Perancangan Purwarupa Mobil Otonom Menggunakan Simulator *Udacity* dengan Metode Convolutional Neural Network

Alexander Parikesit[1)], **Edwin Lesmana Tjiong**[2)]

[1,2)] Informatika, Fakultas Ilmu Komputer dan Desain, Institut Teknologi dan Bisnis Kalbis
Jalan Pulomas Selatan Kav. 22, Jakarta 13210
Email: codengineer15@gmail.com
Email: edwin.tjiong@kalbis.ac.id

Abstract: *This research presents the process and development of an autonomous vehicle prototype using Udacity Simulator with the Convolutional Neural Network (CNN) method. This research uses NVIDIA CNN Architecture for its convolutional neural network architecture. The objective of this research is to design and generate a machine learning model to run the Udacity simulator in autonomous mode, which allowing car objects on the simulator to move autonomously. The proposed approach involves training a CNN model using a labeled dataset of tracks images captured through training mode. The output of the CNN model is then used to control the steering and acceleration commands of the vehicle. The performance of the machine learning model is evaluated using MSE, RMSE, and MAE parameters. In addition, it is also evaluated on its ability to navigate one of the tracks in the simulator.*
Keywords: *Autonomous Vehicle, Udacity Simulator, Convolutional Neural Network, Machine Learning Model, Steering Angle*

Abstrak: *Penelitian ini menyajikan proses dan pengembangan prototipe kendaraan otonom menggunakan simulator Udacity dengan metode Convolutional Neural Network (CNN). Penelitian ini menggunakan NVIDIA CNN Architecture untuk arsitektur convolutional neural networknya. Tujuan dari penelitian ini adalah merancang dan menghasilkan model pembelajaran mesin untuk menjalankan simulator Udacity dalam mode autonomous, yang memungkinkan objek mobil pada simulator bergerak secara otonom.Pendekatan yang diusulkan melibatkan pelatihan Model CNN menggunakan dataset berlabel gambar lintasan yang diambil melalui mode training. Output dari model CNN kemudian digunakan untuk mengontrol perintah kemudi dan akselerasi kendaraan. Kinerja model pembelajaran mesin dievaluasi menggunakan parameter MSE, RMSE, dan MAE. Selain itu, evaluasi juga dilakukan terhadap kemampuannya untuk menavigasi salah satu lintasan yang terdapat di simulator.*
Kata kunci: *Kendaraan Otonom, Simulator Udacity, Jaringan Saraf Konvolusi, Model Pembelajaran Mesin, Sudut Kemudi*

I. PENDAHULUAN

Secara teori kendaraan otonom – secara spesifik akan disebut mobil otonom – atau *Autonomous Vehicle* (AV), adalah kendaraan yang memiliki sistem autopilot dan dapat berpindah dari satu lokasi ke lokasi lain tanpa bantuan pengemudi. Menurut definisi lain, kendaraan otonom adalah kendaraan yang dapat merasakan lingkungannya (*sensing of its environment*) dan menavigasi sendiri [1]. Sehingga secara teoritis, sistem kendaraan otomatis hanya dapat disebut "otonom" jika mampu menangani semua tugas mengemudi yang dinamis di semua lingkungan berkendara [2].

Mobil otonom pertama muncul tahun 1980-an, sebagai proyek *Nav Lab* dan *ALV* dari *Carnegie Mellon University* pada 1984 dan *Mercedes Benz* dan *Bundeswehr University Munich* pada tahun 1987 [1].

Riset tentang mobil otonom sudah banyak dicoba setidaknya selama sedekade terakhir. Pada tahun 2010

perusahaan Google telah mengembangkan kendaraan otonomnya, pada Juli 2013 Vislab mendemonstrasikan *BRAiVE* yaitu kendaraan yang bergerak secara otonom pada rute kombinasi yang terbuka untuk lalu lintas umum, dan pada tahun 2014 *Tesla Motors*sudah memasang sistem *autopilot*, merupakan asisten mengemudi semi otonom pada seluruh kendarannya.

Penelitian ini berfokus pada perancangan purwarupa mobil otonom menggunakan simulator *Udacity* dengan menggunakan metode *Convolutional Neural Network* (CNN). Metode CNN dipilih karena simulator *Udacity* menggunakan citra (*images*) sebagai data untuk melakukan proses pelatihan [3]. Meskipun menggunakan citra sebagai data pelatihannya, simulator *Udacity* juga menyediakan data CSVfile yang memuat informasi seperti, nama file dari citra yang berhasil diambil dan informasi lain terkait pergerakan mobil seperti *steering*, *throttle*, *reverse*, sampai dengan *speed*.

Selain memuat informasi lain terkait pergerakan mobil, data CSV file digunakan untuk menentukan *feature* yang akan menjadi *output* dari model CNN. Dalam kasus ini sudut kemudi (*steering angle*) akan digunakan sebagai *feature* untuk merancang model CNN. Untuk arsitektur CNN yang digunakan dalam penelitian ini adalah arsitektur yang dikembangkan oleh NVIDIA yaitu, NVIDIA CNN *Architecture* yang digunakan untuk merancang mobil otonom milik mereka [4].

A. Kecerdasan Buatan

Kecerdasan buatan atau dalam Bahasa Inggris disebut sebagai *Artificial Intelligence* (AI) adalah salah satu bidang studi dalam ilmu komputer. Bidang ini mencakup pembuatan program komputer untuk melakukan tugas- tugas yang seharusnya membutuhkan kecerdasan manusia. Algoritma AI dapat digunakan untuk memecahkan masalah dalam pembelajaran, persepsi, pemecahan masalah, pemahaman bahasa, dan/atau penalaran logis [5].

B. Mobil Otonom

Mobil otonom (*self-driving cars* atau *autonomous vehicle*) adalah kendaraan yang dapat merasakan lingkungannya (*sensing of its environment*) dan menavigasi sendiri [1]. Pada tahun 2014, *SAE International* (*Society of Automotive Engineers International*), merilis *J3016* yaitu sebuah sistem klasifikasi untuk mobil otonom dengan enam level otomasi. Di tahun 2016, SAE mengubah klasifikasi standarnya tersebut yang kemudian dikenal sebagai *SAE J3016 Standard, 2016*. Keenam level otomasi tersebut di antaranya:

1. ***Level 0 – No Automation***: di level ini mobil masih dikemudikan manual secara penuh oleh pengemudi melalui setir, gas,dan rem.
2. ***Level 1 – Driver Assistance***: di level ini terdapat sistem *driver assistance* yang memungkinkan pengemudi untuk menjalankan mobil tanpa mengontrol pedal gas secara manual dalam kondisi tertentu. Pengemudi harus mengawasi dan mengambil alih sistem ketika kendaraan memintanya. Untuk setir dan rem masih dikontrol oleh pengemudi.
3. ***Level 2 – Partial Assistance***: di level ini mobil dapat mengontrol setir, gas, dan rem namun tetap dibutuhkan seorang pengemudi untuk mengambil alih sistemketika mobil melihat sebuah objek seperti lampu lalu lintas, kendaraan di sekitar, cuaca, dan kondisi jalan.
4. ***Level 3 – Conditional Assistance***: di level ini mobil dapat mengontrol setir, gas, dan rem. Mobil juga dapat memantau objek ada di sekitarnya. Namun, tetap dibutuhkan seorang pengemudi untuk mengambil alih sistemketika sistem memintanya.
5. ***Level 4 - High Automation***: di level ini mobil dapat mengontrol setir, gas, dan rem dan dapat memantau objek yang

ada di sekitarnya namun dalam jangkauan yang lebih luas. Meskipun begitu, tetap dibutuhkan seorang pengemudi untuk mengambil alih sistem ketika terjadi distraksi seperti cuaca buruk.

6. **Level 5 – *Full Automation***: di level ini mobil dapat mengontrol setir, gas, dan rem dan dapat memantau objek yang ada di sekitarnya. Pada level ini pengemudi tidak melakukan intervensi apapun terhadap mobil.

C. Simulator *Udacity*

Simulator *Udacity* adalah aplikasi simulator *open-source* yang dibuat dan dikembangkan oleh *Udacity*.

Aplikasi simulator ini menyediakan *environment* yang dapat digunakan oleh pengguna untuk merancang atau mengimplementasikan purwarupa mobil otonom menggunakan algoritma *neural network*. Pada simulator *Udacity* terdapat dua mode, yaitu mode *training* (mode manual) dan mode *autonomous* (mode otonom) [6].

1. **Mode *Training***: di mode ini pengguna mengoperasikan kendaraan secara manual di lintasan untuk merekam gaya mengemudinya. Melalui mode ini dihasilkan data berupa citra dan CSV.
2. **Mode *Autonomous***: di mode ini pengguna dapat melakukan pengujian dari model pembelajaran mesin menggunakan data citra yang sudah dilatih untuk mengukur seberapa efektif model tersebut dapat mengarahkan mobil secara otomatis.

Simulator *Udacity* memiliki 2 (dua) jenis lintasanyang dapat digunakan untuk merancang dan melatih model pembelajaran mesin yaitu, *the lake track* dan *the jungle track*.

Gambar 1 Dua Jenis Lintasan di Simulator *Udacity*

D. Machine Learning

Machine learning adalah cabang ilmu komputer yang secara luas bertujuan untuk memungkinkan komputer belajar tanpa memprogramnya secara langsung. Komputer "belajar" di dalam *machine learning* dengan meningkatkan kinerja mereka pada tugas-tugas tertentu melalui pengalaman. Arti dari pengalaman adalah bagaimana komputer melakukan kinerja pada setiap tugas yang diberikan dengan menyesuaikan data [7].

E. Deep Learning

Deep learning adalah bagian dari *machine learning* yang memungkinkan komputer untuk belajar dari pengalaman (serangkaian pelatihan) menggunakan data. Cara tersebut dilakukan menggunakan *artificial neural network* dan algoritma *machine learning*. Pada arsitektur *artificial neural network*, model *deep learning* digambarkan sebagai jaringan yang terdiri lebih dari satu *layer*. Secara umum, model dari *deep learning* memiliki tiga jenis *layer*: *input layer* (*received data*), *hidden layer* (*extracts patterns*), *output layer* (*produces the results*). Setiap *output* dari satu *layer* akan digunakan untuk *layer* berikutnya [8].

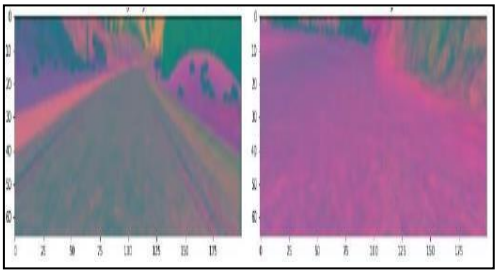
F. Convolutional Neural Network

Convolutional Neural Network (CNN) merupakan salah satu pengembangan dari jaringan saraf tiruan yang terinspirasi dari jaringan saraf manusia. CNN merupakan pengembangan dari *multilayer perceptron* yang didesain untuk mengolah data dua dimensi. Pada model CNN, setiap *neuron* direpresentasikan dalam bentuk dua dimensi, tidak seperti *multilayer perceptron* yang setiap *neuron*-nya hanya berukuran satu dimensi [9].

G. NVIDIA CNN Architecture

NVIDIA CNN *Architecture* terdiri dari sebuah *normalization layer*, 5 *convolutional layer*, dan 3 *fully connected layer*. Untuk citra yang akan dimasukkan ke dalam arsitektur CNN ini akan dikonversi terlebih dahulu ke bidang warna (*color space*) *YUV* [4].

Color space YUV digunakan untuk memproses citra karena lebih efektif dan praktis [3].



Gambar 2 Contoh *Color Space* YUV

II. METODE PENELITIAN

Penelitian ini menggunakan beberapa teori pendukung yang berkaitan dengan perancangan mobil otonom. Untuk pengembangan perangkatlunak, penelitian ini akan menggunakan *Software Development Life Cycle* (SDLC) *Incremental*.

A. Metode *Incremental*

Metode pengembangan perangkat lunak *incremental* adalah sebuah model yang menggabungkan proses *linear* dan *parallel*. Proses *linear* adalah proses untuk mengembangkan perangkat dari awal hingga akhir. Sementara proses *parallel* merupakan kumpulan proses *linear* yang berurutan, dimana masing-masing proses *linear* ini menghasilkan pengembangan yang bersifat lebih lanjut dari proses *linear* sebelumnya. Metode ini merupakan model yang menggunakan proses penambahan sedikit demi sedikit yang berfokus pada pengiriman modul yang bersifat operasional pada setiap tahapan [10].

B. Incremental I

Tahap *Incremental I* dilakukan untuk membuat model pembelajaran mesin yang digunakan untuk mengontrol pergerakan mobil secara otomatis saat menggunakan mode *autonomous* pada simulator *Udacity*.

Berikut adalah tahapan dari Incremental I:

1. Analisis

Penelitian ini dimulai dengan proses pengumpulan dan pembuatan *dataset* berupa data citra dan data CSV. *Dataset* dibuat dengan mode *training* menggunakan fitur *recording*.

Fitur *recording* akan mengambil video pergerakan mobil dan kemudian video tersebut akan dipotong *frame by frame* untuk menghasilkan data citra dari bagian tengah (*center*), kiri (*left*), dan kanan (*right*) mobil.

Di mode *training* pengguna akan menjalankan objek mobil pada simulator secara manual menggunakan *keyboard* sebagai *input*.

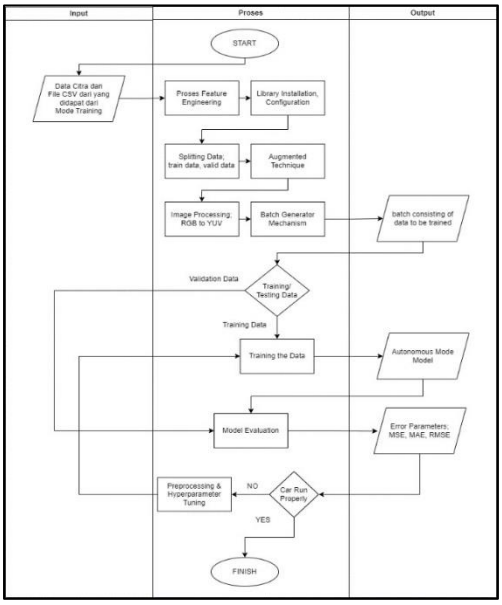
Untuk *dataset* dalam penelitian ini dibuat menggunakan lintasan bernama *The Jungle Track*. Total citra yang berhasil dikumpulkan sebanyak 22.815 dengan ukuran masing- masing yaitu 320×160 *pixels*.

Gambar 3 Contoh Data Citra untuk *Dataset*

2. Desain

Tahap desain akan berfokus pada perancangan arsitektur CNN, pemorsesan citra, teknik augmentasi, dan perancangan mekanisme *batch generator* untuk memproses data dalam *batch* (kelompok) yang lebih kecil saat model dilatih.

Pembuatan model pembelajaran mesin dalam penelitian ini menggunakan NVIDIA CNN *Architecture* yang terdiri dari sebuah 5 *convolutional layer* dan 3 *fully connected layer*. Untuk pelatihan model akan menggunakan 10 hingga 15 *epochs*. Karena ukuran *dataset* yang digunakan besar, pada saat proses pelatihan model akan menggunakan mekanisme *batch generator*, yaitu mekanisme untuk memproses data dalam *batch* (kelompok) yang lebih kecil, selain itu *batch generator* juga digunakan untuk melakukan manajemen memori yang baik untuk ukuran data yang besar.



Gambar 4 *Flowchart Incremental* I

3. Implementasi

Dalam tahap implementasi dilakukan proses penerapan desain yang sudah dirancang ke dalam *source code*. Model pembelajaran mesin akan dibuat menggunakan Bahasa Pemrograman Python 3.9 dengan bantuan *library keras* untuk merancang model CNN, *OpenCV* untuk melakukan pemrosesan citra, dan *imgaug* untuk melakukan teknik augmentasi.

4. Pengujian

Tahap pengujian dilakukan dengan parameter *MSE* (*Mean Square Error*), *RMSE* (*Root Mean Square Error*), dan *MAE* (*Mean Absolute Error*). Ketiga parameter tersebut digunakan sebagai evaluasi dari model pembelajaran mesin yang dihasilkan. Selain itu, model pembelajaran mesin akan diuji coba langsung di simulator *Udacity* menggunakan mode *autonomous*.

C. Incremental II

Tahap *Incremental II* dilakukan untuk membuat *script* yang akan digunakan untuk menghubungkan model dengan aplikasi simulator *Udacity*.

Berikut adalah tahapan dari Incremental II:

1. Analisis

Pada tahap ini dilakukan proses Analisa terkait bagaimana menghubungkan model pembelajaran mesin dengan aplikasi simulator *Udacity* menggunakan komunikasi *client server*.

2. Desain

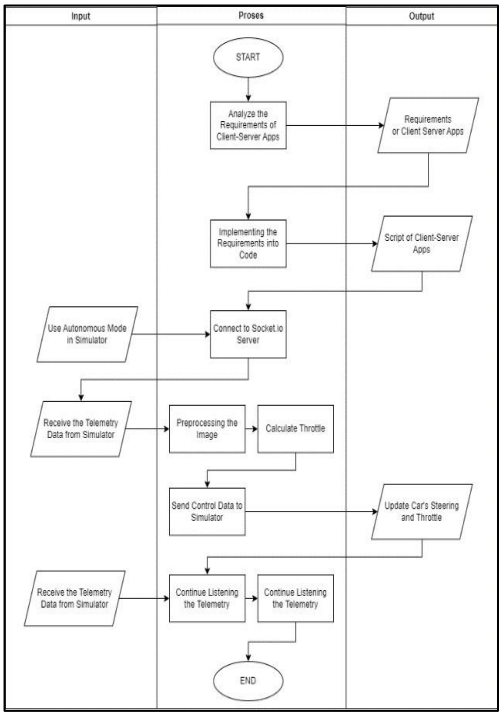
Pada tahap desain dirancang sebuah *script* untuk memenuhi kebutuhan sesuai dengan analisis.

3. Implementasi

Pada tahap implementasi dilakukan proses penerapan desain yang sudah dirancang ke dalam *source code*. *Script* akan dibuat menggunakan Bahasa Pemrograman Python 3.9 dengan bantuan *library socketio* dan *Flask*. *Script* untuk menghubungkan model pembelajaran mesin dan aplikasi simulator *Udacity* akan dinamai *drive.py*.

4. Pengujian

Pengujian *script* akan dilakukan menggunakan skema *whitebox testing* untuk menghubungkan model pembelajaran mesin dengan aplikasi simulator *Udacity*.



Gambar 5 *Flowchart Incremental* II

III. HASIL DAN PEMBAHASAN

A. Incremental

Untuk hasil dan pembahasan pada *Incremental I* model pembelajaran mesin akan dievaluasi menggunakan metrics *MSE*, *RMSE*, dan *MAE*. Selain itu model akan dijalankandengan mode *autonomous* pada aplikasi simulator *Udacity*.

1. Pengujian Pertama

Model CNN dilatih menggunakan data pelatihan yang dikelompokan dalam ukuran *batch* sebesar 100. Jumlah *batch* yang diproses dalam satu *epochs* adalah sebesar 300. Untuk pengujian pertama, model dilatih dengan menggunakan 10 *epochs*. Jadi pada pengujian pertama, model dilatih menggunakan 30.000 data pelatihan.

Summary dari arsitektur model CNN di pengujian pertama adalah sebagai berikut:

```
model = cnn_model()
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 31, 98, 24)        1824

conv2d_1 (Conv2D)            (None, 14, 47, 36)        21636

conv2d_2 (Conv2D)            (None, 5, 22, 48)         43248

conv2d_3 (Conv2D)            (None, 3, 20, 64)         27712

flatten (Flatten)            (None, 3840)              0

dense (Dense)                (None, 100)               384100

dense_1 (Dense)              (None, 50)                5050

dense_2 (Dense)              (None, 10)                510

dense_3 (Dense)              (None, 1)                 11

=================================================================
Total params: 484,091
Trainable params: 484,091
Non-trainable params: 0
_________________________________________________________________
None
```

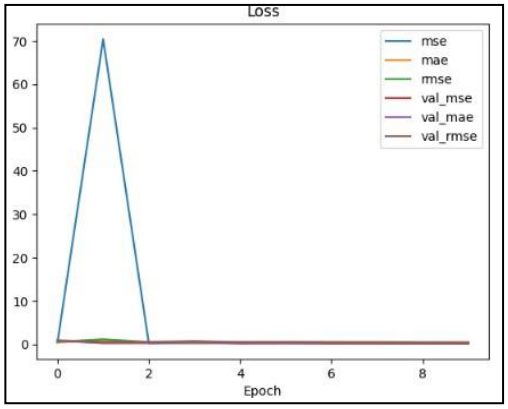
Gambar 6 *Summary Model CNN* Pengujian 1

Hasil pelatihan model pertama yang diukur menggunakan *metrics MSE*, *RMSE*, dan *MAE* adalah sebagai berikut:

Batch Size Train	Batch Size Valid	Epoch	Train MSE	Train MAE	Train RMSE	Val MSE	Val MAE	Val RMSE
100	100	10	0.1874	0.3446	0.4315	0.1779	0.3369	0.4207

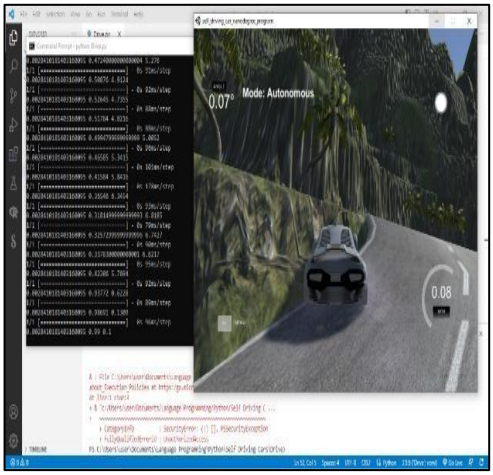
Gambar 7 *Metrics* MSE, RMSE, MAE Model 1

Berikut adalah visualisasi data *metrics loss* dari tabel 1:



Gambar 8 Visualisasi *Metrics Loss Model* Pengujian 1

Setelah dilatih, model pembelajaran mesin disimpan dengan *file* berformat .h5 dan kemudian dihubungkan dengan aplikasi simulator *Udacity*. Pada pengujian pertama didapatkan hasil yang belum maksimal, dimana mobil belum mampu untuk melewati tikungan pertama yang ada di lintasan data *metrics loss* dari gambar 11:



Gambar 9 Hasil Pengujian Pertama

2. Pengujian Dua

Setelah gagal di pengujian pertama, dilakukan proses *hyperparameter tuning* untukmelatih model CNN kedua. Model tersebut dilatih menggunakan data pelatihan yang dikelompokan dalam ukuran *batch* 100. Jumlah*batch* yang akan diproses dalam satu *epochs* sebesar 300. Untuk pengujian kedua, model dilatih dengan menggunakan 15 *epochs*. Jadi pada pengujian kedua, model dilatih menggunakan 30.000 data pelatihan.

Summary dari arsitektur model CNN kedua adalah sebagai berikut:

```
1 model = cnn_model()
2 print(model.summary())

Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 31, 98, 24)        1824

conv2d_1 (Conv2D)            (None, 14, 47, 36)        21636

conv2d_2 (Conv2D)            (None, 5, 22, 48)         43248

conv2d_3 (Conv2D)            (None, 3, 20, 64)         27712

flatten (Flatten)            (None, 3840)              0

dense (Dense)                (None, 100)               384100

dense_1 (Dense)              (None, 50)                5050

dense_2 (Dense)              (None, 10)                510

dense_3 (Dense)              (None, 1)                 11

=================================================================
Total params: 484,091
Trainable params: 484,091
Non-trainable params: 0
_________________________________________________________________
None
```

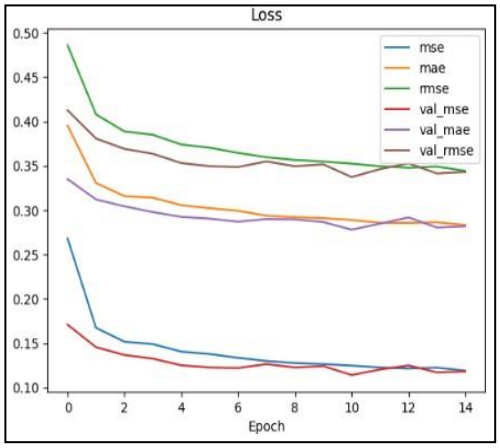
Gambar 10 *Summary Model CNN* Pengujian 2

Hasil pelatihan model kedua yang diukur menggunakan *metrics MSE*, *RMSE*, dan *MAE* adalah sebagai berikut:

Batch Size Train	Batch Size Valid	Epoch	Train MSE	Train MAE	Train RMSE	Val MSE	Val MAE	Val RMSE
100	100	15	0.1190	0.2833	0.3442	0.1182	0.2818	0.3431

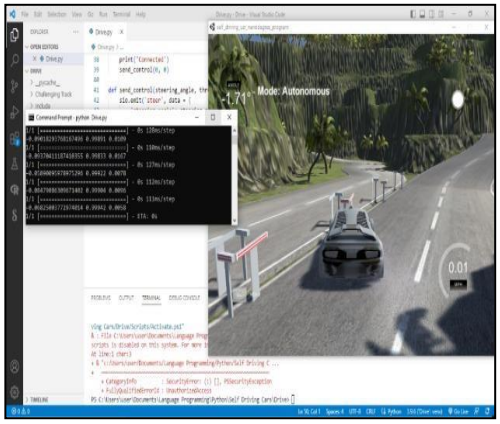
Gambar 11 *Metrics* MSE, RMSE, MAE Model 2

Berikut adalah visualisasi data *metrics loss* dari gambar 11:



Gambar 12 Visualisasi *Metrics Loss Model* Pengujian 2

Pada pengujian kedua didapatkan hasil yang juga belum maksimal, dimana mobil langsung menabrak pembatas lintasan pada saat dijalankan.



Gambar 73 Hasil Pengujian Kedua

3. Pengujian Tiga

Setelah gagal di pengujian pertama dan kedua, dilakukan proses *hyperparameter tuning* dan penambahan jumlah data pelatihan untuk model CNN pengujian ketiga. Model tersebut akan dilatih menggunakan data pelatihan yang lebih besar dari dua pengujian sebelumnya. Data yang digunakan akan dikelompokan dengan ukuran *batch* 256. Jumlah *batch* yang akan diproses dalam satu *epochs* sebesar 250 Untuk pengujian ketiga, model dilatih dengan menggunakan 15 *epochs*. Jadi pada pengujian ketiga, model dilatih menggunakan 64.000 data pelatihan.

Summary dari arsitektur model CNN ketiga adalah sebagai berikut:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 31, 98, 24)        1824

 conv2d_1 (Conv2D)           (None, 14, 47, 36)        21636

 conv2d_2 (Conv2D)           (None, 5, 22, 48)         43248

 conv2d_3 (Conv2D)           (None, 3, 20, 64)         27712

 conv2d_4 (Conv2D)           (None, 1, 18, 64)         36928

 flatten (Flatten)           (None, 1152)              0

 dense (Dense)               (None, 100)               115300

 dense_1 (Dense)             (None, 50)                5050

 dense_2 (Dense)             (None, 10)                510

 dense_3 (Dense)             (None, 1)                 11

=================================================================
Total params: 252,219
Trainable params: 252,219
Non-trainable params: 0
_________________________________________________________________
None
```

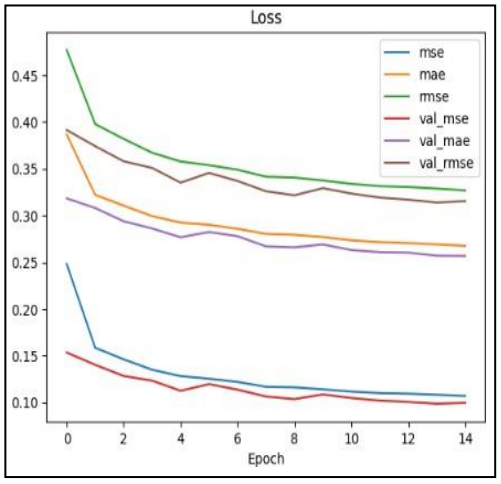
Gambar 14 *Summary Model* CNN Pengujian 3

Hasil pelatihan model ketiga yang diukur menggunakan *metrics MSE*, *RMSE*, dan *MAE* adalah sebagai berikut:

Batch Size Train	*Batch Size Valid*	*Epoch*	*Train MSE*	*Train MAE*	*Train RMSE*	*Val MSE*	*Val MAE*	*Val RMSE*
256	*256*	*15*	*0.1070*	*0.2675*	*0.3268*	*0.0996*	*0.2567*	*0.3153*

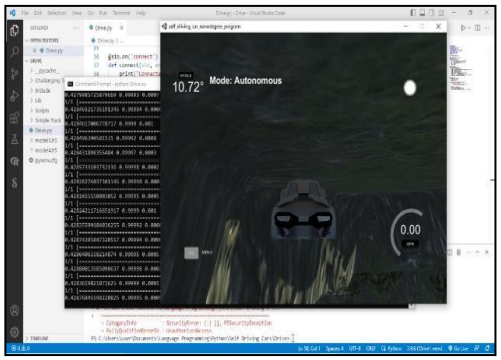
Gambar 15 *Metrics* MSE, RMSE, MAE Model 2

Berikut adalah visualisasi data *metrics loss* dari gambar 15:



Gambar 86 Visualisasi *Metrics Loss Model* Pengujian 3

Pada pengujian ketiga didapatkan hasil yanglebih baik, meskipun mobil terbalik saat belum berhasil melewati tikungan besar, namun pada pengujian ketiga mobil sudah berjalan cukup jauh dari titik awal mobil tersebut bergerak.



Gambar 97 Hasil Pengujian Kedua

Setelah dilakukan perobaan sebanyak tiga kali, didapatkan *summary* dari masing-masing *metrics loss* dari setiap percobaan.

Pengujian	*loss mse*	*loss mae*	*loss rmse*	*val mse*	*val mae*	*val rmse*
Pengujian Pertama	*0.1874*	*0.3446*	*0.4315*	*0.1779*	*0.3369*	*0.4207*
Pengujian Kedua	*0.1190*	*0.2833*	*0.3442*	*0.1182*	*0.2818*	*0.3431*
Pengujian Ketiga	*0.1070*	*0.2675*	*0.3268*	*0.0996*	*0.2567*	*0.3153*

Gambar 18 *Summary Metrics* dari Ketiga Pengujian

B. Incremental II

Pada *Incremental II* dihasilkan sebuah *script* yang berfungsi untuk menghubungkan model pembelajaran mesin dengan aplikasi simulator *Udacity*. Pada tahap ini dilakukan juga pengujian *whitebox testing*

No	*Pengujian*	*Hasil Yang diharapkan*	*Hasil Yang diharapkan*	*Status*
1	*Menjalankan Script "Drive.py"*	*Script "Drive.py" tidak menampilkan error message.*	*Tidak menghasilkan error message apapun pada saat dijalankan*	*Berhasil*
2	*Script "Drive.py" dihubungkan dengan aplikasi simulator Udacity*	*Script "Drive.py" memberikan feedback berupa string "Connected" pada saat dihubungkan ke simulator*	*Pada saat dijalankan, script memberikan feedback string "Connected" yang menandakan script berhasil terhubung ke simulator*	*Berhasil*
3	*Script "Drive.py" memberikan data telemetri*	*Script "Drive.py" mampu mengirimkan data telemetri seperti steering_angle, throttle, dan speed ke command prompt saat simulator dijalankan dalam mode autonomous*	*Pada saat dijalankan, script mampu mengirimkan data telemetri ke command prompt saat simulator dijalankan dalam mode autonomous*	*Berhasil*
4	*Script "Drive.py" terhubung dengan simulator Udacity dalam mode autonomous*	*Aplikasi simulator Udacity dapat berjalan dalam mode autonomous dan membaca model pembelajaran mesin yang dihubungkan melalui Script "Drive.py"*	*Aplikasi simulator Udacity dapat berjalan dalam mode autonomous dan mampu membaca model pembelajaran mesin yang dihubungkan melalui Script "Drive.py"*	*Berhasil*

Gambar 19 Skenario Pengujian *Whitebox Testing*

IV. SIMPULAN

Berdasarkan hasil analisis dan uji coba yangdilakukan pada penelitian ini, maka dapat diambil kesimpulan sebagai berikut:

1. Metode *convolutional neural network* dapat digunakan untuk merancang sistem kendaraan otonom. Selain itu, NVIDIA CNN *Architecture* dapat digunakan untuk merancang sistem kendali mobil otonom.
2. Model pembelajaran mesin yang dihasilkan sudah berhasil untuk menjalankan mode *autonomous* pada aplikasi simulator *Udacity*. Selain itu, *script Drive.py* juga berhasil untuk menghubungkan model pembelajaran mesin tersebut dengan aplikasi simulator *Udacity* melalui arsitektur *client-server*.
3. Berdasarkan enam tingkatan mobil otonom, perancangan ini menghasilkan

mobil otonom dengan level 3 (*ConditionalAssistance*), dimana pada level ini mobil dapat mengontrol setir, gas, dan rem dan mobil dapat memantau objek atau lingkungan yang ada di sekitarnya melalui model pembelajaran mesin yang telah dirancang.

4. Saat melakukan proses pelatihan, kualitas dan jumlah *dataset* yang digunakan mempengaruhi model pembelajaran mesin yang dihasilkan. Selain itu, parameter seperti *epochs* juga mempengaruhi model yang dihasilkan.

DAFTAR RUJUKAN

[1] J. Ondruš, E. Kolla, P. Vertaľ, and Ž. Šarić, "How Do Autonomous Cars Work?," in *Transportation Research Procedia*, Elsevier B.V., 2020, pp. 226–233. doi: 10.1016/j.trpro.2020.02.049.

[2] A. Faisal, T. Yigitcanlar, M. Kamruzzaman, and G. Currie, "Understanding autonomous vehicles:A systematic literature review on capability, impact, planning and policy," *J Transp Land Use*, vol. 12, no.1, pp. 45–72, 2019, doi: 10.5198/jtlu.2019.1405.

[3] S. Ali, "Self-Driving Cars: Automation Testing Using *Udacity* Simulator," *International Research Journal of Engineering and Technology*, 2021, [Online]. Available: www.irjet.net

[4] M. Bojarski *et al.*, "End to End Learning for Self-Driving Cars," Apr. 2016, doi: https://doi.org/10.48550/arXiv.1604.07316.

[5] Ziyad Mohammed, *Artificial Intelligence Definition, Ethics and Standards*. 2019. Accessed: Dec. 11, 2022. [Online]. Available: https://www.researchgate.net/publication/332548325_Artificial_Intelligence_Definition_Ethics_and_Standards

[6] A. Kumar, B. A. Shivani, A. Nath, A. Singh, and N. S. Tengli, "Self-Driving Cars Using A Simulator," *International Journal of Advanced Research in Computer Science*, vol. 11, 2020, [Online]. Available: www.ijarcs.info

[7] Q. Bi, K. E. Goodman, J. Kaminsky, and J. Lessler, "What is machine learning? A primer for the epidemiologist," *Am J Epidemiol*, vol. 188, no. 12, pp. 2222–2239, Dec. 2019, doi: 10.1093/aje/kwz189.

[8] R. Khallaf and M. Khallaf, "Classification and analysis of deep learning applications in construction: A systematic literature review," *Autom Constr*, vol. 129, Sep. 2021, doi: 10.1016/j.autcon.2021.103760.

[9] W. Saputra and D. Prabowo, "Pengembangan Aplikasi Klasifikasi Gambar Menggunakan Library Tensorflow yang Menerapkan Algoritma Convolutional Neural Network Studi Kasus: Galeri Foto Kegiatan Ibadah Gereja Shoot Fellowship," 2022. doi: Volume 8, No. 3, September 2022.

[10] C. Oktaviany and T. L. Marselino, "Pengembangan Perangkat Lunak untuk Pengelompokkan Tumbuhan Berdasarkan Citra Digital Daun Menggunakan CNN," 2022. doi: Volume 8, No.2, Mei, 2022.