JURNAL TEKNIK POMITS Vol. 1, No. 1, (2012) 1-6

RANCANG BANGUN MODEL DEEP LEARNING PADA SISTEM NAVIGASI MOBIL OTONOM DENGAN SENSOR KAMERA

I Made Guna Winangun, Achmad Affandi, Endroyono
Teknik Eletkro, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember (ITS)
Jl. Arief Rahman Hakim, Surabaya 60111
E-mail: rumahgugun@gmail.com, affandi@ee.its.ac.id, endroyono@ee.its.ac.id

***Abstrak*— Fenomena yang melatar belakangi penelitian ini yaitu susahnya computer vision dalam menavigasi mobil pada situasi yang belum pernah dilalui oleh mobil sebelumnya. Untuk mengatasi fenomena tersebut diperlukan pendekatan baru dalam menavigasi mobil. Pendekatan tersebut yaitu penggunaan deep learning dalam menavigasi mobil. Pada penelitian ini didapatkan suatu rancangan model deep learning yang dapat menavigasi mobil. Model menggunakan salah satu arsitektur deep learning yaitu convolutional neural network. Model ini dilatih berdasarkan data yang didapat dari simulator. Data yang diambil adalah data berupa arah kemudi dengan gambar yang berasal dari kamera yang terpasang pada mobil di simulator. Dengan adanya model ini, mobil dapat dikatakan sebagai mobil otonom karena dapat menavigasi dirinya sendiri. Model ini memiliki validasi galat sebesar 0.0827 derajat. Model ini dapat menavigasi mobil dengan baik pada kecepatan 15 – 20 satuan kecepatan pada simulator.**

***Kata Kunci*— Deep Learning, Mobil Otonom, Navigasi, Sensor Kamera**

I. PENDAHULUAN

Perkembangan teknologi transportasi telah mencapai tahap yang maju, sejalan dengan perkembangan teknologi 4.0. Pada tahun 2020, ITS memiliki program pengembangan kendaraan autonomous yang akan dijadikan sebagai penelitian flaghship ITS yang dinamakan project ICAR[1]. ICAR dapat menjadi state of the art technology multi disiplin di ITS dan pengembangan tingkat lanjut teknologi kendaraan listrik cerdas. Multi disiplin yang dimaksud terdiri dari berbagai akademisi di departemen yang ada di Institut Teknologi Sepuluh Nopember. Pada tanggal 17 Agustus 2020, Institut Teknologi Sepuluh Nopember melakukan soft launching terhadap ICAR.[2]

ICAR memiliki beberapa proses pengembangan. Proses pengembangan tersebut dibagi menjadi lima pengembangan. Pengembangan – pengembangan tersebut yaitu: pengembangan platform electric vehicle, pengembangan mekatronik, pengembangan sistem navigasi, pengembangan sistem komunikasi, dan pengembangan sistem UI/UX dan command control. Pada pengembangan sistem navigasi, ICAR menggunakan berbagai jenis sensor untuk menavigasi kendaraannya. Sensor – sensor tersebut berupa lidar dan kamera.

Kamera merupakan sensor yang digunakan untuk menangkap gambar. Kamera dapat memproses gambar menjadi berbagai perintah kerja pada suatu sistem dengan bantuan kecerdasan tambahan.[3] Kecerdasan itu yaitu Deep learning[4]. Deep Learning bekerja dengan cara mempelajari data yang sebelumnya sudah ada lalu memprediksi variabel yang ingin diketahui. Deep Learning pada kendaraan autonomous digunakan agar kendaraan tersebut dapat berjalan pada kondisi jalanan yang berbeda – beda.[5] Terdapat berbagai arsitektur Deep Learning. Convolutional Neural Network merupakan salah satu arsitektur yang pada umumnya digunakan untuk pengolahan gambar. NVIDIA telah merancang suatu arsitektur CNN dengan nama PilotNet[6] yang dikatakan dapat menavigasi mobil secara andal.

Ruang lingkup permasalahan yang akan diselesaikan dalam tugas akhir ini adalah merancangan model yang dapat menavigasi mobil otonom. Tujuan dari tugas akhir ini adalah mendapatkan rancangan model deep learning yang dapat menavigasi mobil serta kinerjanya dengan menguji kemampuan model dalam menavigasi mobil pada simulator mobil oleh udacity.

II. DASAR TEORI

A. Mobil Otonom

Mobil Otonom adalah kendaraan yang mampu mencermati lingkungan sekitarnya dan bergerak dengan aman dengan sedikit atau tanpa kendali dari manusia. Kendaraan ini menggunakan salah satu atau menggabungkan berbagai sensor untuk melihat sekelilingnya, seperti kamera, radar, lidar, sonar, GPS, odometri, dan unit pengukuran inersia.[7] Sensor – sensor tersebut dapat dirancang menjadi sistem navigasi pada kendaraan otonom. Sistem ini dapat menafsirkan informasi sensorik untuk mengidentifikasi jalur navigasi yang sesuai, serta rintangan dan marka jalan.

B. Sistem Navigasi

Sistem navigasi adalah sistem yang memiliki peran dalam menavigasi suatu objek. Dalam menavigasi suatu objek,

dapat digunakan beberapa metode. Salah satunya dengan metode navigasi secara visual.[8] Dalam navigasi mobil otonom, metode ini menggunakan input berupa gambar yang diambil dari kamera pada mobil. Sedangkan outputnya dapat berupa arah kemudi mobil maupun kecepatan mobil. Hal inilah yang menyebabkan mobil dapat dinavigasi dengan metode navigasi secara visual.

C. Sensor Kamera

Kamera merupakan perangkat untuk menangkap gambar. Hasil tangkapan gambar ini akan disimpan dalam bentuk gambar digital di suatu memori digital. Gambar digital ini memiliki elemen citra yang disebut pixel. Pixel adalah elemen citra yang memiliki nilai yang menunjukkan intensitas warna.[9] Perangkat ini dapat menjadi sensor pada suatu sistem elektronik.[10] Sensor kamera dapat menggantikan beberapa sensor sekaligus untuk mengurangi kompleksitas sistem.

D. Computer Vision

Computer Vision membahas bagaimana komputer dapat dirancang untuk mendapatkan pemahaman tingkat tinggi dari gambar atau video digital.[11] Pemahaman tingkat tinggi yang dimaksud diantaranya yaitu mengenali objek, mencari objek, merubah gambar, dan lainnya. Tugas Computer Vision meliputi metode untuk memeroleh, memroses, menganalisis, dan memahami gambar digital dan ekstrasi data berdimensi tinggi dari dunia nyata untuk menghasilkan informasi yang diperlukan, bisa berupa numerik maupun simbolik.[12] Dalam penggunaannya, computer vision berusaha untuk mengotomatiskan tugas yang dapat dilakukan oleh manusia secara visual.[13]

E. Machine Learning

Machine learning[14] merupakan salah satu cabang dalam sistem kecerdasan buatan yang memungkinkan komputer dapat belajar dari data dan pengalamannya. Machine learning digunakan pada saat permasalahan yang ingin diselesaikan terlalu kompleks untuk diprogram secara analitis. Sebagai ganti dari pemecahan masalah secara analitis, machine learning perlu data dalam jumlah besar ke algoritma machine learning dan membiarkan algoritma machine learning tersebut mencari karakteristik model yang diinginkan oleh pembuat program proses ini dikenal sebagai model training.

F. Deep Learning

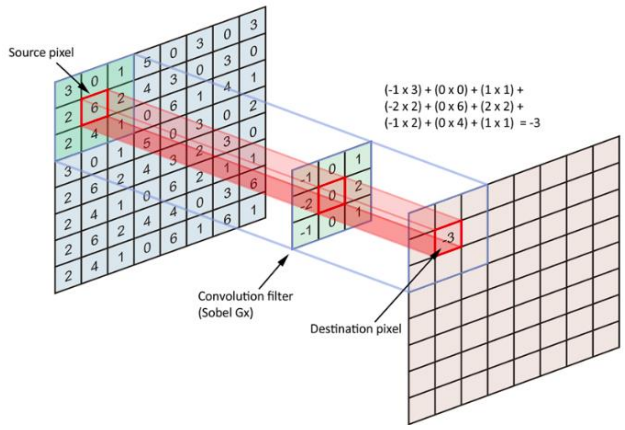
Deep Learning adalah bagian dari Machine Learning yang menggunakan lapisan - lapisan untuk mengekstrak fitur tingkat yang lebih tinggi secara progresif dari input. Fitur ini yaitu untuk melakukan tugas yang mirip manusia, seperti mengenali ucapan, mengidentifikasi gambar, atau membuat prediksi. Alih-alih mengatur data untuk dijalankan melalui persamaan yang telah ditentukan sebelumnya, deep learning menetapkan parameter dasar tentang data dan melatih komputer untuk belajar sendiri dengan mengenali pola menggunakan banyak lapisan pemrosesan. Deep Learning adalah area baru dalam penelitian Machine Learning, yang diperkenalkan dengan tujuan untuk membuat Machine Learning lebih dekat ke salah satu tujuan awalnya: Kecerdasan Buatan.

G. Jaringan Syaraf Tiruan

Sistem jaringan syaraf tiruan[15] merupakan salah satu pengembangan algoritma yang diterapkan pada mesin atau komputer untuk mengerjakan suatu tugas dengan cara kerja selayaknya otak manusia. Dalam sistem otak manusia mampu untuk membangun, mengambil kesimpulan, dan belajar dari pengalaman yang telah diterima. Sistem jaringan syaraf utama pada manusia terbagi atas tiga tahap yaitu reseptor, jaringan syaraf, dan efektor.

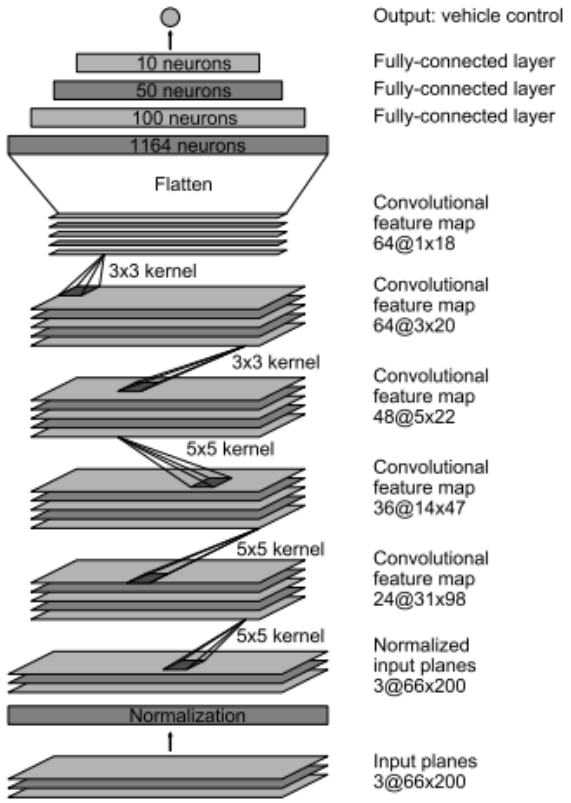
Pada dasarnya neuron pada neural network saling terhubung satu dengan lainnya agar dapat digunakan untuk melakukan proses pembelajaran. Secara umum arsitektur neural network dibagi menjadi beberapa bagian yakni: Perceptron, Multi-Layer Feedfoward Neural Network, Recurrent Neural Network, Symmetrically Connected Networks, dan Convolutional Neural Network.

Convolutional Neural Networks pada umumnya diaplikasikan pada analisa gambar dan video. Terdapat kata Convolutional pada CNN mengindikasikan bahwa jaringan tersebut menggunakan operasi matematika yang disebut konvolusi. Jaringan konvolusional adalah jenis jaringan saraf khusus yang menggunakan konvolusi sebagai pengganti perkalian matriks umum pada salah satu layer atau lebih.



Gambar 1 Konvolusi pada CNN[16]

Salah satu contoh arsitektur CNN adalah arsitektur yang dirancang oleh NVIDIA yaitu PilotNet. Arsitektur ini dikatakan mampu untuk meminimalisir kompleksitas sistem.[5] Total jaringan ini terdiri dari sembilan layer yaitu lapisan normalisasi, lima lapisan konvolusional, dan tiga lapisan terkoneksi penuh. Lapisan konvolusional digunakan untuk mengekstrak fitur dari gambar. Lapisan terkoneksi penuh untuk menentukan nilai derajat kemudi berdasarkan fitur yang didapatkan dari hasil konvolusi. Dilakukan pre-processing terlebih dahulu pada input gambar sebelum masuk ke arsitektur ini. Arsitektur ini akan menggunakan gambar sebagai input dan derajat kemudi sebagai output.



Gambar 2 Arsitektur PilotNet[5]

H. *Teknik Augmentasi*

Augmentasi data adalah teknik meningkatkan ukuran data yang digunakan untuk melatih model. Hal ini dilakukan agar data yang dikoleksi tidak perlu terlalu banyak. Untuk prediksi yang andal, model deep learning sering kali memerlukan banyak data pelatihan, yang tidak selalu tersedia. Oleh karena itu, data yang ada ditambah untuk membuat model yang dapat membaca situasi umum yang lebih baik.[17] Apabila data yang dimiliki terlalu banyak (data kurang memiliki variasi yang banyak) maka berakibat kepada efek overfitting. Sedangkan apabila data yang dimiliki terlalu sedikit maka berakibat kepada efek underfitting.

I. *Simulator Mobil Otonom*

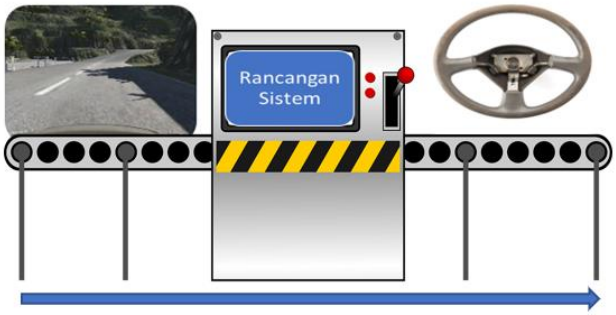
Terdapat berbagai simulator untuk menyimulasikan mobil otonom. Simulator tersebut memiliki spesifikasi yang dibutuhkan untuk dijalankan yang berbeda – beda. Selain spesifikasi yang berbeda – beda, terdapat pula perberdaan tingkat kompleksitas data yang dapat diambil tiap simulator. Simulator – simulator tersebut dapat dilihat pada table dibawah.

Tabel 1 Simulator - simulator mobil otonom[18]

Name	Graphic requirement	Data collection	OS requirements (Mainly support)	Autonomous driving mode
Udacity	low	easy	Window, Mac OS, Linux	Yes
TORCS	medium	complex	Linux	Yes
Euro Truck Simulator 2	high	complex	Window, Mac OS, Linux	Yes
Gazebo	medium	complex	Linux	Yes
CARLA	high	complex	Window, Linux	Yes

III. PERANCANGAN MODEL

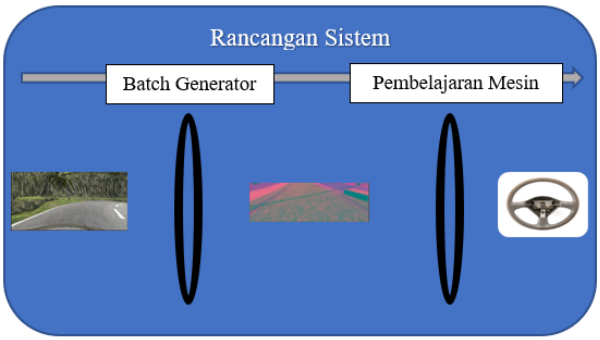
A. *Rancangan Sistem Secara Umum*



Gambar 3 Gambaran rancangan sistem secara umum

Pada rancangan ini akan dirancang sebuah sistem yang dapat mengolah data berupa gambar sehingga menghasilkan output berupa arah kemudi mobil. Dalam pembentukan rancangan ini terdiri dari serangkaian tahapan pengolahan data. Tahapan – tahapan tersebut yaitu:

1. Pengolahan gambar sebelum diproses (pre-processing) dengan 5 langkahw:
   1) Membuang background pada data gambar
   2) Merubah format gambar RGB menjadi YUV
   3) Melakukan GaussianBlur dengan kernel 3x3
   4) Mengubah dimensi gambar menjadi 66x200 pixels
   5) Melakukan normalisasi pada nilai tiap pixel
2. Proses penciptaan data gambar yang sudah diaugmentasi dan pre-processing dengan batch generator
3. Proses pembelajaran Machine Learning untuk mendapatkan model machine learning yang dapat menghasilkan arah kemudi dengan input gambar.



Gambar 4 Tahapan dalam pembentukan rancangan model

B. *Pengumpulan Data*

Pengumpulan data dilakukan dengan cara mengemudikan mobil pada simulator sehingga mendapatkan data berupa gambar beserta arah kemudi mobil. Pengambilan data dilakukan dengan mengemudikan mobil pada trek satu sebanyak enam putaran dimana tiga putaran pertama berbeda dengan tiga putaran selanjutnya dan pada trek dua sebanyak empat putaran dimana dua putaran berbeda dengan putaran selanjutnya. Perbedaan tersebut yaitu arah jalur kemudi mobil. Hal ini dilakukan agar terjadi keseimbangan dan tidak terjadi bias.

Setelah pengambilan data dilakukan, didapatkan total data sejumlah 14044. Dimana tiap data terdiri dari gambar kamera

tengah, gambar kamera kiri, gambar kamera kanan, arah kemudi, throttle, reverse, dan kecepatan. Pada perancangan ini akan diambil data gambar kamera tengah, kiri, dan kanan, serta arah kemudi.

Data yang digunakan untuk pelatihan jaringan syaraf yaitu gambar kamera tengah, kiri, dan kanan, serta arah kemudi sebagai label saat pelatihan. Sedangkan data yang digunakan untuk testing menggunakan gambar kamera tengah serta arah kemudinya.

C. Augmentasi

Untuk mendapatkan variasi data pelatihan lebih banyak dari sumber yang terbatas, maka pada proses yang akan dilakukan yaitu augmentasi data. Fungsi augmentasi gambar berfungsi sebagai fungsi yang menciptakan gambar baru dengan modifikasi terhadap gambar asli. Modifikasi yang dilakukan terhadap gambar asli dilakukan secara acak. Gambar asli yang akan dimodifikasi memiliki probabilitas untuk melewati fungsi – fungsi tersebut sebesar 50%. Hal ini ditentukan agar keacakan dari hasil gambar augmentasi terjadi secara merata. Modifikasi tersebut merupakan kombinasi acak dari merubah skala gambar, menggeser gambar, mencerminkan gambar, menambahkan bayangan, dan merubah kecerahan gambar.

D. Pre-processing

Pada tahapan ini akan dilakukan proses yang dinamakan pre-processing. Hal ini akan melakukan perubahan terhadap data yang telah ditelah diaugmentasi maupun data validasi. Perubahan – perubahan yang terjadi yaitu perubahan format gambar, ukuran dimensi pixel gambar, aplikasi GaussianBlur untuk mengurangi noise pada gambar, dan normalisasi nilai tiap pixel pada gambar. Hal ini dilakukan karena arsitektur model kecerdasan yang dirancang oleh NVIDIA menyarankan format gambar yang akan dilatih merupakan format YUV bukan RGB dan dimensi gambar berukuran 200x66 pixels.

E. Arsitektur Jaringan

Rancangan arsitektur yang digunakan untuk perancangan model yang akan dibangun menggunakan arsitektur CNN dari NVIDIA yaitu PilotNet. Arsitektur ini dapat dilihat pada Gambar 2. Arsitektur akan dilatih untuk meminimalisir nilai *mean squared error* diantara nilai kemudi yang dihasilkan oleh jaringan dan nilai kemudi yang berasal dari data yang diambil. Input dari jaringan ini berupa gambar yang telah dikonversi dari format RGB ke YUV dan nilai tiap pixelnya telah dinormalisasi.

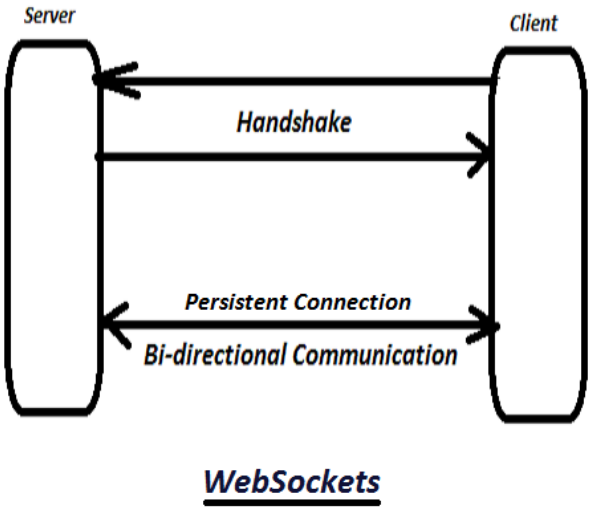
Arsitektur ini terdiri dari 5 lapisan konvolusi dan 3 lapisan jaringan terhubung penuh. 5 lapisan konvolusi berguna untuk mendapatkan fitur dari gambar. Pada 3 lapisan pertama digunakan lapisan konvolusi yang menggunakan 5x5 kernel dengan stride 2x2. Untuk 2 lapisan selanjutnya menggunakan 3x3 kernel dengan stride 1x1. 3 lapisan setelah lapisan konvolusi yaitu lapisan jaringan terhubung penuh. Lapisan ini berfungsi dalam mendapatkan nilai kemudi.

F. Pembelajaran Jaringan

Setelah arsitektur jaringan dan batch generator siap, maka akan dilakukan proses pembelajaran model. Pada proses ini batch generator akan memberikan gambar – gambar untuk dilatih oleh model. Tiap epochnya model akan mengevaluasi bobot tiap node. Pada proses pelatihan ini ditambah pula fungsi callback untuk mempercepat proses pelatihan apabila nilai dari val_loss tidak kunjung naik dalam kurun lima kali epoch. Untuk optimizer yang digunakan yaitu Adam dengan learning rate 0.0001.

G. Deployment

Saat melakukan pemasangan model pada simulator, hal yang terjadi yaitu komunikasi dua arah antar simulator (sebagai client) dengan program computer (sebagai server) melalui web app dengan menggunakan tools Flask dan socket.io.



Gambar 5 Alur komunikasi antar simulator dan program

Saat server dan client sudah terhubung, maka server akan meminta gambar tangkapan dari client. Setelah diterima, gambar akan diproses dengan proses pre-processing. Lalu hasil dari proses itu akan menjadi input dari fungsi prediksi. Fungsi prediksi ini merupakan fungsi bawaan dari model yang sudah terlatih. Setelah dilakukan prediksi maka akan dihasilkan output berupa arah kemudi. Arah kemudi ini lalu dikirimkan ke client untuk menavigasi mobil.

Navigasi throttle mobil dilakukan karena mobil (client) menunggu input throttle dari server. Sehingga server mengirimkan throttle dengan nilai yang dapat dilihat pada persamaan 1. Hal ini dapat menjadikan mobil melakukan manuver sehingga kecepatan maksimal atau melakukan pengereman saat kecepatan mobil melebihi kecepatan maksimal.

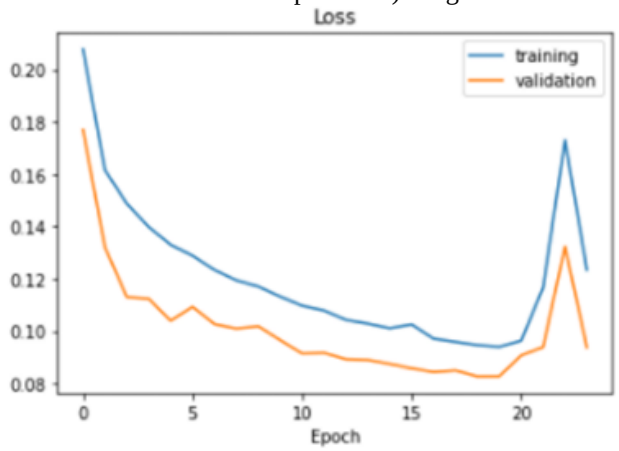
$$Throttle = 1 - \frac{kecepatan\ mobil}{kecepatan\ maximal} \quad \textbf{(1)}$$

IV. HASIL DAN ANALISA

Pada proses pelatihan, digunakan fungsi callback untuk mempercepat proses pelatihan. Fungsi callback tersebut akan menghentikan pelatihan saat nilai val_loss tidak mengecil dalam lima epoch dan mengembalikan nilai tersebut. Sehingga nilai akhir dari proses pembelajaran tersebut memiliki nilai val_loss sebesar 0.0827 derajat. Nilai ini merupakan nilai yang didapatkan dengan mencari nilai rata – rata dari nilai error yang dikuadratkan. Nilai error merupakan perbedaan nilai arah kemudi yang dihasilkan oleh model

dengan nilai arah kemudi dari data.

Setelah didapatkan nilai mse dari model tersebut, selanjutnya akan menampilkan grafik loss untuk mengetahui sifat dari model yang dirancang (underfit, overfit atau good fit). Model yang didapatkan dari rancangan ini ditunjukkan pada Gambar 6. Dari gambar tersebut tergambarkan bahwa model memiliki karakteristik good fit, artinya model dapat menavigasi mobil dalam lingkungan yang lebih general (bisa membaca situasi diluar data pelatihan) dengan baik.



Gambar 6 Plot Loss Pelatihan

Setelah dilakukan plotting terhadap grafik loss, dilakukan evaluasi model terhadap seluruh data berupa gambar yang relative terhadap arah kemudi pada track 1 dan track 2. Hal ini dilakukan dengan pemanggilan fungsi evaluate pada model deep learning tersebut dengan inputan gambar dan arah kemudi sebagai evaluasinya. Hal ini dapat dilihat pada Gambar 7. Nilai yan didapatkan yaitu 0.0545 derajat. Nilai tersebut merupakan nilai rata – rata dari nilai error yang dikuadratkan. Nilai error merupakan perbedaan nilai arah kemudi yang dihasilkan oleh model dengan nilai arah kemudi dari data.

```
[30] model.evaluate(batch_generator(image_paths, steerings, 1, 0), batch_size = 1, steps=42132)

     42132/42132 [==============================] - 173s 4ms/step - loss: 0.0545
     0.05447138100862503
```

Gambar 7 Evaluasi Model

```
Epoch 1/25
759/759 [==============================] - 509s 648ms/step - loss: 0.2696 - val_loss: 0.1770
Epoch 2/25
759/759 [==============================] - 490s 646ms/step - loss: 0.1649 - val_loss: 0.1321
Epoch 3/25
759/759 [==============================] - 489s 645ms/step - loss: 0.1517 - val_loss: 0.1130
Epoch 4/25
759/759 [==============================] - 488s 642ms/step - loss: 0.1421 - val_loss: 0.1124
Epoch 5/25
759/759 [==============================] - 487s 642ms/step - loss: 0.1342 - val_loss: 0.1039
Epoch 6/25
759/759 [==============================] - 485s 639ms/step - loss: 0.1310 - val_loss: 0.1093
Epoch 7/25
759/759 [==============================] - 487s 641ms/step - loss: 0.1249 - val_loss: 0.1027
Epoch 8/25
759/759 [==============================] - 488s 643ms/step - loss: 0.1208 - val_loss: 0.1010
Epoch 9/25
759/759 [==============================] - 488s 642ms/step - loss: 0.1166 - val_loss: 0.1019
Epoch 10/25
759/759 [==============================] - 488s 643ms/step - loss: 0.1135 - val_loss: 0.0965
Epoch 11/25
759/759 [==============================] - 487s 641ms/step - loss: 0.1111 - val_loss: 0.0914
Epoch 12/25
759/759 [==============================] - 487s 641ms/step - loss: 0.1088 - val_loss: 0.0917
Epoch 13/25
759/759 [==============================] - 488s 643ms/step - loss: 0.1054 - val_loss: 0.0892
Epoch 14/25
759/759 [==============================] - 488s 643ms/step - loss: 0.1023 - val_loss: 0.0890
Epoch 15/25
759/759 [==============================] - 488s 643ms/step - loss: 0.1014 - val_loss: 0.0874
Epoch 16/25
759/759 [==============================] - 487s 641ms/step - loss: 0.1021 - val_loss: 0.0857
Epoch 17/25
759/759 [==============================] - 487s 642ms/step - loss: 0.0976 - val_loss: 0.0844
Epoch 18/25
759/759 [==============================] - 488s 642ms/step - loss: 0.0960 - val_loss: 0.0850
Epoch 19/25
759/759 [==============================] - 487s 641ms/step - loss: 0.0957 - val_loss: 0.0827
Epoch 20/25
759/759 [==============================] - 488s 643ms/step - loss: 0.0940 - val_loss: 0.0827
Epoch 21/25
759/759 [==============================] - 487s 642ms/step - loss: 0.0933 - val_loss: 0.0907
Epoch 22/25
759/759 [==============================] - 488s 643ms/step - loss: 0.1055 - val_loss: 0.0939
Epoch 23/25
759/759 [==============================] - 489s 644ms/step - loss: 0.1449 - val_loss: 0.1323
Epoch 24/25
759/759 [==============================] - 487s 642ms/step - loss: 0.1369 - val_loss: 0.0937
```

Gambar 8 Proses Pembelajaran Model

Pada evaluasi model dalam menavigasi mobil, terjabarkan pada **Tabel 2**. Analisa terhadap pengujian yang telah dilakukan yakni mobil dapat melaju dengan baik pada track 1 dengan kecepatan apapun karena track 1 memiliki karakteristik jalan pada track yang cenderung datar sedangkan pada track dua memiliki karakteristik jalanan yang ada tanjakan dan turunan sehingga ini mengakibatkan mobil menjadi terbang sehingga menabrak dan tersangkut. Sedangkan pada kecepatan 25 pada track 3 tidak berhasil karena karakteristik pada jalanan track 3 bergelombang mengakibatkan mobil sempat terbang lalu menabrak tiang.

Tabel 2 Pengujian Navigasi Mobil

Kecepatan	Track 1	Track 2	Track 3	Keterangan
10	3 putaran berhasil	Tidak berhasil	Finish	Mobil tidak dapat bermanuver saat tanjakan pada track 2
15	3 putaran berhasil	3 putaran berhasil	Finish	
20	3 Putaran berhasil	3 Putaran berhasil	Finish	
25	3 Putaran berhasil	Tidak berhasil	Tidak berhasil	Menabrak dan tersangkut
30	3 Putaran berhasil	Tidak berhasil	Finish	Menabrak dan tersangkut

V. KESIMPULAN

Dari hasil simulasi dan analisa dapat disimpulkan beberapa kesimpulan yang bisa didapat pada Tugas Akhir ini. Diantaranya:

1. Rancangan ini dapat menavigasi mobil dengan baik pada 15 – 20 satuan kecepatan simulator
2. Data yang diambil dari track 1 dan track 2 dapat menjadi bahan untuk menavigasi mobil pada track 3
3. Rancangan ini memiliki arsitektur berupa PilotNet dengan nilai val_loss sebesar 0.0827 derajat

LAMPIRAN

Segala dokumentasi (code, rekaman uji coba, dan lainnya) akan dapat diakses melalui link berikut: https://github.com/imadegunawinangun/Autonomous-Car-CNN-Model

UCAPAN TERIMA KASIH

Penulis mengucapkan terimakasih terhadap Departemen Teknik Elektro dan civitas akademi ITS karena senantiasa memberikan ilmu dan pengalaman selama penulis mengampu perkuliahan di ITS.

DAFTAR PUSTAKA

[1] "CENIM 2020 Breaker Page," dalam *2020 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM)*, Surabaya, Nov 2020, hlm. i–v. doi: 10.1109/CENIM51130.2020.9297977.

[2] "Menristek/Kepala BRIN Luncurkan iCar ITS," *ristekdikti*. https://www.brin.go.id/menristek-kepala-

brin-luncurkan-icar-its/ (diakses Jul 22, 2021).

[3] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, dan D. Scaramuzza, "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM," *Int. J. Robot. Res.*, vol. 36, no. 2, hlm. 142–149, Feb 2017, doi: 10.1177/0278364917691115.

[4] I. Goodfellow, Y. Bengio, dan A. Courville, *Deep learning*. MIT press, 2016.

[5] M. Bojarski *dkk.*, "End to End Learning for Self-Driving Cars," *ArXiv160407316 Cs*, Apr 2016, Diakses: Jul 22, 2021. [Daring]. Tersedia pada: http://arxiv.org/abs/1604.07316

[6] M. Bojarski *dkk.*, "The NVIDIA PilotNet Experiments," *ArXiv201008776 Cs*, Okt 2020, Diakses: Jul 22, 2021. [Daring]. Tersedia pada: http://arxiv.org/abs/2010.08776

[7] D. Göhring, M. Wang, M. Schnürmacher, dan T. Ganjineh, "Radar/lidar sensor fusion for car-following on highways," dalam *The 5th International Conference on Automation, Robotics and Applications*, 2011, hlm. 407–412.

[8] F. Bonin-Font, A. Ortiz, dan G. Oliver, "Visual navigation for mobile robots: A survey," *J. Intell. Robot. Syst.*, vol. 53, no. 3, hlm. 263–296, 2008.

[9] J. F. Blinn, "What is a pixel?," *IEEE Comput. Graph. Appl.*, vol. 25, no. 5, hlm. 82–87, 2005.

[10] R. Khatry dan A. Thompson, "Camera and Lidar sensor models for autonomous vehicles," 2021.

[11] D. Forsyth dan J. Ponce, *Computer vision: A modern approach*. Prentice hall, 2011.

[12] G. H. Granlund dan H. Knutsson, *Signal processing for computer vision*. Springer Science & Business Media, 2013.

[13] H. Tian, T. Wang, Y. Liu, X. Qiao, dan Y. Li, "Computer vision technology in agricultural automation—A review," *Inf. Process. Agric.*, vol. 7, no. 1, hlm. 1–19, 2020.

[14] T. M. Mitchell, "Machine learning," 1997.

[15] B. Yegnanarayana, *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.

[16] "Understanding Deep Self-attention Mechanism in Convolution Neural Networks | by Shuchen Du | AI Salon | Medium." https://medium.com/ai-salon/understanding-deep-self-attention-mechanism-in-convolution-neural-networks-e8f9c01cb251 (diakses Jul 23, 2021).

[17] N. A. Batubara dan R. M. Awangga, *TUTORIAL OBJECT DETECTION PLATE NUMBER WITH CONVOLUTION NEURAL NETWORK (CNN)*, vol. 1. Kreatif, 2020.

[18] Y. Wang, D. Liu, H. Jeon, Z. Chu, dan E. T. Matson, "End-to-end Learning Approach for Autonomous Driving: A Convolutional Neural Network Model.," dalam *ICAART (2)*, 2019, hlm. 833–839.