TUGAS AKHIR – EE 184801

Rancang Bangun Model Deep Learning Pada Sistem Navigasi Mobil Otonom Dengan Sensor Kamera

I Made Guna Winangun
NRP 07111740000050

Dosen Pembimbing
Dr. Ir. Achmad Affandi, DEA.
Dr. Ir. Endroyono, DEA.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2021

--- halaman ini sengaja dikosongkan ---

TUGAS AKHIR – EE 184801

Rancang Bangun Model Deep Learning Pada Sistem Navigasi Mobil Otonom Dengan Sensor Kamera

I Made Guna Winangun
NRP 07111740000050

Dosen Pembimbing
Dr. Ir. Achmad Affandi, DEA.
Dr. Ir. Endroyono, DEA.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2021

--- halaman ini sengaja dikosongkan ---

TUGAS AKHIR – EE 184801

Design Model of Deep Learning in Autonomous Car Navigation System with Camera Sensor

I Made Guna Winangun
NRP 07111740000050

Supervisors
Dr. Ir. Achmad Affandi, DEA.
Dr. Ir. Endroyono, DEA.

DEPARTEMENT OF ELECTRICAL ENGINEERING
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2021

--- halaman ini sengaja dikosongkan ---

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "Rancang Bangun Model Deep Learning Pada Sistem Navigasi Mobil Otonom Dengan Sensor Kamera" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 21 Juli 2021

I Made Guna Winangun
NRP. 0711 17 4000 0050

--- halaman ini sengaja dikosongkan ---

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

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik

Pada

Bidang Studi Telekomunikasi
Departemen Teknik Elektro
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Ir. Achmad Affandi, DEA	Dr. Ir. Endroyono, DEA
NIP. 1965 1014 1990 02 1001	NIP. 1965 0404 1991 02 1001

SURABAYA
JULI, 2021

--- halaman ini sengaja dikosongkan ---

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

Nama : I Made Guna Winangun
Dosen Pembimbing I : Dr. Ir. Achmad Affandi, DEA
Dosen Pembimbing II : Dr. Ir. Endroyono, DEA

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

--- halaman ini sengaja dikosongkan ---

DESIGN MODEL OF DEEP LEARNING IN AUTONOMOUS CAR NAVIGATION SYSTEM WITH CAMERA SENSOR

Name : I Made Guna Winangun
Supervisor I : Dr. Ir. Achmad Affandi, DEA
Supervisor II : Dr. Ir. Endroyono, DEA

ABSTRACT

The phenomenon behind this research is the difficulty of computer vision in navigating a car in a situation that has never been passed by a car before. To overcome this phenomenon requires a new approach in navigating the car. The approach is the use of deep learning in navigating the car. In this study, we obtained a design of a deep learning model that can navigate the car. The model uses one of the deep learning architectures, namely convolutional neural network. This model is trained based on the data obtained from the simulator. The data taken is data in the form of steering direction with images coming from the camera installed on the car in the simulator. With this model, the car can be said to be an autonomous car because it can navigate itself. This model has a validation error of 0.0827 degrees. This model can navigate the car well at 15 – 20 speed units on the simulator.

Keywords: **Autonomous Car, Camera Sensors, Deep Learning, Navigation**

--- halaman ini sengaja dikosongkan ---

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Yang Maha Kuasa berkat kuasa-Nya sehingga penulis dapat melaksanakan mampu menyelesaikan tugas akhir dengan judul "**RANCANG BANGUN MODEL DEEP LEARNING PADA SISTEM NAVIGASI MOBIL OTONOM DENGAN SENSOR KAMERA**" dengan tepat waktu. Tugas akhir ini disusun sebagai salah satu persyaratan untuk menyelesaikan pendidikan S1 Bidang Studi Teknik Sistem Telekomunikasi dan Multimedia, Departemen Teknik Elektro, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember. Atas selesainya penyusunan tugas akhir ini, penulis mengucapkan terima kasih kepada:

1. Yang Maha Kuasa
2. Kepada orang tua dan sanak keluarga
3. Dosen – dosen dan civitas ITS
4. Rekan-rekan kampus

Penulis telah berusaha maksimal dalam penyusunan tugas akhir ini, namun besar harapan penulis akan saran dari pembaca agar nantinya laporan tugas akhir ini dapat menjadi referensi yang berguna bagi penelitian selanjutnya.

Surabaya, 21 Juli 2021

I Made Guna Winangun
NRP. 07111740000050

--- halaman ini sengaja dikosongkan ---

DAFTAR ISI

DAFTAR GAMBAR

--- halaman ini sengaja dikosongkan ---

DAFTAR TABEL

--- halaman ini sengaja dikosongkan ---

BAB 1
PENDAHULUAN

Pada bab ini akan dibahas terkait dengan latar belakang, perumusan masalah yang dihadapi, batasan permasalahan, tujuan, metodologi, sistematika penulisan, dan relevansi.

1.1. Latar Belakang

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 menjad 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 menavigasi kendaraannya. Sensor – sensor tersebut berupa lidar dan kamera.

Kamera[3] merupakan sensor yang digunakan untuk menangkap gambar. Kamera dapat memproses gambar menjadi berbagai perintah kerja pada suatu sistem dengan bantuan kecerdasan tambahan[4]. Kecerdasan itu yaitu Deep learning[5]. 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.[6] 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[7] yang dikatakan dapat menavigasi mobil dengan baik.

1.2. Rumusan Masalah

Kamera dapat digunakan sebagai sensor pada sistem navigasi dengan syarat kamera tersebut memiliki kecerdasan untuk memproses hasil tangkapan kamera. Perumusan masalah yang dihadapi dalam pengerjaan tugas akhir ini:

1. Bagaimana cara menavigasi mobil dengan sensor kamera?
2. Bagaimana Deep Learning dapat membantu dalam menavigasi mobil pada lingkungan yang belum pernah dilintasi?

1.3. Batasan Masalah

Hal-hal yang akan diperhatikan dalam penelitian ini adalah sebagai berikut:

1. Menggunakan data berupa gambar yang diambil dari simulator untuk merancang rancangan kecerdasan ini.
2. Kinerja diukur berdasarkan performa dari rancangan kecerdasan dalam mengendalikan mobil pada track yang sudah disediakan oleh simulator.
3. Simulator yang digunakan dalam mensimulasikan rancangan ini menggunakan open-source simulator oleh Udacity.

1.4. Tujuan

Tujuan tercapainya penelitian ini yaitu mendapatkan rancangan kecerdasan yang dapat menavigasi mobil.

1.5. Metodologi

Pada sub bab ini akan dibahas metodologi dalam perancangan ini. Terdapat bahan – bahan, peralatan, serta alur pengerjaan dari rancangan ini.

1.5.1. Bahan – Bahan

Bahan – bahan yang digunakan dalam perancangan ini yaitu data berupa gambar yang didapatkan dari simulator mobil.

1.5.2. Peralatan

Peralatan yang digunakan dalam perancangan ini yaitu:

1. Simulator (open-source simulator oleh Udacity)

2. Colaboratory (https://colab.research.google.com/) sebagai tempat untuk mengeksekusi Python di browser dengan akses gratis ke GPU dan tidak memerlukan konfigurasi
3. Flask & Socket.io untuk melakukan bi-directional client-server communication. Hal ini dibutuhkan untuk menghubungkan model dengan simulator
4. Numpy sebagai alat komputasi numerik pada bahasa pemrograman Python
5. matplotlib.pyplot untuk menampilkan hasil plot berbasis MATLAB
6. Keras sebagai framework machine learning
7. OpenCV untuk mengolah gambar
8. sklearn menyediakan berbagai algoritma supervised dan unsupervised
9. imgaug menyediakan algoritma augmentasi
10. pandas untuk memanipulasi dan analisis data
11. ntpath menyediakan fungsi os.path pada platfrom windows
12. random untuk melakukan algoritma pengacakan

1.5.3. Alur Pengerjaan

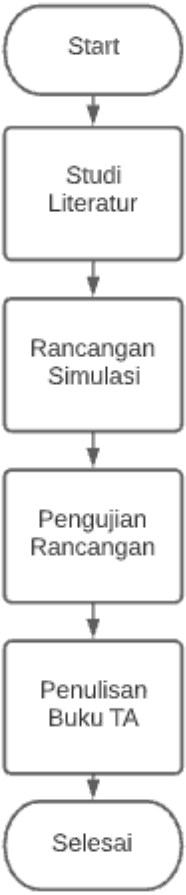
Dalam alur pengerjaan rancangan ini terdapat empat langkah alur pengerjaan. Alur tersebut digambarkan pada **Gambar 1.1**. Alur – alur pengerjaan tersebut yaitu:

1.5.3.1. Studi Literatur

Langkah pertama dalam metodologi ini yaitu melakukan studi literatur. Studi literatur merupakan proses peneliti mencari tinjauan – tinjauan yang menunjang dasar ilmu penelitian yang akan dilakukan. Selain itu, pada tahap ini dilakukan perancangan dalam melanjutkan proses selanjutnya. Perancangan tersebut yaitu menulis proposal, menginstal tools pemrograman, dan menginstal simulator.

1.5.3.2. Perancangan Simulasi

Pada langkah ini yang dilakukan yaitu merancangan rancangan model yang dapat menavigasi mobil secara otonom. Dalam perancangan ini terdapat berbagai bahan dan alat yang digunakan untuk dapat menyelesaikan rancangan.



Gambar 1.1 Diagram Alur Pengerjaan

1.5.3.3. Pengujian Rancangan

Pada langkah ini yang dilakukan yaitu melakukan pengujian terhadap rancangan yang berhasil dirancang. Pengujian – pengujian yang dilakukan yaitu pengujian rancangan dalam menavigasi mobil pada track 1, track 2, dan track 3.

1.5.3.4. Penyusunan Buku Laporan Tugas Akhir

Langkah terakhir yaitu penyusunan buku laporan tugas akhir. Dalam penyusunan ini akan dilaporkan performa model deep learning yang dirancang serta kesimpulannya.

1.6. Sistematika Penulisan

Laporan tugas akhir ini terdiri dari lima bab dengan sistematika penulisan sebagai berikut:

1. Bab 1: Pendahuluan

   Bagian pendahuluan berisi tentang latar belakang, perumusan masalah, maksud dan tujuan penulisan, pembatasan masalah, dan metode penelitian yang digunakan, serta sistematika penulisan.

2. Bab 2: Dasar Teori

   Pada bab ini berisi teori pendukung yaitu Mobil Otonom, Sistem Navigasi, Sensor Kamera, Computer Vision, Machine Learning, Deep Learning, Jaringan Syaraf Tiruan, Augmentation Technique, dan Simulator Mobil Otonom.

3. Bab 3: Desain dan Perancangan Simulasi

   Pada bab ini berisi langkah-langkah desain sistem yang akan menjadi bahan simulasi beserta dengan rancangan analisis simulasi.

4. Bab 4: Hasil dan Analisis

   Pada bab ini berisi hasil eksekusi desain dan perancangan model simulasi serta analisis kinerja dari model kecerdasan yang dihasilkan.

5. Bab 5: Penutup

   Pada bab ini akan diberikan kesimpulan dan saran terhadap rancangan kecerdasan dan hasil analisisnya.

1.7. Relavansi

Manfaat atau relevansi dari hasil penelitian ini yaitu terdapat kecerdasan yang dapat diterapkan pada kamera mobil sehingga mobil konvensional berpotensi menjadi mobil otonom.

BAB 2
DASAR TEORI

Pada bab ini akan dijelaskan mengenai istilah-istilah serta materi dasar yang digunakan dalam pengerjaan tugas akhir.

2.1. 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.[8] 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.[9]

SAE International mengeluarkan standar dalam tingkatan dari kemudi otomatis.[10] Terdapat enam tingkatan dalam standar tersebut. Tingkatan – tingkatan tersebut yaitu:

1. Tingkat (level) 0

   Otomasi pada level ini adalah kendaraan masih dikendalikan oleh manusia sehingga tidak ada fitur autonomous pada kendaraan di tingkat ini.

2. Tingkat (level) 1

   Pada tingkat ini fitur-fitur otomasi mulai diterapkan untuk mendukung aspek keselamatan, keamanan dan kenyamanan pengemudi selama berkendara. Fitur auto braking atau pengereman otomatis adalah ciri khas level 1 automation dimana suatu kendaraan hanya mampu melakukan satu tugas dalam satu kesempatan.

3. Tingkat (level) 2

   Pada tingkat ini kendaraan memiliki sistem otomatisasi parsial. Kendaraan memiliki minimal dua fitur otomatis seperti Steering dan Lane Control Assistant termasuk Traffic Jam Assistant, membuat mengemudi sehari-hari jauh lebih mudah. Sistem pengereman secara otomatis, akselerasi otomatis, dan perlahan–lahan mengambil alih sistem kendali.

4. Tingkat (level) 3

   Kendaraan otonomous level 3 mampu mengemudi sendiri, tetapi hanya dalam kondisi ideal dan dengan keterbatasan, seperti akses terbatas yang terbagi jalan raya dengan kecepatan tertentu. Meski tangan terlepas

dari kemudi, pengemudi tetap diharuskan di belakang kemudi. Seorang pengemudi manusia masih harus mengambil alih jika kondisi jalan berada di bawah yang ideal.

5. Tingkat (level) 4

Kendaraan otonom level 4 dapat mengemudi sendiri tanpa interaksi manusia (selain memasuki tujuan Anda) tetapi akan dibatasi untuk kasus penggunaan yang diketahui. Operasional kendaraan Autonomous level 4 ini masih dibatasi oleh hukum dan regulasi. Fitur autonomous Kendaraan level 4 masih dapat dioperasikan hanya pada lingkungan tertentu.

6. Tingkat (level) 5

Pada kendaraan Autonomous level mobil dapat dikatakan sebagai mobil tanpa pengemudi sejati. Kendaraan berkemampuan level 5 harus dapat memonitor dan bermanuver melalui semua kondisi jalan dan tidak memerlukan intervensi manusia apa pun, menghilangkan kebutuhan akan roda kemudi dan pedal. Meskipun banyak komponen teknologi artificial intelligent yang memungkinkan terwujudnya kendaraan ini, namun dikarenakan peraturan dan regulasi hukum, kendaraan Level 5 mungkin masih membutuhkan waktu beberapa tahun ke depan.

2.2. 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.[11] 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.

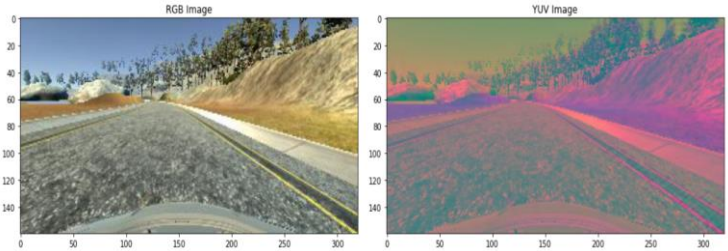
2.3. Sensor Kamera

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.[12] Perangkat ini dapat menjadi sensor pada suatu sistem elektronik.[13] Sensor kamera dapat menggantikan beberapa sensor sekaligus untuk mengurangi kompleksitas sistem.

2.4. Computer Vision

Computer Vision membahas bagaimana komputer dapat dirancang untuk mendapatkan pemahaman tingkat tinggi dari gambar atau video digital.[14] 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.[15] Dalam penggunaannya, computer vision berusaha untuk mengotomatiskan tugas yang dapat dilakukan oleh manusia secara visual.[16]

Salah satu Computer Vision Library yang terkenal adalah OpenCV[17]. Berbagai hal dapat dilakukan dalam proses pengolahan gambar dengan library ini. Salah satu contoh penggunaannya yaitu merubah gambar RGB menjadi YUV seperti yang digambarkan pada **Gambar 2.1**, dimana hal ini diperlukan dalam pre-processing input gambar pada arsitektur PilotNet. Selain itu, hal yang dibutuhkan dalam arsitektur ini yaitu memotong gambar untuk menentukan fitur, merubah ukuran pixel gambar, dan melakukan GaussianBlur pada gambar seperti yang digambarkan pada **Gambar 2.2**.



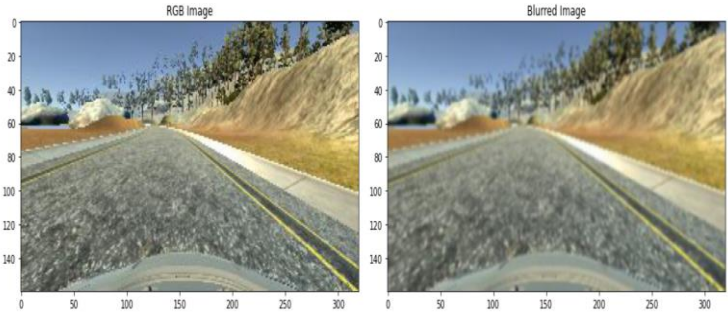
Gambar 2.1 Aplikasi konversi RGB ke YUV

Konversi nilai pixel RGB menjadi YUV dapat dilihat pada persamaan dibawah:

$$Y = 0.299R + 0.587G + 0.114B \quad (1)$$
$$U = -0.147R - 0.289G + 0.436B \quad (2)$$
$$V = 0.615R - 0.515G - 0.100B \quad (3)$$

Konversi nilai pixel YUV menjadi RGB dapat dilihat pada persamaan dibawah:

$$R = Y + 1.140V \quad (4)$$
$$G = Y - 0.395U - 0.581V \quad (5)$$
$$B = Y + 2.032U \quad (6)$$



Gambar 2.2 Aplikasi GaussianBlur terhadap suatu gambar

2.5. Machine Learning

Machine learning[18] 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.

Setiap algoritma tentunya memiliki keunggulan dan kelemahannya. Beberapa contoh tugas yang paling baik diselesaikan dengan pembelajaran mesin meliputi: (1) Pengenalan Pola, misalnya mengenali objek pada gambar, identitas wajah, ekspersi, dan ucapan; (2) Pengenalan Anomali, misalnya transaksi kartu kredit serta pola pembacaan sensor yang tidak biasa; dan (3) Prediksi, misalnya prediksi pergerakan harga saham, atau prediksi barang yang akan dibeli oleh seseorang berdasarkan histori belanjanya. Sementara itu, hal yang bukan merupakan pengenalan pola, anomali maupun prediksi sebaiknya

diselesaikan menggunakan metode analitis untuk menghemat proses komputasi. Dalam machine learning terdapat dua metode dalam proses pembelajaran yaitu supervised learning dan unsupervised learning.[19]

2.5.1. Supervised Learning

Supervised learning merupakan metode pembelajaran dimana data yang akan digunakan sebagai pembelajaran diberi pelabelan untuk dapat diklasifikasikan per kelas dan dijadikan sebagai model prediksi dari data tersebut. Metode ini disebut sebagai supervised learning karena proses pembelajaran algoritma dari dataset pembelajaran dapat dianggap sebagai guru yang mengawasi proses pembelajaran. Jawaban yang benar diketahui, kemudian algoritma iteratif membuat prediksi pada data pelatihan dan dikoreksi. Belajar berhenti ketika algoritma mencapai tingkat kinerja yang dapat diterima. Supervised learning dapat dibedakan menjadi permasalahan regresi dan permasalahan klasifikasi.

2.5.1.1. Permasalahan Regresi

Permasalahan regresi merupakan permasalahan dimana nilai variabel keluarannya merupakan bilangan terukur, misalnya: berat, tinggi, dan ukuran.

2.5.1.2. Permasalahan Klasifikasi

Permasalahan klasifikasi merupakan permasalahan dimana nilai variabel keluarannya merupakan kategori, misalnya: penyakit, atau bukan penyakit.

2.5.2. Unsupervised Learning

Unsupervised learning merupakan metode pembelajaran dimana data pembelajaran yang akan digunakan tidak diberikan pelabelan. Tujuan unsupervised learning adalah untuk memodelkan struktur atau distribusi yang mendasari data untuk mempelajari lebih lanjut tentang data tersebut.

Metode pembelajaran ini disebut unsupervised learning karena tidak seperti supervised learning, tidak disediakan jawaban yang benar dan tidak ada guru. Algoritma dibiarkan sendiri untuk menemukan dan menyajikan struktur yang menarik dalam data. Permasalahan yang dapat diselesaikan menggunakan metode unsupervised learning dapat dikelompokkan menjadi permasalahan clustering dan asosiasi. Beberapa algoritma unsupervised learning yang cukup populer misalnya K-Means untuk permasalahan clustering dan Apriori untuk permasalahan asosiasi.

2.5.2.1. Permasalahan Clustering

Permasalahan clustering adalah menemukan cluster yang melekat dalam data, misalnya mengelompokkan pelanggan berdasarkan perilaku belanja mereka.

2.5.2.2. Permasalahan Asosiasai

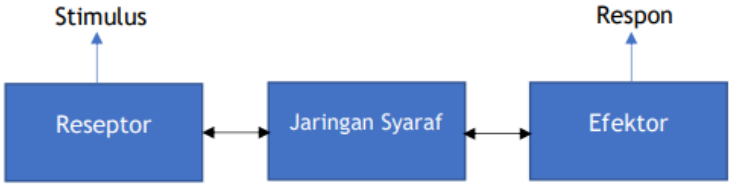
Permasalahan asosiasi adalah permasalahan di mana Anda ingin menemukan aturan yang menggambarkan sebagian besar data Anda, seperti orang yang membeli X juga cenderung membeli Y.

2.6. 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.

2.7. Jaringan Syaraf Tiruan

Sistem jaringan syaraf tiruan[20] 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 bagian seperti pada **Gambar 2.3**. Bagian – bagian tersebut yaitu reseptor, jaringan syaraf, dan efektor.

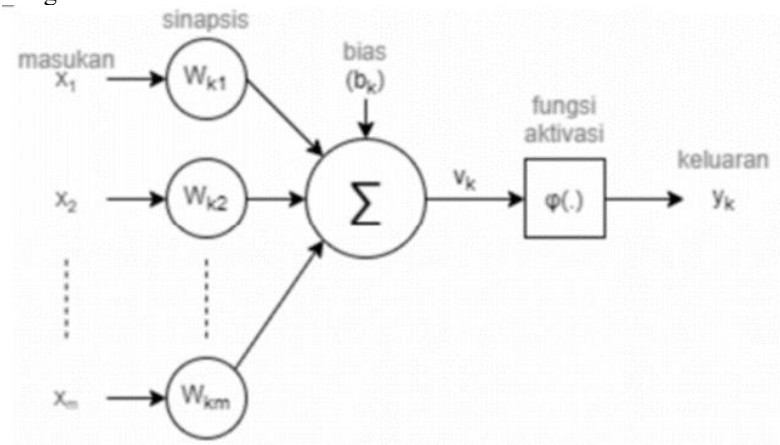


Gambar 2.3 Skema sistem jaringan syaraf utama pada manusia

Reseptor berfungsi untuk mengubah stimulus dari rangsangan yang membawa informasi untuk diproses menuju ke jaringan syaraf. Sedangkan effector berfungsi untuk merubah informasi dari jaringan syaraf untuk dijadikan sebagai keluaran.

Unit pemrosesan dalam jaringan syaraf (neural network) disebut sebagai neuron atau node. Dengan kata lain suatu sistem proses komputasional yang terinspirasi oleh bagaimana jaringan syaraf pada manusia bekerja. Dalam implementasinya neuron dapat direpresentasikan sebagai node yang menghubungkan antar jaringan satu dengan lainnya.

Jaringan syaraf tiruan pada manusia seperti yang digambarkan pada **Gambar 2.4** apabila dikelompokkan berdasarkan fungsinya, dapat dibagi menjadi tiga elemen dasar yakni Sinapsis, Summing Junction, dan Fungsi Aktivasi.



Gambar 2.4 Diagram jaringan syaraf tiruan (perceptron) [21]

1. Sinapsis

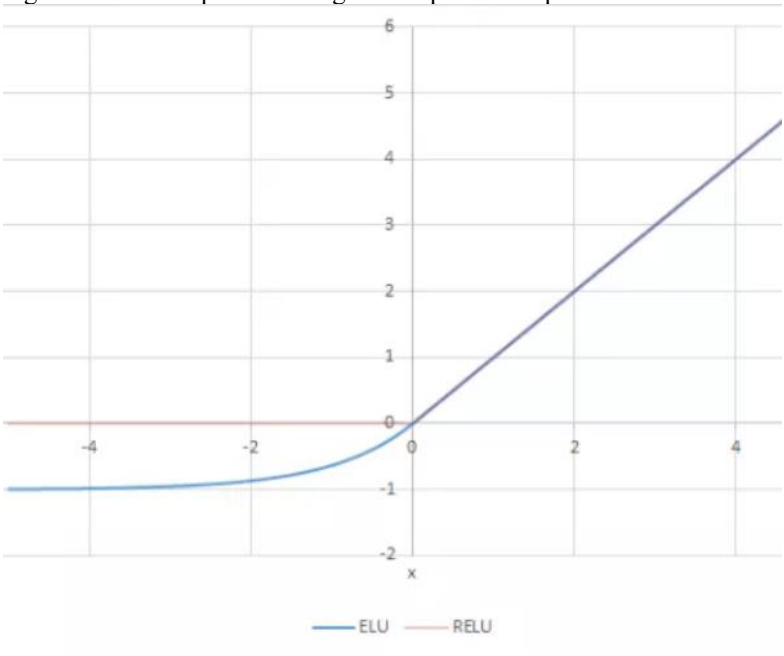
Sinapsis merupakan elemen yang terdapat nilai pembobotan dan sinyal masukan. Nilai pembobotan merupakan nilai acak dengan rentang nilai negatif hingga nilai positif.

2. Summing Junction

Summing Junction berfungsi sebagai penjumlahan untuk masing masing nilai pembobotan sebelumnya. Nilai keluaran pada neuron diperoleh dari fungsi aktivasi. Elemen ini digunakan untuk membatasi rentang sinyal keluaran pada neuron

3. Fungsi Aktivasi

Fungsi aktivasi berfungsi sebagai penentu bobot dari output yang dihasilkan dari jaringan ini. Salah satu contoh fungsi aktivasi yaitu fungsi Exponential Linear Unit atau disingkat elu. Fungsi ini akan mengembalikan nilai x untuk nilai x pada sumbu x lebih dari nol. Sedangkan untuk nilai x kurang dari nol akan mengembalikan nilai secara logaritmik. Untuk plot dari fungsi ini dapat dilihat pada **Gambar 2.5**.



Gambar 2.5 Plot dari fungsi ELU

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.

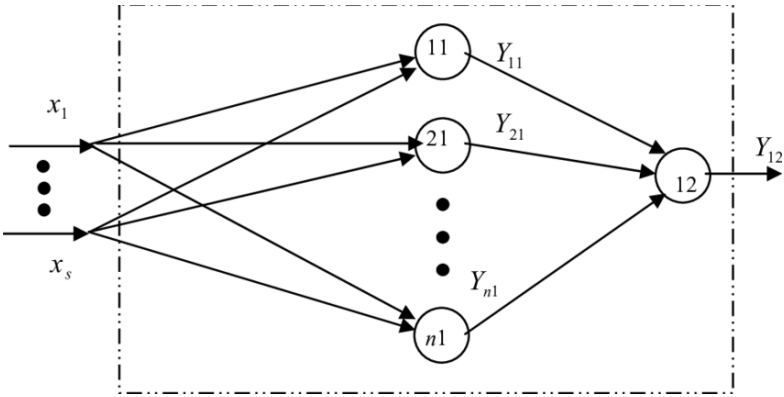
2.7.1. Single Layer Feedforward Neural Network

Single Layer Feedforward Neural Network / Perceptron Dianggap sebagai generasi pertama dari jaringan saraf, perceptron hanyalah model komputasi dari satu neuron. Mereka dipopulerkan oleh Frank Rosenblatt pada awal 1960. Mereka tampaknya memiliki algoritma pembelajaran yang sangat kuat dan banyak klaim besar dibuat untuk apa yang bisa mereka pelajari.

Pada tahun 1969, Menganalisis apa yang dapat dilakukan oleh perceptron dan menunjukkan keterbatasannya. Banyak orang mengira keterbatasan ini berlaku untuk semua model jaringan saraf. Namun, prosedur pembelajaran perceptron hingga saat ini masih banyak digunakan untuk tugas-tugas dengan ukuran vektor fitur yang sangat besar (berisi jutaan fitur).

2.7.2. Multi-Layer Feedforward Neural Network

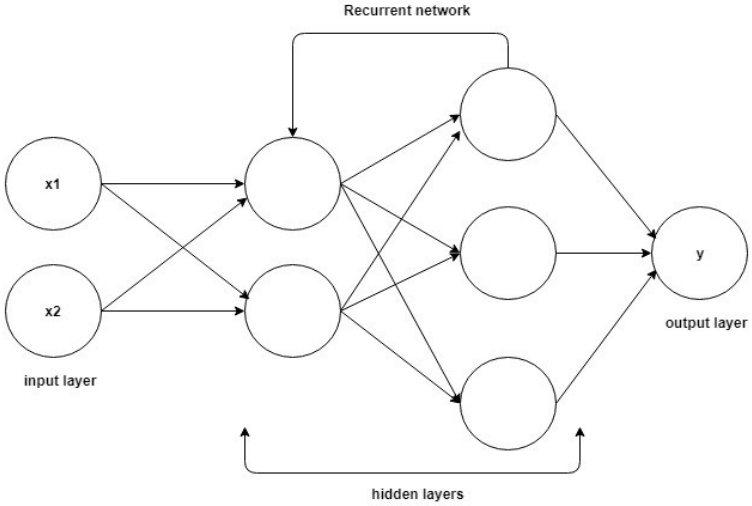
Jaringan multi-layer feedforward[22] memiliki arsitektur yang mirip dengan jaringan perceptron namun dengan adanya satu atau lebih hidden layer yang mana node perhitunganya disebut sebagai hidden neuron, atau disebut sebagai hidden unit. Hidden neuron berfungsi untuk melakukan proses perhitungan dengan orde tinggi. **Gambar 2.6** menunjukan jaringan fully connected multi-layer feedforward dimana tiap node di setiap layer terhubung ke tiap-tiap node yang lain pada layer berikutnya. Namun jika terdapat salah satu node yang tidak terhubung disebut sebagai partially connected.



Gambar 2.6 Multi-layer feedfoward neural network[23]

2.7.3. Recurrent Neural Network

Recurrent neural network merupakan tipe jaringan yang berbeda dengan lainnya. Recurrent network mempunyai satu atau lebih feedback loop. **Gambar 2.7** menunjukan ilustrasi recurrent network yang mana keluaran tiap neuron dijadikan sebagai masukan dari neuron lainnya. Tujuan dari feedback loop jaringan yaitu untuk meningkatkan kemampuan dalam proses pembelajaran.



Gambar 2.7 Recurrent Neural Network[24]

RNN memiliki kemampuan yang handal karena tersusun atas distributed hidden state yang mampu menyimpan banyak informasi seiring berjalannya waktu secara efisien dan non-linear dynamics yang membuat sistem mampu melakukan proses update pada hidden state kompleks. Dengan jumlah neuron dan waktu yang mencukupi, RNN mampu melakukan proses komputasi apa saja.

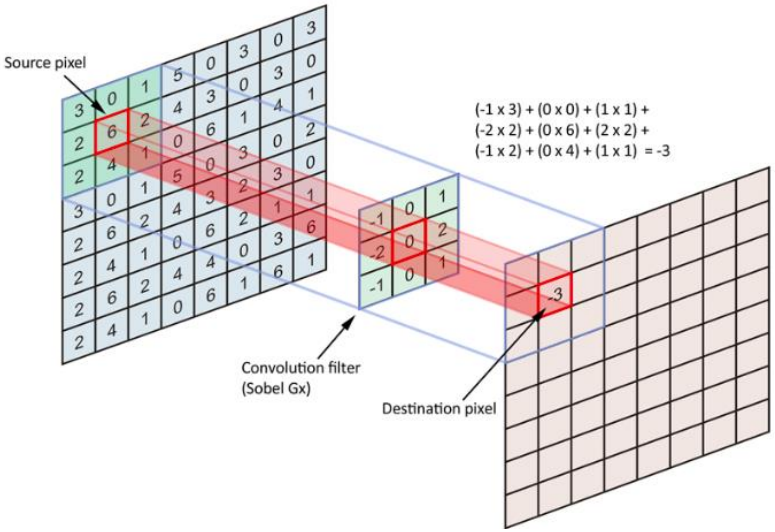
Walaupun memiliki kemampuan komputasi yang luar biasa, proses training pada RNN tergolong kompleks dibandingkan jenis jaringan syaraf tiruan yang lain. Proses training RNN lebih rumit karena dilakukan proses backpropagation pada banyak layer sedangkan gradient pada layer tersebut membesar dan mengecil secara eksponensial dan tidak bisa diprediksi. Proses training RNN dilakukan dalam waktu yang panjang sehingga ukuran gradient dapat meledak dikarenakan membesar berlebihan secara eksponensial dan dapat hilang karena mengecil berlebihan secara eksponensial. Walaupun sudah diawali dengan initial weight yang ideal, sulit untuk mendeteksi target output terkini dari input beberapa saat yang lalu.

2.7.4. Symmetrically Connected Networks

Symmetrically Connected Networks memiliki arsitektur seperti Recurrent Neural Network, tetapi koneksi antar unit simetris (memiliki bobot yang sama di kedua arah). Symmetrically Connected Networks jauh lebih mudah untuk dianalisis daripada Recurrent Neural Network. Jaringan ini juga lebih dibatasi dalam apa yang dapat mereka lakukan karena mereka mematuhi fungsi energi. Jaringan yang terhubung secara simetris tanpa unit tersembunyi disebut Hopfield Nets, sementara itu jaringan yang terhubung secara simetris dengan unit hidden layer disebut Boltzmann Machine.

2.7.5. 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. Pada **Gambar 2.8** menggambarkan bagaimana proses konvolusi terjadi. Jaringan konvolusional adalah jenis jaringan saraf khusus yang menggunakan konvolusi sebagai pengganti perkalian matriks umum pada salah satu layer atau lebih.

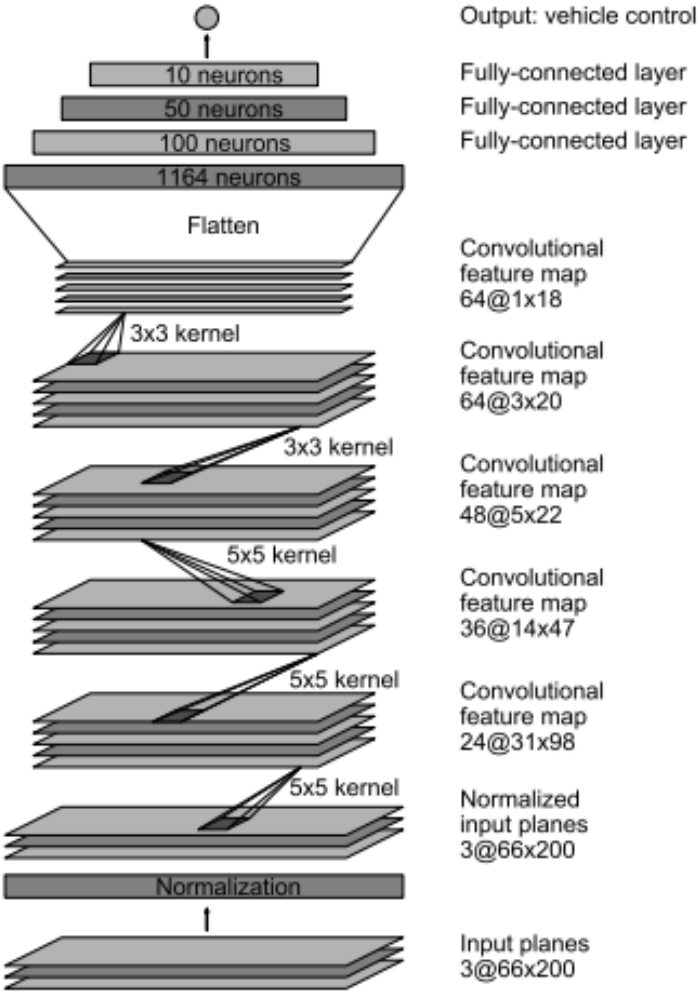


Gambar 2.8 Konvolusi pada CNN[25]

Salah satu contoh arsitektur CNN adalah arsitektur yang dirancang oleh NVIDIA yaitu PilotNet. Arsitektur ini dikatakan mampu untuk meminimalisir kompleksitas sistem.[6] Total jaringan ini terdiri dari sembilan layer yaitu lapisan normalisasi, lima lapisan konvolusional, dan tiga lapisan terkoneksi penuh yang digambarkan pada **Gambar 2.9**. 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.

Terdapat beberapa hal yang perlu diperhatikan dalam pembentukan dari suatu model kecerdasan buatan. Hal – hal tersebut yaitu arsitektur kecerdasan buatan, compiler dari model, dan proses pembelajarannya. Pada arsitektur kecerdasan buatan terdiri dari beberapa komponen tersebut yaitu jumlah layer dan jenis layer. Dalam arsitektur PilotNet terdiri dari dua jenis layer yaitu layer konvolusional, dan layer terkoneksi penuh. Pada compiler terdiri dari beberapa parameter yaitu optimizer dan loss. Dan yang terakhir pada proses pembelajaran terdiri

dari beberapa parameter. Parameter tersebut yaitu input, label, epoch, validation data.



Gambar 2.9 Arsitektur PilotNet[6]

2.8. 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.[26] 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.

2.9. 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 **Tabel 2.1**.

Tabel 2.1 Simulator - simulator mobil otonom[27]

Name	Graphic require-ment	Data collection	OS requirements (Mainly sup-port)	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

Simulator Udacity merupakan salah satu contoh simulator untuk menyimulasikan mobil otonom. Simulator ini dapat dijalankan dengan dua mode. Mode pertama yaitu mode training. Mode ini digunakan sebagai pengambilan data dengan cara mengemudikan dan merekam mobil yang ada di simulator tersebut. Dalam training mode, simulator ini mengambil sampel data dengan periode rata – rata 85 milisekon. Mode selanjutnya yaitu mode otonom. Mode ini digunakan untuk menguji apakah model yang dimiliki sudah mampu untuk melewati rintangan pada arena mobil secara otomatis. Pada mode ini, total waktu rata – rata yang dibutuhkan model untuk merespon tiap input gambar menjadi arah kemudi yaitu 500 milisekon.

BAB 3
DESAIN DAN PERANCANGAN SIMULASI

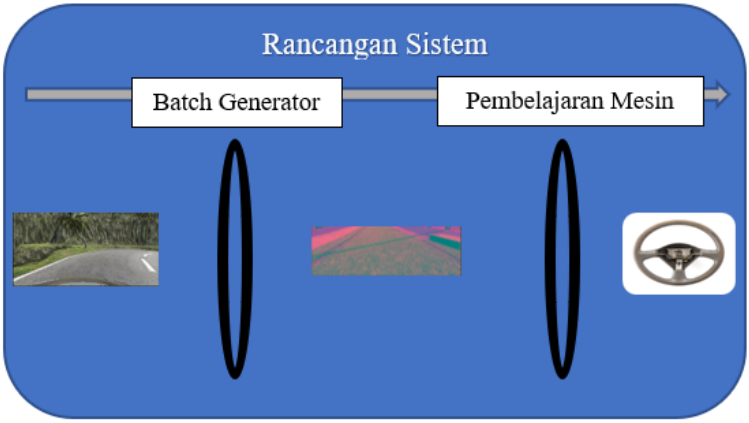
Pada Bab ini akan dibahas tentang perancangan model yang dapat menavigasi mobil.

3.1. Rancangan Sistem Secara Umum

Pada rancangan ini akan dirancang sebuah sistem yang dapat mengolah data berupa gambar sehingga menghasilkan output berupa arah kemudi mobil yang digambarkan pada **Gambar 3.1**. Rancangan ini terdiri dari serangkaian tahapan pengolahan data dalam perancangannya yang digambarkan pada **Gambar 3.2**. Tahapan – tahapan tersebut yaitu:

1. Pengolahan gambar sebelum diproses (pre-processing) dengan 5 langkah:
   1) Membuang background pada data gambar
   2) Merubah format gambar RGB menjadi YUV
   3) Melakukan GaussianBlur dengan kernel 3x3
   4) Mengubah dimensi gambar menjadi 200x66 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 jaringan syaraf untuk menciptakan model deep learning yang dapat menghasilkan arah kemudi dengan input gambar.

Gambar 3.1 Gambaran sistem secara umum



Gambar 3.2 Serangkaian tahapan pengolahan data dalam rancangan sistem

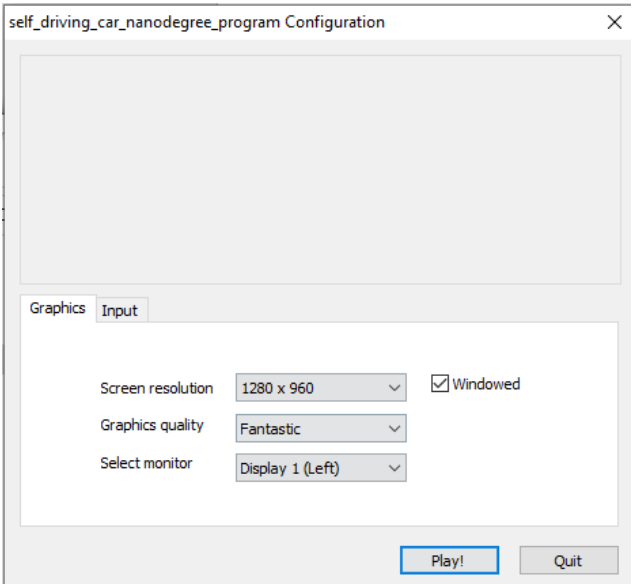
3.2. Konfigurasi Perangkat Lunak Simulator

Dalam konfigurasi perangkat lunak simulator, akan ditampilkan beberapa konfigurasi – konfigurasi dalam mempersiapkan perancangan model deep learning.

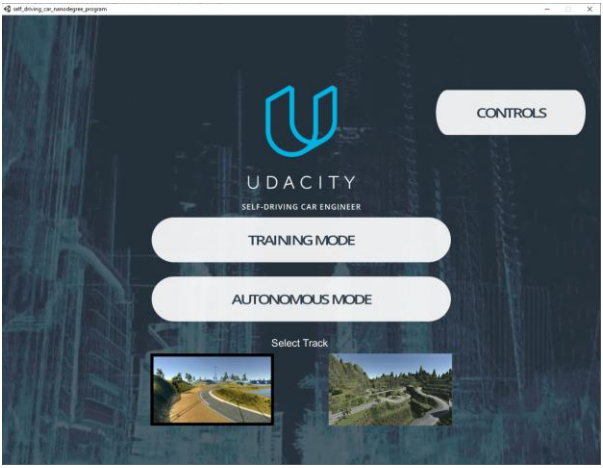
Tahap ini merupakan konfigurasi untuk menjalankan aplikasi simulator. Dalam konfigurasi ini terdapat pemilihan resolusi layar, kualitas grafis, serta pengaturan tombol kendali. Pemilihan konfigurasi ini dapat disesuaikan dengan kualitas komputer yang digunakan dalam penelitian namun pemilihan konfigurasi yang berbeda tidak akan mempengaruhi kualitas data yang akan diambil. Setelah konfigurasi dipilih, tekan tombol Play! untuk masuk ke konfigurasi selanjutnya dalam simulator.

Tahap ini merupakan konfigurasi untuk pemilihan mode beserta trek pada aplikasi simulator. Dalam pemilihan mode, terdapat mode training dan mode otonom. Pada mode training, mobil dapat digerakkan secara manual untuk mendapatkan data. Sedangkan pada mode otonom dapat sebagai tempat uji coba model deep learning yang sudah diciptakan. Dalam konfigurasi ini terdapat dua pilihan trek mobil sebagai alternatif pemilihan trek. Untuk pengambilan data akan digunakan trek satu dan dua. Sedangkan untuk menguji model akan dilakukan pada track 1, track

2, dan track 3. Aplikasi ini dapat diunduh pada link yang tertera pada lampiran.



Gambar 3.3 Tampilan Awal Aplikasi Simulasi

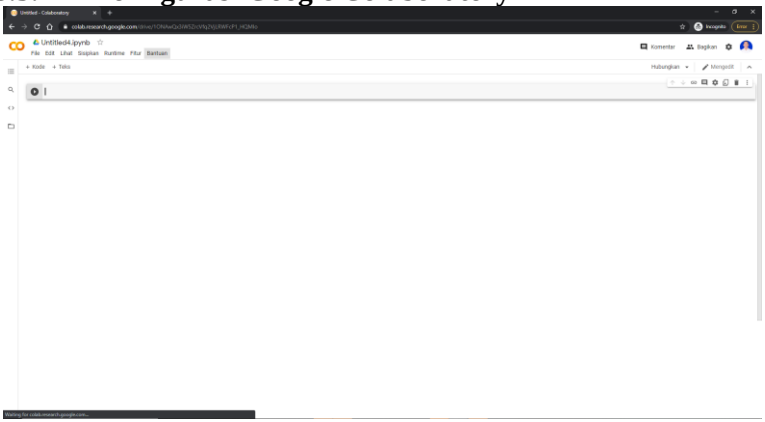


Gambar 3.4 Pemilihan mode beserta track

Gambar 3.5 Mode Training

Gambar 3.6 Mode Autonomous

3.3. Konfigurasi Google Colaboratory

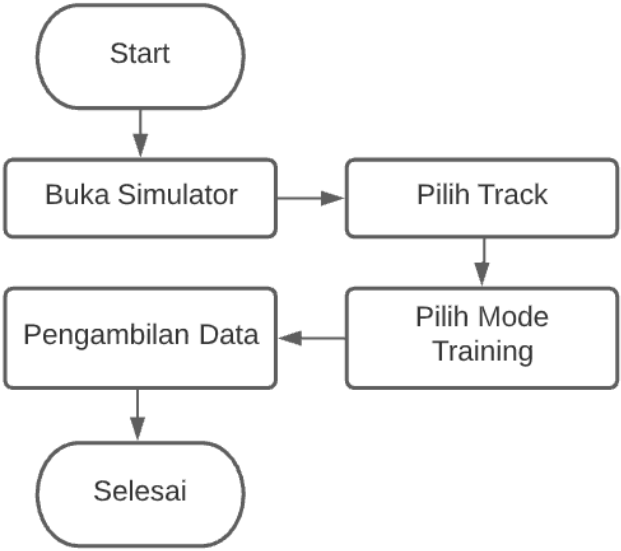


Gambar 3.7 Google Colaboratory sebagai tempat pemrograman

Buka https://colab.research.google.com/ dan login menggunakan akun Google. Konfigurasi ini merupakan tempat untuk merancang berbagai program. Akan digunakan bahasa python dengan library – library seperti keras, opencv, dan lain – lain. Klik tombol +Kode untuk menambahkan sel kode sebagai sel untuk tiap kode untuk konfigurasi selanjutnya. Setelah menuliskan kode lalu klik tombol bergambar mainkan atau bertuliskan jalankan sel.

3.4. Pengumpulan Data dari Simulator

Pada tahapan ini, peneliti akan mengemudikan mobil pada simulator untuk 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. Alur pengambilan data digambarkan pada **Gambar 3.8.**



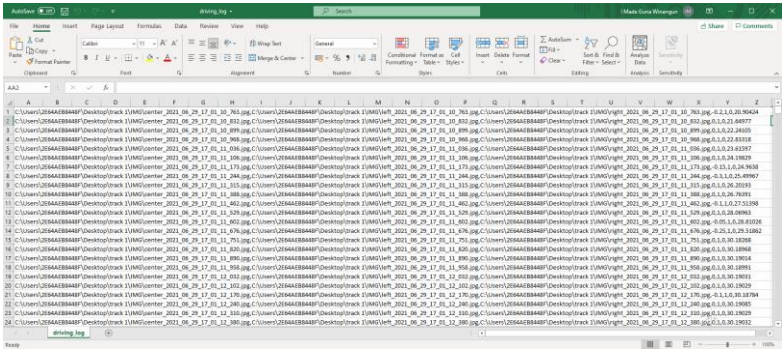
Gambar 3.8 Diagram alur pengambilan data

Gambar 3.9 Tiga putaran kesatu pada track 1

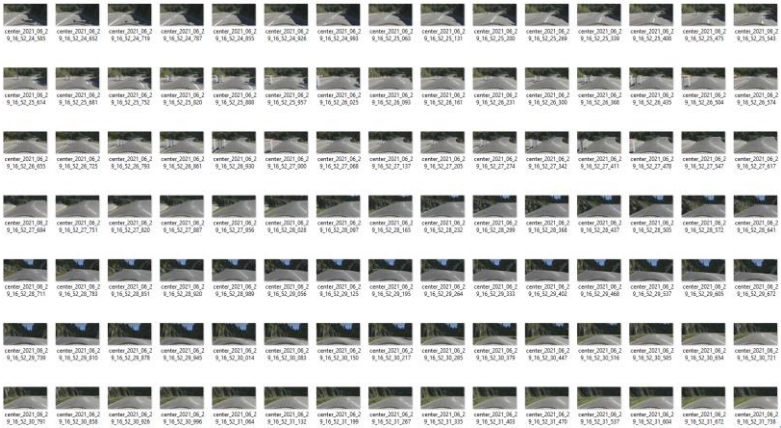
Gambar 3.10 Tiga putaran kedua pada track 1

Gambar 3.11 Dua putaran kesatu pada track 2

Gambar 3.12 Dua putaran kedua pada track 2



Gambar 3.13 Tampilan data yang telah dikoleksi

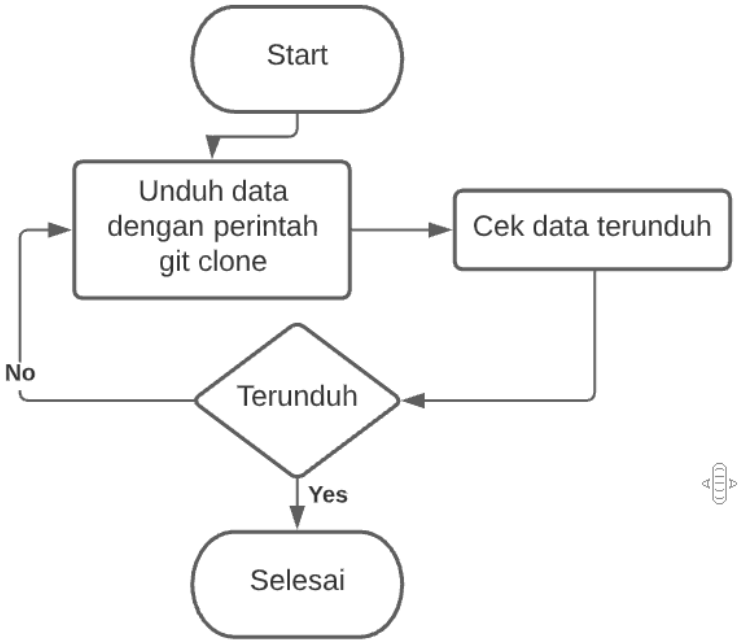


Gambar 3.14 Kumpulan gambar dari tampilan kamera pada mobil di simulator

Setelah data kemudi mobil didapatkan, data tersebut akan disimpan dalam repositori Git (https://github.com/imadegunawinangun/final-track). Hal ini dilakukan agar data kemudian dapat digunakan pada Google Colab secara mudah. 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.

3.5. Pengunduhan Data

Pada tahapan ini, akan dilakukan pengunduhan data yang telah didapatkan. Pengunduhan ini dilakukan agar data dapat diolah dalam environment Google Colab. Untuk melakukan pengunduhan dilakukan perintah git clone. Setelah data tersebut dikloning, peneliti memastikan bahwa data telah ada dengan perintah ls. Untuk alur pengunduhan data tergambarkan pada **Gambar 3.15.**



Gambar 3.15 Diagram alur pengunduhan data

```
!git clone https://github.com/imadegunawinangun/final-track
```

```
Cloning into 'final-track'...
remote: Enumerating objects: 42135, done.
remote: Total 42135 (delta 0), reused 0 (delta 0), pack-reused 42135
Receiving objects: 100% (42135/42135), 531.62 MiB | 36.70 MiB/s, done.
Resolving deltas: 100% (1/1), done.
Checking out files: 100% (42135/42135), done.
```

```
!ls final-track/final
```

```
track1  track2
```

Gambar 3.16 Pengunduhan data

Pada **Gambar 3.16** terlihat folder track 1 dan track 2 telah terbaca. Di dalam masing – masing folder tersebut terdapat data yang telah direkam pada simulator. Data tersebut terdiri dari file driving_log.csv menunjukkan kumpulan data sedangkan folder IMG merupakan kumpulan dari hasil gambar.

3.6. Import Library

Sebelum melanjutkan ketahapan pengolahan data, akan dilakukan import library agar dapat menggunakan tools – tools yang diperlukan. Implementasi import library pada Google Colaboratory digambarkan pada **Gambar 3.17.**

```
import os
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import keras
from keras.models import Sequential
from keras.optimizers import Adam
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from imgaug import augmenters as iaa
import cv2
import pandas as pd
import ntpath
import random
```

Gambar 3.17 Library yang diperlukan dalam perancangan model

3.7. Visualisasi Data

Dalam memvisualisasikan data, alur yang terjadi tergambarkan pada **Gambar 3.18.** Namun sebelum dilakukan visualisasi, perlu dilakukan pengolahan data. Pengolahan data dimulai dengan dengan menghapus nama file yang tidak dibutuhkan (contoh: C:\Users\2E64AEB8448F\Desktop\final\track2\IMG\center_2021_07_20_20_08_24_044.jpg menjadi center_2021_07_20_20_08_24_044.jpg). **Gambar 3.19** menunjukkan data sebelum dibersihkan sedangkan **Gambar 3.20** menunjukkan data setelah dibersihkan.

Gambar 3.18 Diagram alur visualisasi data

Gambar 3.19 Membaca data menggunakan library pandas

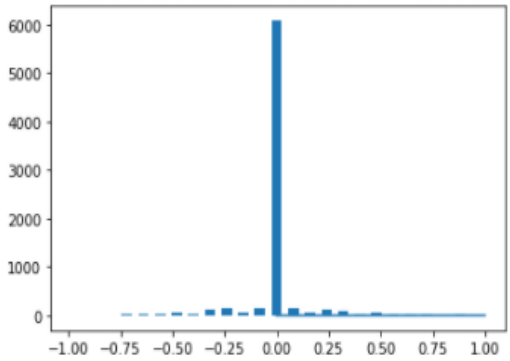
	center	left	right	steering	throttle	reverse	speed
0	center_2021_07_20_19_46_02_347.jpg	left_2021_07_20_19_46_02_347.jpg	right_2021_07_20_19_46_02_347.jpg	0.00	1	0	10.70280
1	center_2021_07_20_19_46_02_415.jpg	left_2021_07_20_19_46_02_415.jpg	right_2021_07_20_19_46_02_415.jpg	-0.15	1	0	11.35815
2	center_2021_07_20_19_46_02_482.jpg	left_2021_07_20_19_46_02_482.jpg	right_2021_07_20_19_46_02_482.jpg	-0.30	1	0	12.10183
3	center_2021_07_20_19_46_02_551.jpg	left_2021_07_20_19_46_02_551.jpg	right_2021_07_20_19_46_02_551.jpg	-0.50	1	0	12.93047
4	center_2021_07_20_19_46_02_618.jpg	left_2021_07_20_19_46_02_618.jpg	right_2021_07_20_19_46_02_618.jpg	-0.65	1	0	13.34923

Gambar 3.20 Setelah dilakukan proses pemotongna nama file

Langkah selanjutnya yaitu membuat histogram dari data yang telah dikumpulkan guna memvisualisasikan distribusi data yang telah dikumpulkan. Nilai pada arah kemudi berupa bilangan dengan rentang nilai -1 sampai 1. nilai -1 menunjukkan arah kemudi 90 derajat berlawanan arah jarum jam serangkan nilai 1 menunjukkan arah kemudi 90 derajat searah jarum jam. Agar memudahkan dalam membaca data pada saat memvisualisasikannya, maka nilai akan dibuat secara interval. Nilai interval yang diambil yaitu 25 karena dari data yang didapat, nilai 25 dapat merepresentasikan grafik yang dapat merepresentasikan nilai arah kemudi relatif maksimal secara baik.

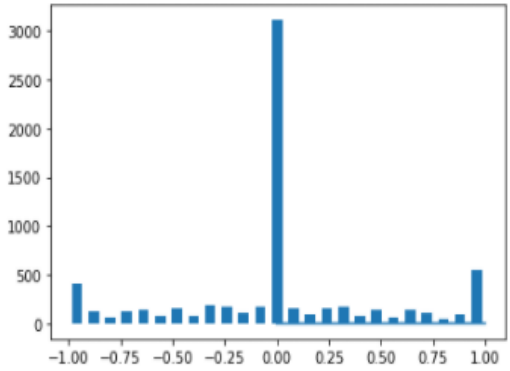
Pada grafik yang ditampilkan pada **Gambar 3.21** dan **Gambar 3.22**, sumbu x merupakan banyaknya interval pada grafik. Sedangkan sumbu y merupakan jumlah data di tiap – tiap intervalnya.

```
[-0.96 -0.88 -0.8  -0.72 -0.64 -0.56 -0.48 -0.4  -0.32 -0.24 -0.16 -0.08
  0.    0.08  0.16  0.24  0.32  0.4   0.48  0.56  0.64  0.72  0.8   0.88
  0.96]
```



Gambar 3.21 Visualisasi data track 1

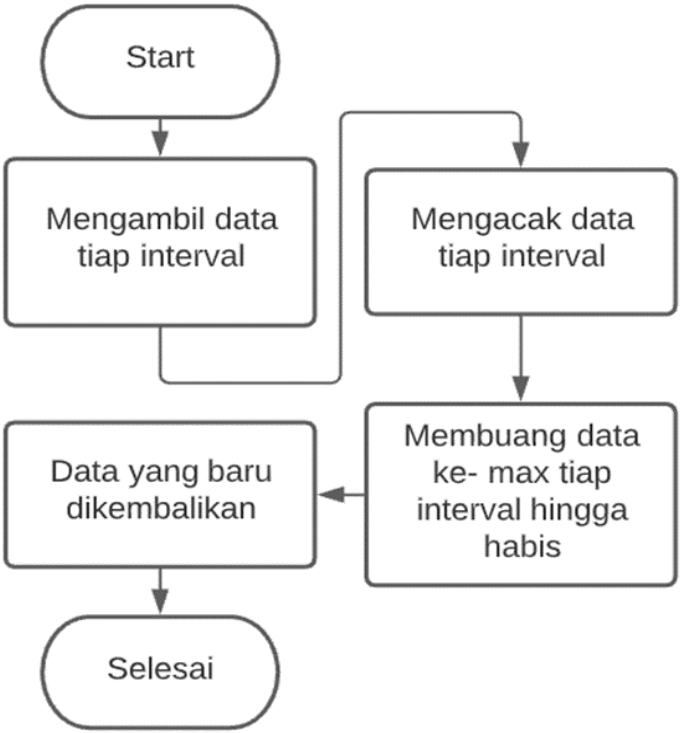
```
[-0.96 -0.88 -0.8  -0.72 -0.64 -0.56 -0.48 -0.4  -0.32 -0.24 -0.16 -0.08
  0.    0.08  0.16  0.24  0.32  0.4   0.48  0.56  0.64  0.72  0.8   0.88
  0.96]
```



Gambar 3.22 Visualisasi data track 2

3.8. Data Balancing

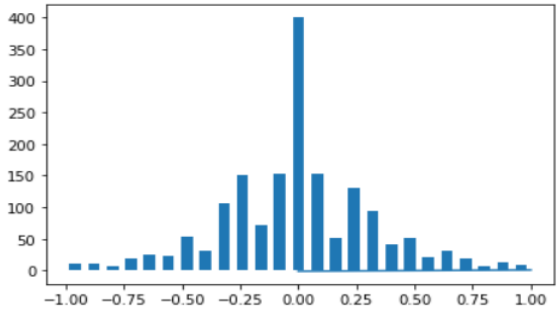
Dalam mempersiapkan data perlu dilakukan penyetabilan data agar tidak terjadi bias terhadap model yang akan dibuat. Alur penyeimbangan data dapat dilihat di **Gambar 3.23.** Untuk mencegah bias tersebut, pada track 1 untuk jumlah sampel pada tiap interval akan dibatasi dengan jumlah 400. Jumlah tersebut dipilih karena membuat data dapat merepresentasikan situasi berkendara yang cukup menurut penulis. Hal ini dapat dilihat pada **Gambar 3.24.** Sehingga total data yang awalnya berjumlah 7367 berubah menjadi 1690. Sedangkan untuk track 2 tiap interval dibatasi dengan jumlah 1500 sehingga data total awal 6677 menjadi 5065. Hal ini dapat dilihat pada **Gambar 3.25.**



Gambar 3.23 Diagram alur penyeimbangan data

```
df1 = data_balancing(data1, num_bins1, 400, center1, bins1)
hist, _ = np.histogram(df1['steering'], (num_bins1))
plt.bar(center1, hist, width=0.05)
plt.plot((np.min(df1['steering']), np.max(df1['steering'])))
```

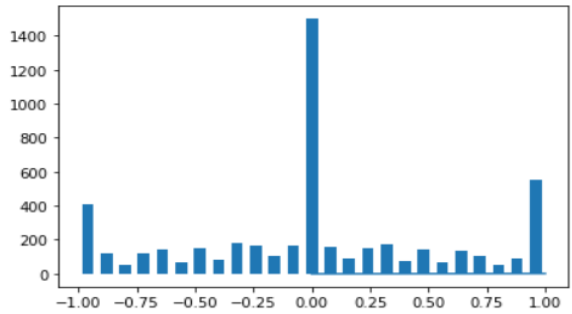
```
total data: 7367
removed: 5677
remaining: 1690
[<matplotlib.lines.Line2D at 0x7f8e15569510>]
```



Gambar 3.24 Distribusi data pada track 1

```
df2 = data_balancing(data2, num_bins2, 1500, center2, bins2)
hist, _ = np.histogram(df2['steering'], (num_bins1))
plt.bar(center1, hist, width=0.05)
plt.plot((np.min(df2['steering']), np.max(df2['steering'])))
```

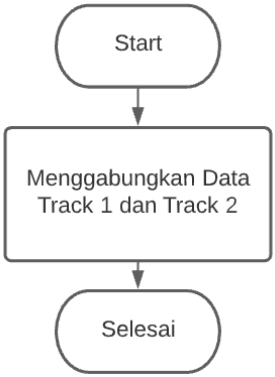
```
total data: 6677
removed: 1612
remaining: 5065
[<matplotlib.lines.Line2D at 0x7f8e14f9bd50>]
```



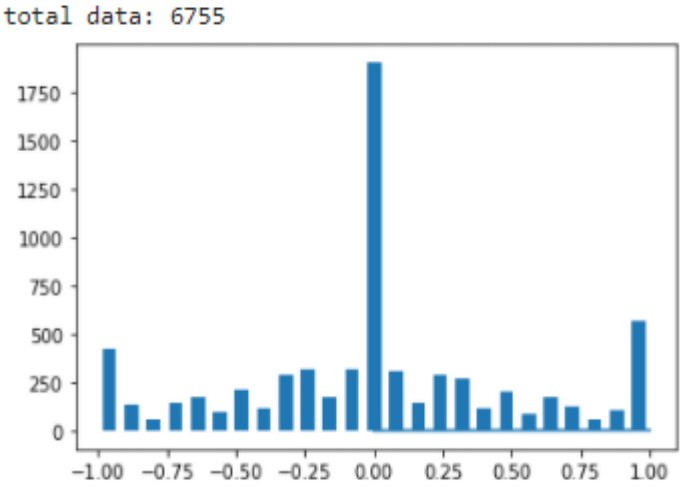
Gambar 3.25 Distribusi data pada track 2

3.9. Penggabungan Data Track 1 dan Track 2

Pada tahapan ini akan dilakukan penggabungan data track 1 dan track 2. Alur penggabungan dapat dilihat pada **Gambar 3.26.** Dengan penggabungan ini maka total data yang akan melalui proses selanjutnya yaitu 6755 yang dapat dilihat pada **Gambar 3.27.**



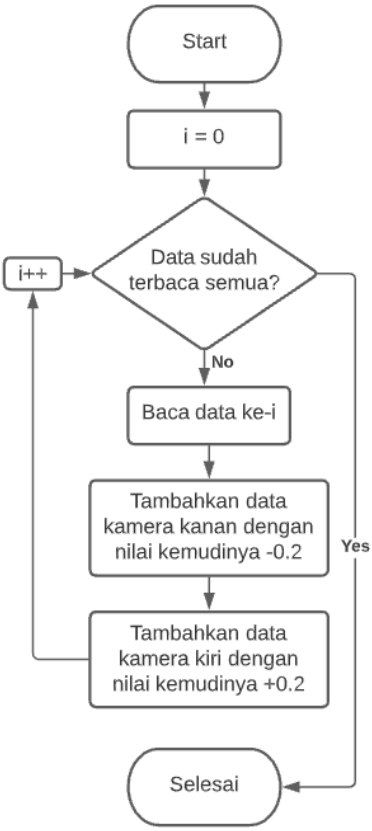
Gambar 3.26 Diagram alur penggabungan data track 1 dan track 2



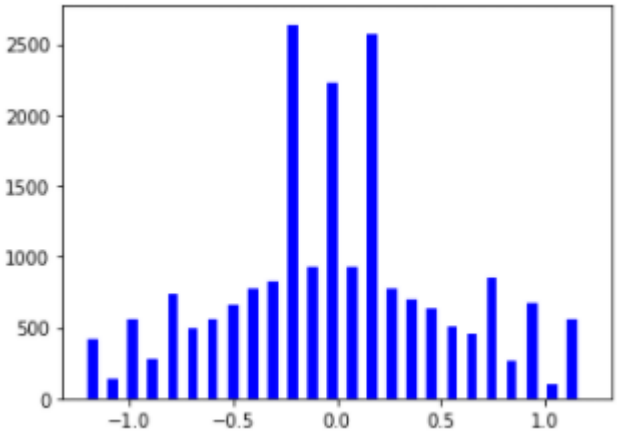
Gambar 3.27 Data setelah digabungkan

3.10. Penggabungan Kamera Kiri, Tengah dan Kanan

Distribusi data sebelumnya yang ditunjukkan pada **Gambar 3.27** merupakan gambaran dari distribusi arah kemudi terhadap gambar kamera tengah. Untuk memperkaya data maka dilakukan penggabungan, yang alur penggabungannya digambarkan pada **Gambar 3.28,** kamera kiri dan kanan sehingga jumlah data menjadi 20265. Untuk kamera kiri akan ditambah nilai relatif +0.2 terhadap arah kemudi kamera tengah sedangkan kamera kanan akan ditambah nilai relatif -0.2. Sehingga hasil penggabungan gambar tersebut menghasilkan persebaran nilai kemudi yang ditampilkan pada **Gambar 3.29.**



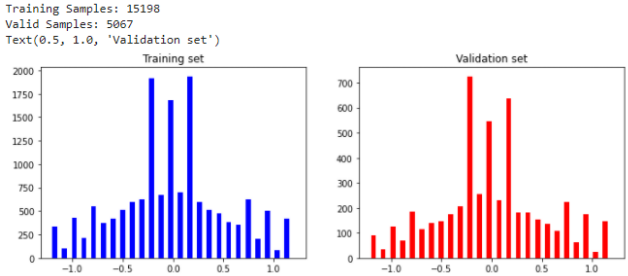
Gambar 3.28 Diagram alur penggabungan kamera kiri, tengah, dan kanan



Gambar 3.29 Distribusi data setelah data kamera kanan dan kamera kiri ditambahkan

3.11. Pemisahan Data Menjadi Data Training dan Validasi

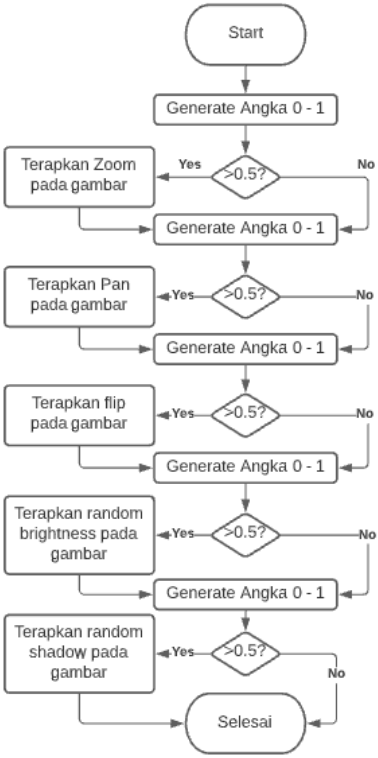
Setelah data tersebut digabungkan, gunakan fungsi train split test untuk mendapatkan data yang akan dilatih dan data yang akan diuji. Perbandingan data training dengan data validasi memiliki nilai perbandingan 0.75:0.25. Seed pada pengacakan akan ditentukan dengan nilai 0. Hal ini dilakukan agar mendapatkan hasil penelitian yang konsisten dan nilai 0 menghasilkan grafik yang serupa pada data validasi dengan data training. Hasil dari pemisahan data digambarkan pada **Gambar 3.30.**



Gambar 3.30 Memisahkan dataset menjadi data validasi dan data pelatihan

3.12. Augmentasi Data

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. Diagram alur dari proses ini digambarkan pada **Gambar 3.31.** 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.



Gambar 3.31 Diagram alur fungsi augmentasi gambar

```
def zoom(image):
  zoom = iaa.Affine(scale=(1, 1.3))
  image = zoom.augment_image(image)
  return image
def pan(image):
  pan = iaa.Affine(translate_percent= {"x" : (-0.1, 0.1), "y": (-0.1, 0.1)})
  image = pan.augment_image(image)
  return image
def img_random_brightness(image):
    brightness = iaa.Multiply((0.2, 1))
    image = brightness.augment_image(image)
    return image
def img_random_flip(image, steering_angle):
    image = cv2.flip(image,1)
    steering_angle = -steering_angle
    return image, steering_angle
def random_shadow(img):
    ht, wd , ch = img.shape
    x1, y1 = wd * np.random.rand(), 0
    x2, y2 = wd * np.random.rand(), ht
    xm, ym = np.mgrid[0:ht, 0:wd]

    mask = np.zeros_like(img[:, :, 1])
    mask[(ym - y1) * (x2 - x1) - (y2 - y1) * (xm - x1) > 0] = 1

    cond = mask == np.random.randint(2)
    s_ratio = np.random.uniform(low=0.6, high=0.9)

    hls = cv2.cvtColor(img, cv2.COLOR_RGB2HLS)
    hls[:, :, 1][cond] = hls[:, :, 1][cond] * s_ratio
    image = cv2.cvtColor(hls, cv2.COLOR_HLS2RGB)
    return image
```

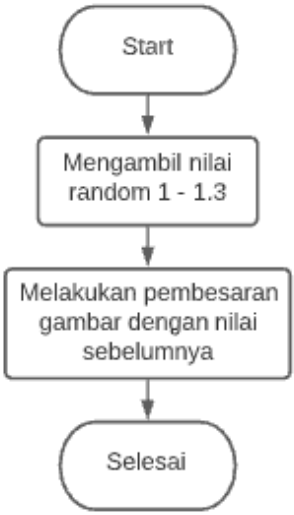
Gambar 3.32 Fungsi - fungsi pembentuk fungsi augmentasi

```
def random_augment(image, steering_angle):
    image = mpimg.imread(image)
    if np.random.rand() < 0.5:
      image = pan(image)
    if np.random.rand() < 0.5:
      image = zoom(image)
    if np.random.rand() < 0.5:
      image = img_random_brightness(image)
    if np.random.rand() < 0.5:
      image, steering_angle = img_random_flip(image, steering_angle)
    if np.random.rand() < 0.5:
      image = random_shadow(image)

    return image, steering_angle
```

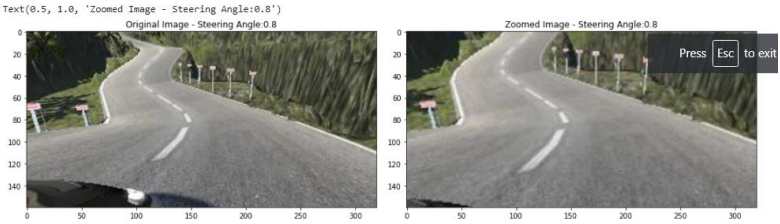
Gambar 3.33 Fungsi augmentasi gambar

3.12.1. Zoom

Fungsi merubah skala gambar yaitu fungsi yang memperbesar atau memperkecil tampilan data gambar. Tampilan tersebut memiliki data dengan dimensi pixel yang sama dengan gambar sebelum dimodifikasi. Selain itu, gambar asli dengan gambar yang sudah dilakukan modifikasi memiliki arah kemudi yang sama. Untuk pembesaran gambarnya dari skala 1 sampai dengan skala 1.3. Diagram alur dari fungsi ini digambarkan pada **Gambar 3.34.**



Gambar 3.34 Diagram alur fungsi zoom

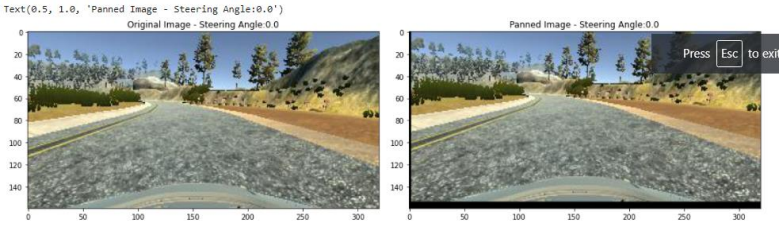


Gambar 3.35 Hasil dari fungsi zoom

3.12.2. Pan

Fungsi menggeser gambar yaitu fungsi yang menggeser pixel tiap gambar. Untuk pixel yang melewati frame yang telah ditentukan akan diabaikan nilainya. Sedangkan nilai pada pixel yang tidak kosong akan bernilai nol yang diindikasikan dengan warna hitam. Untuk arah kemudi pada data gambar asli dan gambar yang telah dimodifikasi memiliki nilai yang sama. Diagram alur dari fungsi ini digambarkan pada **Gambar 3.36.**

Gambar 3.36 Diagram alur fungsi Pan / penggeseran

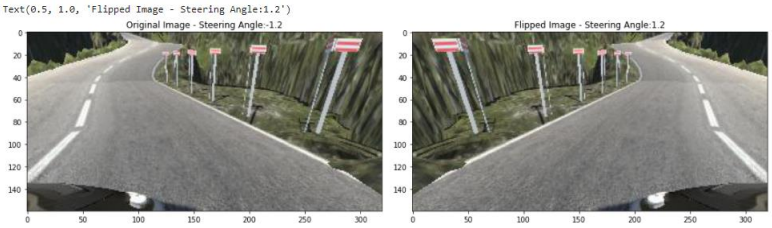


Gambar 3.37 Hasil dari fungsi penggeseran

3.12.3. Flip

Fungsi mencerminkan gambar yaitu fungsi yang memberikan hasil cerminan dari data yang diinputkan. Penggunaan fungsi pencerminan pada kasus ini akan dilakukan pencerminan terhadap sumbu vertikal. Nilai kemudi untuk gambar asli dengan gambar yang telah dimodifikasi memiliki nilai yang berbeda. Perbedaan nilai tersebut yaitu nilai kemudi untuk gambar yang telah dimodifikasi sama dengan nilai negatif dari nilai kemudi untuk gambar asli. Diagram alur dari fungsi ini digambarkan pada **Gambar 3.38**

Gambar 3.38 Diagram alur fungsi flip

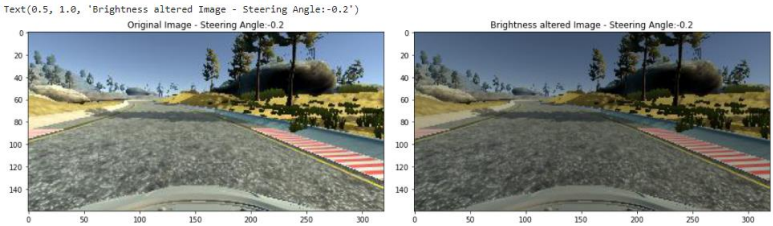


Gambar 3.39 Hasil dari fungsi flip / pencerminan

3.12.4. Brightness / Kecerahan

Fungsi merubah kecerahan gambar yaitu fungsi yang dilakukan untuk merubah kecerahan gambar asli menjadi lebih cerah atau lebih gelap. Untuk nilai ditiap pixelnya akan berubah tergantung dari seberapa perubahan kecerahan yang terjadi pada gambar. Sedangkan untuk nilai arah kemudi untuk gambar asli sama dengan nilai arah kemudi untuk gambar yang telah dimodifikasi. Diagram alur dalam mengubah kecerahan gambar dapat dilihat pada **Gambar 3.40.**

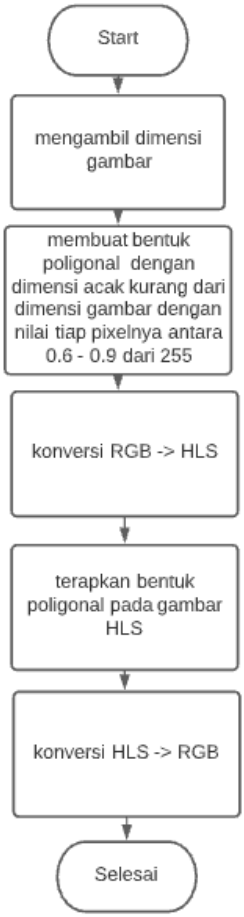
Gambar 3.40 Diagram alur fungsi brightness / kecerahan



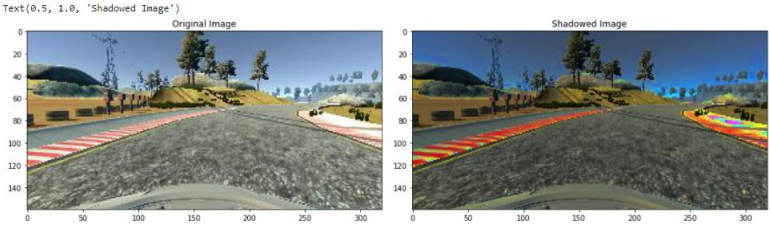
Gambar 3.41 Hasil dari fungsi kecerahan

3.12.5. Shadow

Fungsi menambahkan bayangan yaitu fungsi yang dilakukan untuk menambahkan bayangan pada gambar. Untuk nilai ditiap pixelnya akan berubah tergantung dari seberapa perubahan kecerahan yang terjadi pada gambar. Sedangkan untuk nilai arah kemudi untuk gambar asli sama dengan nilai arah kemudi untuk gambar yang telah dimodifikasi. Diagram alur dari fungsi ini digambarkan pada **Gambar 3.42**



Gambar 3.42 Diagram alur fungsi shadow



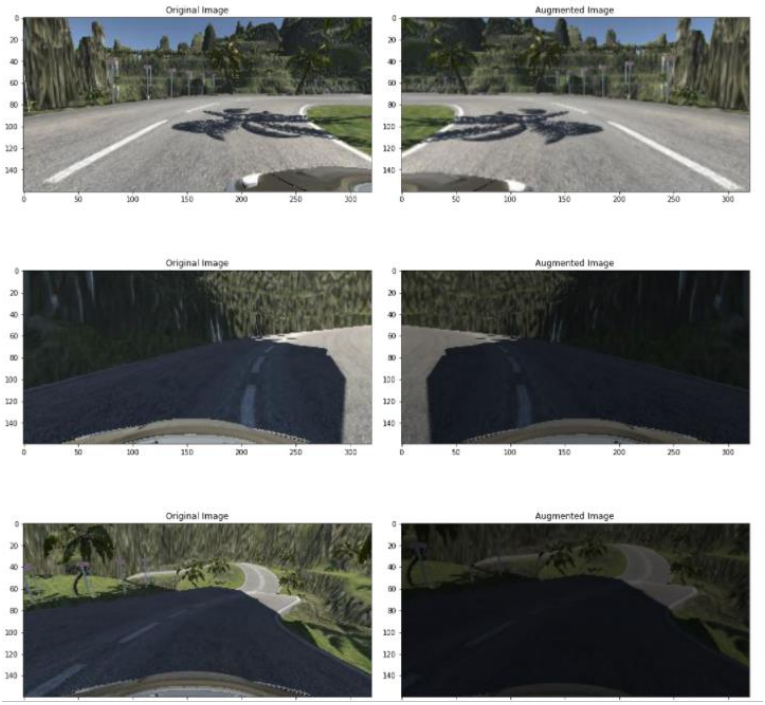
Gambar 3.43 Hasil dari fungsi penambahan bayangan

3.12.6. Pengujian Augmentasi Data

Sebelum melanjutkan ke tahap selanjutnya, akan dilakukan pengujian terhadap fungsi augmentasi. Pengujian ini dilakukan untuk mengetahui apakah fungsi augmentasi sudah melakukan mekanisme modifikasi secara acak. Pada pengujian ini akan dilakukan terhadap tiga data secara acak. Hasil dari pengujian dapat dilihat pada **Gambar 3.45.**

```
ncol = 2
nrow = 10

fig, axs = plt.subplots(nrow, ncol, figsize=(15, 50))
fig.tight_layout()

for i in range(10):
  randnum = random.randint(0, len(image_paths) - 1)
  random_image = image_paths[randnum]
  random_steering = steerings[randnum]

  original_image = mpimg.imread(random_image)
  augmented_image, steering = random_augment(random_image, random_steering)

  axs[i][0].imshow(original_image)
  axs[i][0].set_title("Original Image")

  axs[i][1].imshow(augmented_image)
  axs[i][1].set_title("Augmented Image")
```

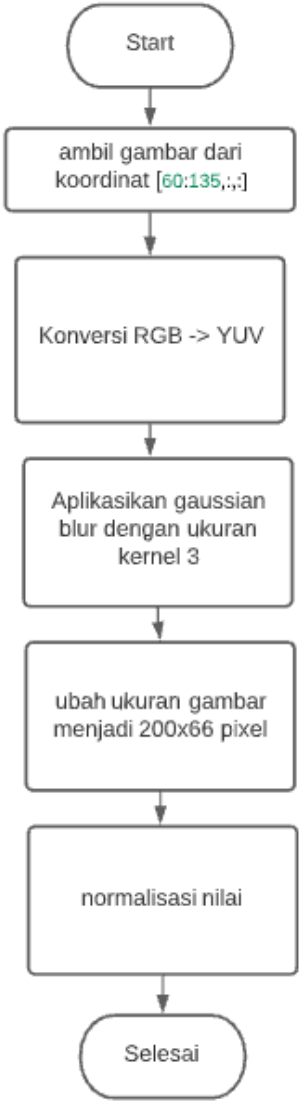
Gambar 3.44 Fungsi untuk memanggil fungsi augmentasi dan menampilkannya



Gambar 3.45 Hasil dari fungsi augmentasi terhadap beberapa gambar pelatihan

3.13. Pre-processing

Pada tahapan ini akan dilakukan proses yang dinamakan pre-processing. Alur diagramnya dapat dilihat pada **Gambar 3.46.** 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.



Gambar 3.46 Diagram alur tahapan pre-processing

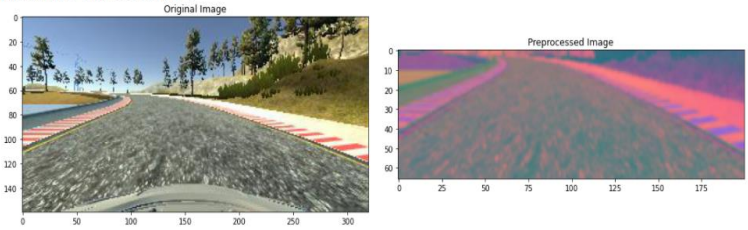
```python
def img_preprocess(img):
    img = img[60:135,:,:]
    img = cv2.cvtColor(img, cv2.COLOR_RGB2YUV)
    img = cv2.GaussianBlur(img,  (3, 3), 0)

    img = cv2.resize(img, (200, 66))

    img = img/255
    return img
```

Gambar 3.47 Fungsi pre-processing

Setelah merancang fungsi yang dapat melakukan pre-processing, selanjutnya akan dilakukan pengujian terhadap fungsi tersebut. Hal ini dilakukan untuk memastikan fungsi ini berjalan sesuai dengan rencana. Hasil dari fungsi ini dapat dilihat pada **Gambar 3.48.**

```
image = image_paths[1221]
original_image = mpimg.imread(image)
preprocessed_image = img_preprocess(original_image)

fig, axs = plt.subplots(1, 2, figsize=(15, 10))
fig.tight_layout()
axs[0].imshow(original_image)
axs[0].set_title('Original Image')
axs[1].imshow(preprocessed_image)
axs[1].set_title('Preprocessed Image')
```

```
Text(0.5, 1.0, 'Preprocessed Image')
```



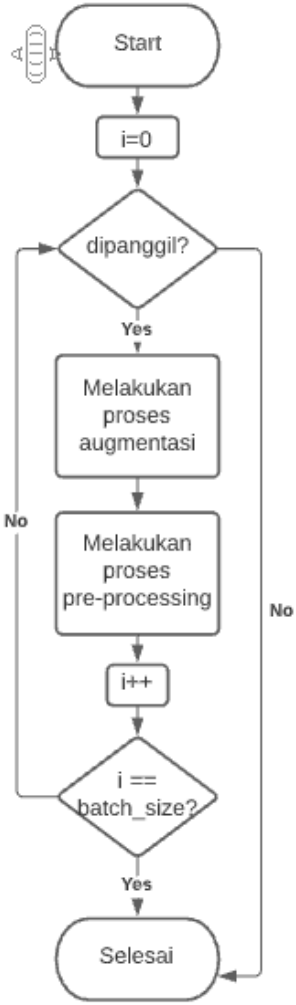
Gambar 3.48 Pengujian fungsi pre-processing

3.14. Batch Generator

Pada saat proses pelatihan dibutuhkan gambar yang telah diolah (pre-processing) sebagai input dengan jumlah yang banyak. Hal ini mengakibatkan memory komputasi yang digunakan akan termakan cukup banyak dan menyebabkan kegagalan memori. Hal ini dapat diatasi dengan cara menggunakan batch generator.

Bacth generator yaitu suatu fungsi yang dapat menghasilkan gambar pada saat dipanggil lalu gambar tersebut akan dihapus setelah

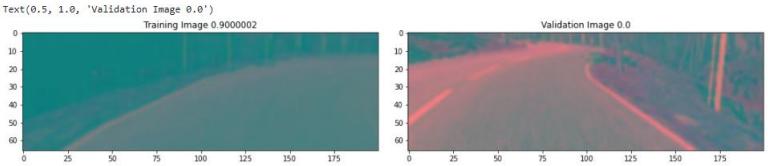
digunakan sehingga memori tidak penuh. Diagram alur generator ini dapat dilihat pada **Gambar 3.49.**



Gambar 3.49 Diagram alur fungsi batch generator

```python
def batch_generator(image_paths, steering_ang, batch_size, istraining):

  while True:
    batch_img = []
    batch_steering = []

    for i in range(batch_size):
      random_index = random.randint(0, len(image_paths) - 1)

      if istraining:
        im, steering = random_augment(image_paths[random_index], steering_ang[random_index])

      else:
        im = mpimg.imread(image_paths[random_index])
        steering = steering_ang[random_index]

      im = img_preprocess(im)
      batch_img.append(im)
      batch_steering.append(steering)
    yield (np.asarray(batch_img), np.asarray(batch_steering))
```

Gambar 3.50 Fungsi batch generator



Gambar 3.51 Uji coba pemanggilan data melalui batch generator

3.15. Rancangan Arsitektur

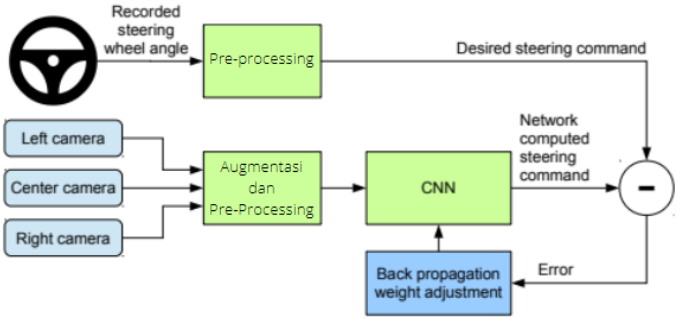
Rancangan arsitektur yang digunakan untuk perancangan model yang akan dibangun menggunakan arsitektur PilotNet. Arsitektur ini tergambarkan pada **Gambar 2.9.** Untuk optimizer yang digunakan yaitu Adam dengan learning rate 0.001. Sedangkan untuk fungsi evaluasi loss yang digunakan yaitu mean squared error. Nilai error pada fungsi ini yaitu perbedaan nilai arah kemudi yang dihasilkan oleh model dengan nilai arah kemudi yang didapatkan pada data.

```
def nvidia_model():
    model = Sequential()
    model.add(keras.layers.Conv2D(24, 5, 2, input_shape=(66, 200, 3), activation='elu'))
    model.add(keras.layers.Conv2D(36, 5, 2, activation='elu'))
    model.add(keras.layers.Conv2D(48, 5, 2, activation='elu'))
    model.add(keras.layers.Conv2D(64, 3, activation='elu'))
    model.add(keras.layers.Conv2D(64, 3, activation='elu'))
#   model.add(Dropout(0.5))


    model.add(Flatten())

    model.add(Dense(100, activation = 'elu'))
#   model.add(Dropout(0.5))

    model.add(Dense(50, activation = 'elu'))
#   model.add(Dropout(0.5))

    model.add(Dense(10, activation = 'elu'))
    #model.add(Dropout(0.5))

    model.add(Dense(1))

    optimizer = Adam(lr=1e-3)
    model.compile(loss='mse', optimizer=optimizer)
    return model
```

Gambar 3.52 Implementasi arsitektur PilotNet pada Tensorflow

3.16. Pelatihan Model

Setelah model dan batch generator siap, maka akan dilakukan proses pelatihan 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 diagram alur proses pelatihan digambarkan pada **Gambar 3.53.**



Gambar 3.53 Proses Pembelajaran

Tabel 3.1 Parameter Pelatihan Model

Parameter	**Nilai**
Batch Size	100
Training_data	batch_generator(X_train, y_train, batch_size, 1)
validation_data	batch_generator(X_valid, y_valid, batch_size, 0)
epoch	25
steps_per_epoch	len(X_train)*10/batch_size
validation_steps	len(X_valid)*3/batch_size
Callbacks	EarlyStopping val_loss patience=5

```
batch_size = 100
history = model.fit(batch_generator(X_train, y_train, batch_size, 1),
                    validation_data=batch_generator(X_valid, y_valid, batch_size, 0),
                    epochs=25,
                    steps_per_epoch=len(X_train)*5/batch_size,
                    validation_steps=len(X_valid)*3/batch_size,

                    callbacks=[
                        tf.keras.callbacks.EarlyStopping(
                        monitor='val_loss',
                        patience=5,
                        restore_best_weights=True
                        )
                    ])
```

Gambar 3.54 Implementasi proses pelatihan model

3.17. Menyimpan Model

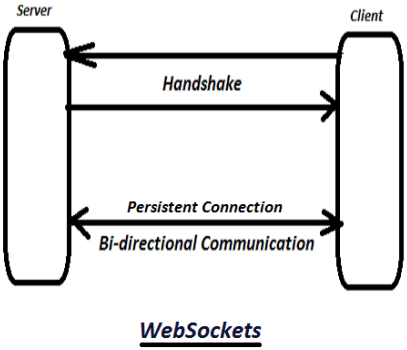
Setelah proses pelatihan selesai, model akan disimpan agar bisa digunakan untuk menavigasi mobil pada simulator. Untuk menyimpan digunakan fungsi save dan import file dari google colaboratory yang digambarkan pada **Gambar 3.55.**

```
model.save('model.h5')
from google.colab import files
files.download('model.h5')
```

Gambar 3.55 Implementasi pengunduhan model yang telah terlatih pada Google Colaboratory

3.18. Model 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. Hal ini digambarkan pada **Gambar 3.56** dimana terjadi komunikasi Bi-directional Communication antara server dan client.



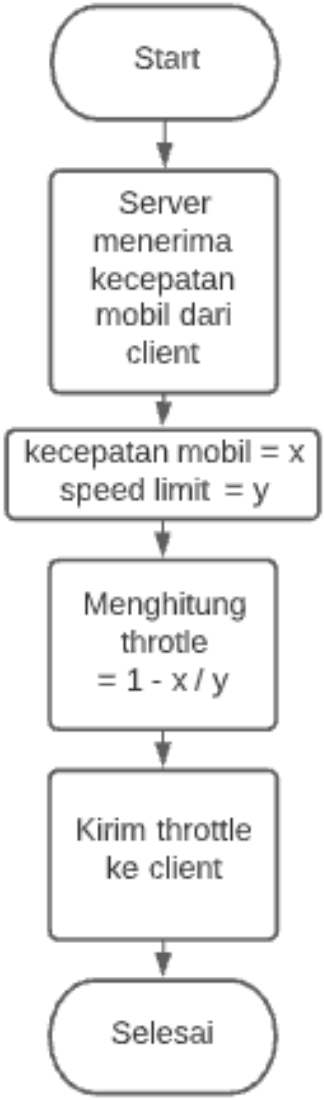
Gambar 3.56 Alur komunikasi antar simulator dan program

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. Alur ini tergambarkan pada **Gambar 3.57.**

Navigasi throttle mobil dilakukan karena mobil (client) menunggu input throttle dari server. Sehingga server mengirimkan throttle dengan nilai pada persamaan **7.** Hal ini dapat menjadikan mobil melakukan manuver hingga mencapai kecepatan maksimal atau melakukan pengereman saat kecepatan mobil melebihi kecepatan maksimal. Alur ini tergambarkan pada **Gambar 3.58.**

$$\boldsymbol{Throttle = 1 - \frac{kecepatan\ mobil}{kecepatan\ maximal}} \quad \textbf{(7)}$$

Gambar 3.57 Diagram alur navigasi arah kemudi mobil



Gambar 3.58 Diagram alur navigasi kecepatan mobil

3.19. Spesifikasi Rancangan Model

Rancangan ini memiliki spesifikasi sebagai berikut:

3.19.1. Konstruksi Model

Model ini memiliki total parameter sebanyak 252219. Selain itu model ini terdiri dari 5-layer konvolusi, 1 layer flatten, 3-layer terhubung dan 1 layer output. Untuk lebih detailnya digambarkan pada **Gambar 3.59.**

```
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
```

Gambar 3.59 Konstruksi rancangan model yang telah terlatih

3.19.2. Kemampuan Navigasi

Model ini telah teruji untuk menavigasi mobil dengan rentang kecepatan 10 hingga 30 (satuan kecepatan pada simulator). Model ini direkomendasi untuk menavigasi mobil yang memiliki batas kecepatan 10 hingga 20 (satuan kecepatan pada simulator).

3.20. Skenario Pengujian

Dalam melakukan pengujian, terdapat beberapa tahapan dalam scenario pengujian. Tahapan – tahapan tersebut yaitu

1. Melakukan konfigurasi bi-directional communication antar server dan client. Dimana mobil bertindak sebagai client dan program computer sebagai server.
2. Menjalankan mobil dengan kecerdasan ini sebagai otak dari sistem navigasi
3. Melakukan pengujian pada track 1 dengan kecepatan maksimal 30, 25, 20, 15, 10 dengan 3 putaran untuk tiap kecepatan
4. Mencatat hasil pengujian pada track 1
5. Melakukan pengujian pada track 2 dengan kecepatan maksimal 30, 25, 20, 15, 10 dengan 3 putaran untuk tiap kecepatan
6. Mencatat hasil pengujian pada track 2
7. Melakukan pengujian pada track 3 dengan kecepatan maksimal 30, 25, 20, 15, 10 sampai finish untuk tiap kecepatan
8. Mencatat hasil pengujian pada track 3

BAB 4
HASIL DAN ANALISA

Pada bab ini akan dijelaskan mengenai simulasi yang dilakukan, hasil simulasi, dan analisa simulasi. Kemudian, tolak ukur dari serangkaian simulasi ini adalah keberhasilan deep learning yang dirancang dalam menavigasi mobil diberbagai lintasan. Karena tujuan dari digunakannya deep learning dalam menavigasi yaitu agar mobil dapat dinavigasi dalam situasi yang belum pernah ditemui sebelumnya.

4.1. Pembelajaran Model

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 4.3**. 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.

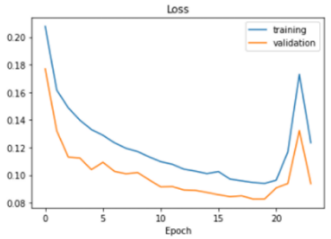
Setelah dilakukan plotting terhadap grafik loss, dilakukan evaluasi model terhadap seluruh data berupa gambar yang relative terhadap arah kemudi pada track 1 dan track 2. Hal ini dilakukan dengan pemanggilan fungsi evaluate pada model deep learning tersebut dengan inputan gambar dan arah kemudi sebagai evaluasinya. Hal ini dapat dilihat pada **Gambar 4.1.** 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 4.1 Evaluasi model mendapatkan nilai mse 0.0545

```
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 4.2 Proses pembelajaran model



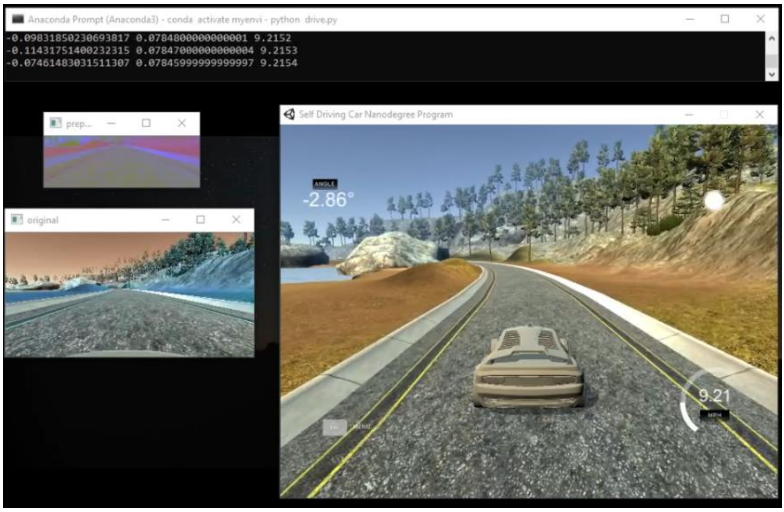
Gambar 4.3 Plot grafik loss pada rancangan model

4.2. Simulasi Model Kecerdasan Buatan pada Simulator Dalam Menavigasi Mobil

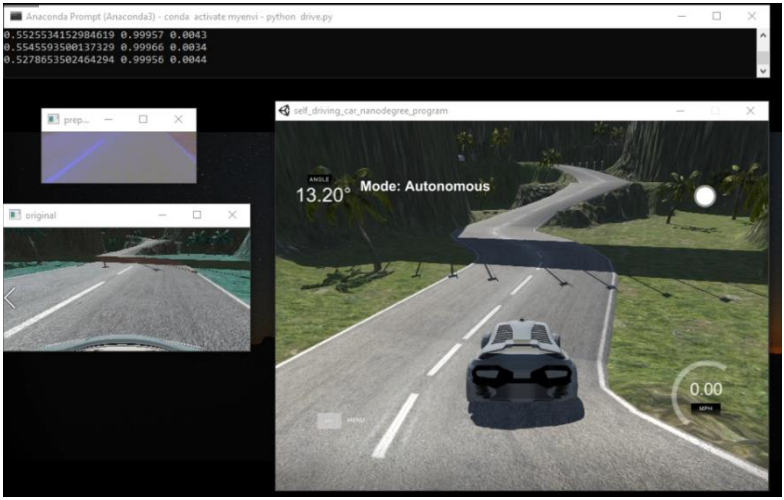
Pada evaluasi model dalam menavigasi mobil, terjabarkan pada **Tabel 4.1.** Analisa terhadap pengujian yang telah dilakukan yakni mobil dapat melaju dengan baik pada track 1 dengan kecepatan apapun karena track 1 memiliki karakteristik jalan pada track yang cenderung datar sedangkan pada track dua memiliki 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. Untuk hasil secara visual dapat dilihat pada **Gambar 4.4** sampai **Gambar 4.18.**

Tabel 4.1 Pengujian Navigasi Mobil

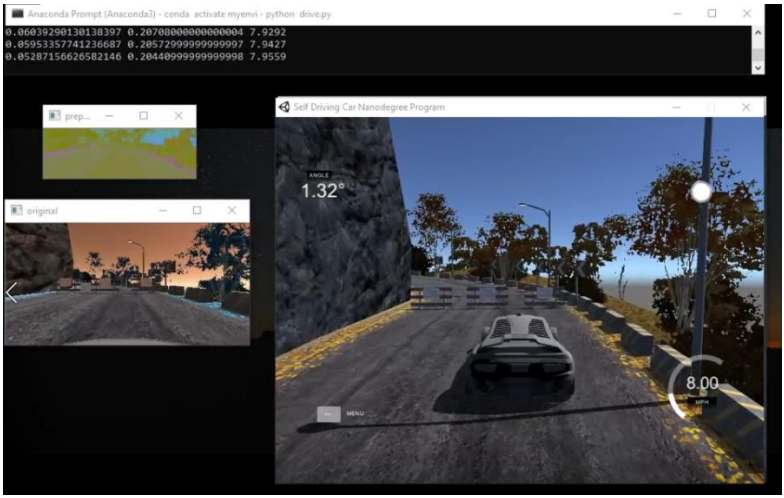
Kecepatan	Track 1	Track 2	Track 3	Keterangan
10	3 putaran berhasil	Tidak berhasil	Finish	Mobil tidak dapat bermanuver saat tanjakan pada track 2
15	3 putaran berhasil	3 putaran berhasil	Finish	
20	3 Putaran berhasil	3 Putaran berhasil	Finish	
25	3 Putaran berhasil	Tidak berhasil	Tidak berhasil	Menabrak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



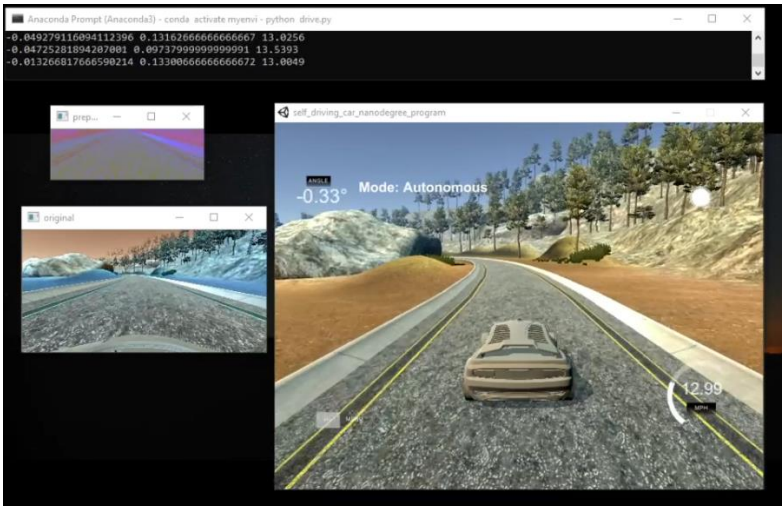
Gambar 4.4 Kecepatan 10 - Pada Track 1 - Berhasil



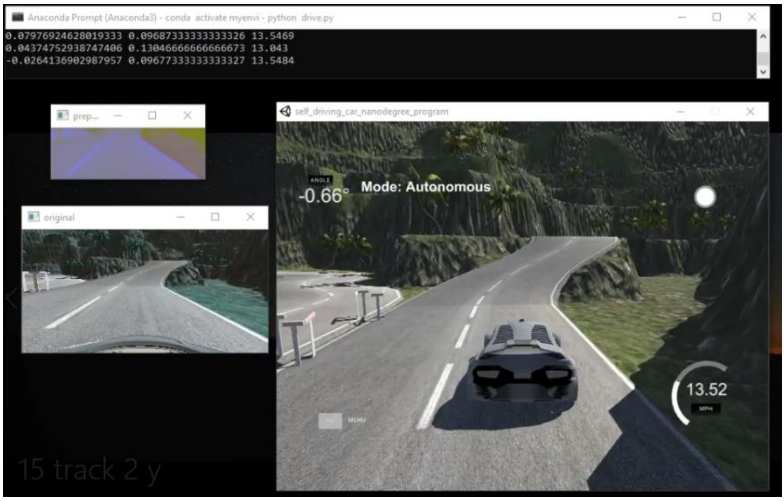
Gambar 4.5 Kecepatan 10 - Track 2 – Gagal



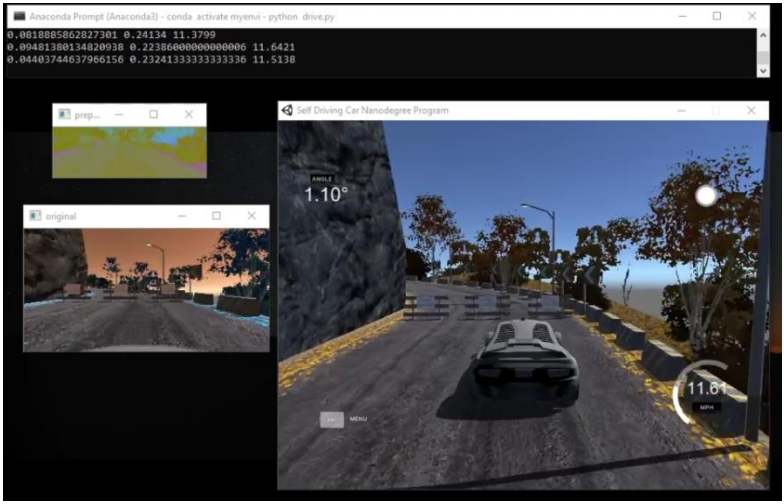
Gambar 4.6 Kecepatan 10 - Track 3 – Finish



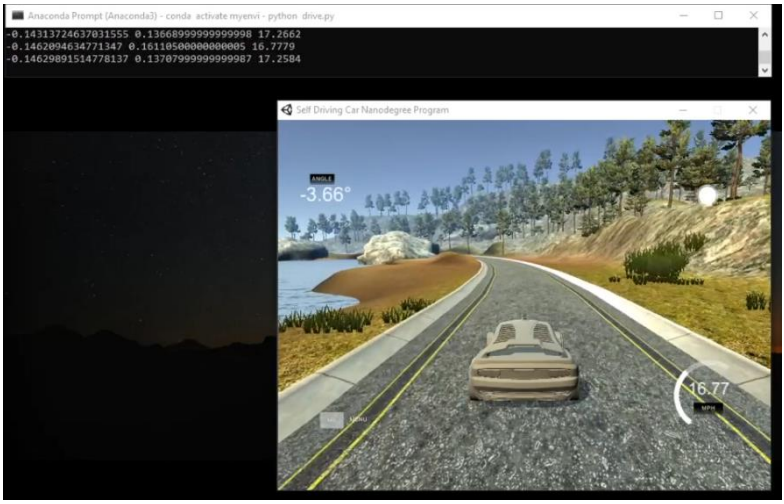
Gambar 4.7 Kecepatan 15 - Track 1 – Berhasil



Gambar 4.8 Kecepatan 15 - Track 2 – Berhasil

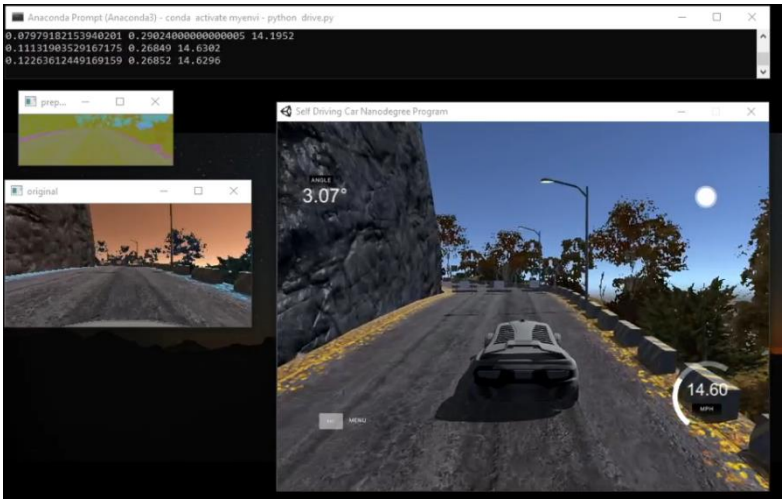


Gambar 4.9 Kecepatan 15 - Track 3 – Finish

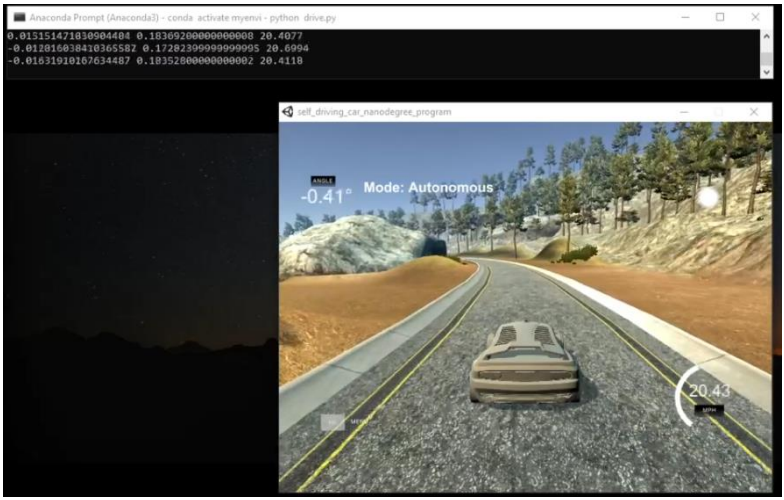


Gambar 4.10 Kecepatan 20 - Track 1 – Berhasil

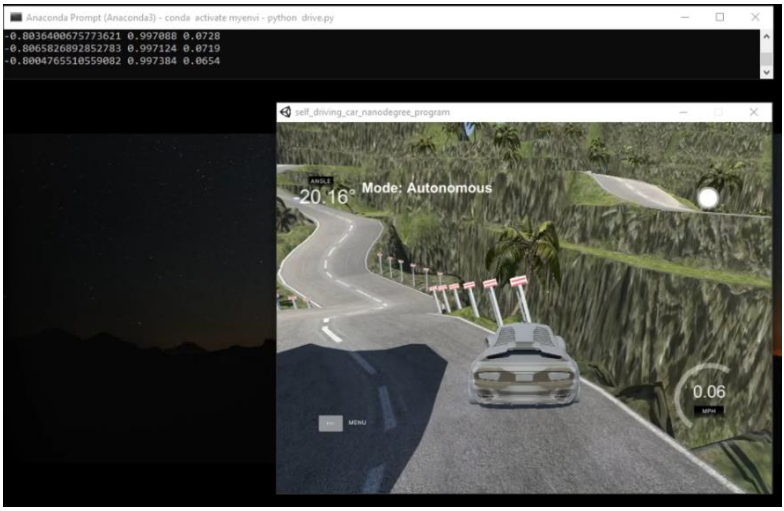
Gambar 4.11 Kecepatan 20 - Track 2 – Berhasil



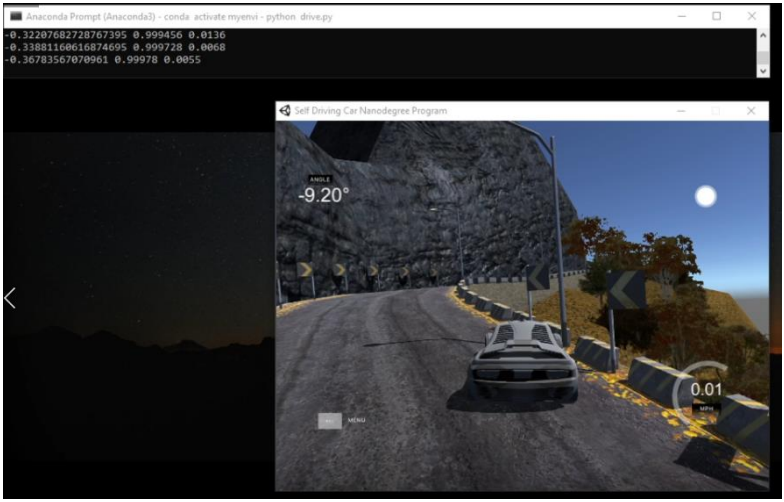
Gambar 4.12 Kecepatan 20 - Track 3 – Finish



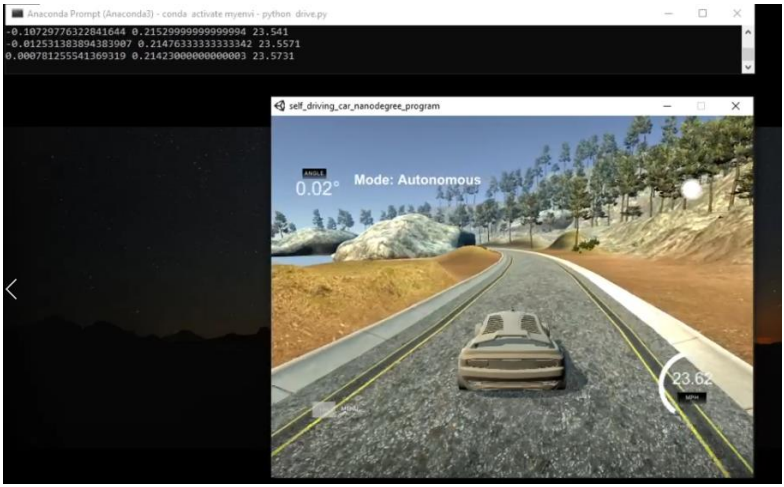
Gambar 4.13 Kecepatan 25 - Track 1 – Finish



Gambar 4.14 Kecepatan 25 - Track 2 – Gagal

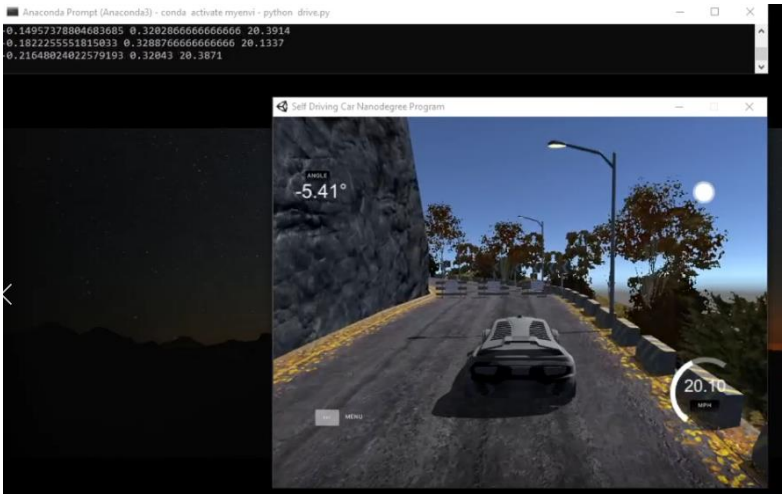


Gambar 4.15 Kecepatan 25 - Track 3 – Gagal



Gambar 4.16 Kecepatan 30 - Track 1 – Berhasil

Gambar 4.17 Kecepatan 30 - Track 2 – Gagal



Gambar 4.18 Kecepatan 30 - Track 3 - Finish

BAB 5
PENUTUP

5.1. 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

5.2. Saran

Dari serangkaian kegiatan yang telah penulis lakukan pada proses pembuatan Tugas Akhir, terdapat beberapa saran yang bisa penulis berikan sekiranya akan dilakukan penelitian lebih lanjut mengenai rancang bangun model deep learning untuk menavigasi mobil menggunakan sensor kamera. Diantaranya:

1. Melakukan / merancang model yang dapat menavigasi kecepatan mobil pada saat tanjakan maupun turunan
2. Melakukan uji coba pada simulator lain
3. Melakukan uji coba pada prototype mobil dalam dunia nyata
4. Penggabungan dengan model lain untuk melakukan navigasi saat berhadapan dengan jalan yang beruas.

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] J. Ponce, "What is a camera?," dalam *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, Jun 2009, hlm. 1526–1533. doi: 10.1109/CVPR.2009.5206668.

[4] 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.

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

[6] 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

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

[8] 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.

[9] E. J. Bernabeu, J. Tornero, dan M. Tomizuka, "A navigation system for unmanned vehicles in automated highway systems," dalam *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002, vol. 1, hlm. 696–701.

[10] S. O.-R. A. V. S. Committee, "Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems," *SAE Stand. J*, vol. 3016, hlm. 1–16, 2014.

[11] 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.

[12] J. F. Blinn, “What is a pixel?,” *IEEE Comput. Graph. Appl.*, vol. 25, no. 5, hlm. 82–87, 2005.
[13] R. Khatry dan A. Thompson, “Camera and Lidar sensor models for autonomous vehicles,” 2021.
[14] D. Forsyth dan J. Ponce, *Computer vision: A modern approach*. Prentice hall, 2011.
[15] G. H. Granlund dan H. Knutsson, *Signal processing for computer vision*. Springer Science & Business Media, 2013.
[16] 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.
[17] G. Bradski dan A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. O’Reilly Media, Inc., 2008.
[18] T. M. Mitchell, “Machine learning,” 1997.
[19] C. Donalek, “Supervised and unsupervised learning,” dalam *Astronomy Colloquia. USA*, 2011, vol. 27.
[20] B. Yegnanarayana, *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.
[21] “Jaringan Saraf Tiruan (JST).” https://www.kajianpustaka.com/2016/11/jaringan-saraf-tiruan-jst.html (diakses Jul 23, 2021).
[22] D. Svozil, V. Kvasnicka, dan J. Pospichal, “Introduction to multi-layer feed-forward neural networks,” *Chemom. Intell. Lab. Syst.*, vol. 39, no. 1, hlm. 43–62, 1997.
[23] “Fig. 4 A feedforward neural network with a single output neuron and...,” *ResearchGate*. https://www.researchgate.net/figure/A-feedforward-neural-network-with-a-single-output-neuron-and-with-one-hidden-layer_fig2_220176384 (diakses Jul 23, 2021).
[24] LaptrinhX, “RNN or Recurrent Neural Network for Noobs,” *LaptrinhX*, Jun 20, 2018. https://laptrinhx.com/rnn-or-recurrent-neural-network-for-noobs-374348076/ (diakses Jul 23, 2021).
[25] “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).
[26] N. A. Batubara dan R. M. Awangga, *TUTORIAL OBJECT DETECTION PLATE NUMBER WITH CONVOLUTION NEURAL NETWORK (CNN)*, vol. 1. Kreatif, 2020.

[27] 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.

LAMPIRAN

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

--- halaman ini sengaja dikosongkan ---

BIODATA PENULIS

I Made Guna Winangun, lahir di Denpasar, Bali, 5 Desember 1998, adalah anak kedua dari empat bersaudara. Penulis menggeluti dunia teknologi elektro dengan pengalaman lebih dari 4 tahun. Keterampilan yang dimiliki Penulis yaitu desain sistem PV, Pengembangan Web, Perangkat IoT, Visi Komputer, Pembelajaran Mesin, Pengembangan Android, Ekonomi Teknik, Kreativitas, Pemecahan Masalah, Inovatif, Kerja Sama Tim, Berpikir kritis. Penulis gemar dalam membuat proyek mini yang memecahkan masalah atau hanya untuk mempelajari sesuatu yang menarik. Penulis dapat dihubungi melalui rumahgugun@gmail.com.