk melakukan *training* dengan parameter yang jauh lebih banyak dibandingkan *custom* arsitektur model yang dibuat pada penelitian ini.

Pembahasan Permasalahan Kecepatan Training Pada Model

Proses training pada model yang menggunakan *custom* arsitektur menghabiskan durasi rata-rata **7** jam untuk melatih satu model. Waktu durasi pada saat training dapat dilihat pada Tabel 5.

Tabel 5. Durasi Proses Training Pada Custom Arsitektur

No	Model	Duration	Number of Steps
1	Deep Q-Network (DQN)	7 Jam 42 Menit 16 Detik	100.000
2	Double Deep Q-Network (DQN)	7 Jam 39 Menit 23 Detik	100.000
3	Dueling Deep Q-Network	7 Jam 21 Menit 51 Detik	100.000
Total		**22 Jam 44 Menit**	

Sedangkan proses *training* dengan menggunakan arsitektur *VGG16* rata-rata menghabiskan waktu ***14 jam*** untuk melatih satu buah model, dengan total menjadi ***43 jam***, dapat dilihat pada Tabel 5. Dimana hampir dua kali lipat dari waktu dengan menggunakan *custom* arsitektur. Hal tersebut dikarenakan *VGG16* memiliki parameter sebesar **15.142.595** dibandingkan *custom* arsitektur yang hanya memiliki **2.601.187** parameter yang harus dilatih.

Tabel 6. Durasi Proses Training Pada VGG16

No	Model	Duration	Number of Steps
1	Deep Q-Network (DQN) + VGG16	14 Jam 26 Menit 29 Detik	100.000
2	Double Deep Q-Network (DQN) + VGG16	14 Jam 45 Menit 46 Detik	100.000
3	Dueling Deep Q-Network + VGG16	14 Jam 36 Menit	100.000
Total		**43 Jam 48 Menit 16 Detik**	

Proses *training* yang cukup lama disebabkan oleh keterbatasan kekuatan komputasi pada perangkat keras atau banyaknya parameter yang harus dilatih. Penggunaan perangkat keras yang berbeda dapat menghasilkan performa yang berbeda pada masing-masing model. Yang dimana permasalahan ini merupakan salah satu kendala dalam penelitian ini. Kendala dengan perangkat keras bisa saja diatasi dengan menggunakan *GPU Cloud* seperti *GPULab*, *Gradient dll*. Tapi berlangganan *GPU Cloud* dapat menguras kantong yang dimana harganya relatif mahal dalam jangka waktu tertentu.

SIMPULAN

Hasil pengukuran performa model dilakukan dengan melatih setiap model dan melakukan pengujian sebanyak **5x**, dan kemudian menghitung nilai rata-rata *reward* setiap model. Hasil dari pengukuran performa, *Double DQN* memiliki performa yang jauh lebih baik dari kedua model lainnya, karena *Double DQN* memiliki dua *Q-network* untuk mengatasi permasalahan *maximization bias* dibandingkan dengan *DQN*. Tapi secara teori *Dueling DQN* memiliki banyak keunggulan dibandingkan *Double DQN* yang dimana seharusnya memiliki performa yang lebih baik, ada beberapa faktor yang menyebabkan *Dueling DQN* memiliki performa yang paling buruk dibandingkan model lainnya, seperti model yang mengalami *overfitting*, parameter yang perlu di *hypertunning* dan pemilihan nilai target *network update* yang kecil.

Hasil pengukuran persentase kenaikan atau penurunan dilakukan dengan mengukur nilai rata-rata *reward* ketika model dilatih dan diuji dengan menggunakan rumus:

(Nilai Awal – Nilai Akhir) / Nilai Awal x 100%

Berdasarkan hasil dari pengukuran hanya model *Double DQN* yang mempunyai persentase kenaikan yaitu sebesar ***28%***, kedua model lainnya memiliki persentase penurunan yang cukup besar yang dimana *DQN* memiliki persentase penurunan ***-68%*** dan *Dueling DQN* sebesar ***-86%.*** Hal tersebut disebabkan oleh kedua model yang mengalami *overfitting* sehingga performa menjadi menurun.

Perbandingan performa model dilakukan dengan cara membandingkan nilai rata-rata *reward* setiap model pada saat pengujian. Model yang memiliki nilai rata-rata *reward* tertinggi mempunyai performa yang lebih baik dari model lainnya. Berdasarkan dari hasil perbandingan, model *Double DQN* mempunyai performa yang lebih baik dari model lainnya, seperti yang sudah dijelaskan sebelumnya model *Double DQN* memiliki dua *Q-network* untuk mengatasi permasalahan *maximization bias*, yang dimana permasalahan tersebut sangat rawan terhadap model *DQN*. *DQN* dan *Dueling DQN* memiliki performa yang buruk karena model yang mengalami *overfitting* yang menyebabkan generalisasi dan performa model yang menurun. Sebagai tambahan juga dilakukan pengujian perbandingan performa dengan menggunakan arsitektur *VGG16*, pada penelitian ini pada *VGG16* adanya perubahan atau modifikasi pada *pooling layer* dan digantikan dengan *strides* pada *convolutional layer*. Hasil dari perbandingan, dimana *VGG16* mempunyai performa yang jauh lebih buruk dari performa arsitektur yang dibuat kustom pada penelitian ini.

DAFTAR PUSTAKA

[1] Widya, Fathonah. (2020). Tren Trading Crypto, Pahami Kelemahan & Bedanya dengan Pendanaan P2P. Diakses dari https://tanifund.com/blog/berita/tren-trading-crypto-pahami-kelemahan-bedanya-dengan-pendanaan-p2p/

[2] Rahmalia, Nadiyah. (2021, April 19). Tertarik dengan Trading? Pahami Dulu Arti, Jenis-Jenis, dan Plus-minusnya. Diakses dari https://glints.com/id/lowongan/trading-adalah/#.YZ8V29BBy00

[3] Nugroho, Adi. (2021, Agustus 14). Trading Bot Solusi Praktis Trading Anda, Begini Cara Kerjanya. Diakses dari https://www.foreximf.com/blog/trading/trading-bot

[4] Ivan. (2020, Februari 08). Is it Possible To Predict Stock Prices With A Neural Network. Diakses dari https://towardsdatascience.com/is-it-possible-to-predict-stock-prices-with-a-

neural-network-d750af3de50b

[5] Lapan, M. (2020). Deep reinforcement learning hands-on. Packt publishing.

[6] Wijaya, D. A., & Darmawan, O. (2017). Blockchain Dari Bitcoin Untuk Dunia. Jakarta: Jasakom.

[7] Winder, P. (2020). Reinforcement Learning. O'Reilly Media.

[8] Shoham, Y., & Powers, R. (2010). Multi-Agent Learning II: Algorithms.

[9] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

[10] Hayes, Adam. (2021, Oktober 30). Trade. Diakses dari https://www.investopedia.com/terms/t/trade.asp

[11] Cholissodin, Imam & Setyawan, Gembong Edhi. (2019). Pengembangan Reinforcement Learning (RL) Single-Agent Menggunakan Improve Q-Learning.

[12] Tummon, E., Raja, M. A., & Ryan, C. (2020, November). Trading Cryptocurrency with Deep Deterministic Policy Gradients. In International Conference on Intelligent Data Engineering and Automated Learning (pp. 245-256). Springer, Cham.

[13] Suharsimi Arikunto, S. (2010). Prosedur Penelitian suatu pendekatan praktik. Jakarta: Rineka Cipta.

[14] Sujarweni, V. W. (2014). Metode penelitian: Lengkap, praktis, dan mudah dipahami.

[15] Sugiyono, P. (2011). Metodologi penelitian kuantitatif kualitatif dan R&D. Alpabeta, Bandung.

[16] Lucarelli, G., & Borrotti, M. (2019, May). A deep reinforcement learning approach for automated cryptocurrency trading. In IFIP International Conference on Artificial Intelligence Applications and Innovations (pp. 247-258). Springer, Cham.

[17] Betancourt, C., & Chen, W. H. (2021). Reinforcement Learning with Self-Attention Networks for Cryptocurrency Trading. Applied Sciences, 11(16), 7377.

[18] Wen, W., Yuan, Y., & Yang, J. (2021). Reinforcement Learning for Options Trading. Applied Sciences, 11(23), 11208.

[19] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.560

[20] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. nature, 518(7540), 529-533.

[21] Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., & Freitas, N. (2016, June). Dueling network architectures for deep reinforcement learning. In International conference on machine learning (pp. 1995-2003). PMLR.

[22] Van Hasselt, H., Guez, A., & Silver, D. (2016, March). Deep reinforcement learning with double q-learning. In Proceedings of the AAAI conference on artificial intelligence (Vol. 30, No. 1).

[23] Sewak, M., Sahay, S. K., & Rathore, H. (2020, October). Value-Approximation based Deep Reinforcement Learning Techniques: An Overview. In 2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA) (pp. 379-384). IEEE.

[24] Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806.

[25] Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., ... & Silver, D. (2020). Mastering atari, go, chess and shogi by planning with a learned model. Nature, 588(7839), 604-609.

[26] Palanisamy, Praveen. (2021, January). TensorFlow 2 Reinforcement Learning Cookbook: Over 50 recipes to help you build, train, and deploy learning agents for real-world applications. Packt Publishing.

[27] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

[28] Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning (pp. 448-456). PMLR.