

**SISTEM PARKIR CERDAS TERINTEGRASI DENGAN *INTERNET OF THINGS*  
(*IoT*) DAN PEMROSESAN CITRA DIGITAL MENGGUNAKAN METODE  
*FASTER REGION BASED CONVOLUTIONAL NEURAL NETWORK*  
*(FASTER R-CNN)* DAN *OPTICAL CHARACTER*  
*RECOGNITION (OCR)***

**SKRIPSI**

**ALBERT LUKAS TALUPAN PANGARIBUAN**

**201402150**



**PROGRAM STUDI S1 TEKNOLOGI INFORMASI  
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI  
UNIVERSITAS SUMATERA UTARA  
MEDAN  
2024**

**SISTEM PARKIR CERDAS TERINTEGRASI DENGAN *INTERNET OF THINGS*  
(*IoT*) DAN PEMROSESAN CITRA DIGITAL MENGGUNAKAN METODE  
*FASTER REGION BASED CONVOLUTIONAL NEURAL NETWORK*  
*(FASTER R-CNN)* DAN *OPTICAL CHARACTER*  
*RECOGNITION (OCR)***

**SKRIPSI**

**Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh izajah Sarjana  
Teknologi Informasi**

**ALBERT LUKAS TALUPAN PANGARIBUAN  
201402150**



**PROGRAM STUDI S1 TEKNOLOGI INFORMASI  
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI  
UNIVERSITAS SUMATERA UTARA  
MEDAN  
2024**

## PERSETUJUAN

Judul : SISTEM PARKIR CERDAS TERINTEGRASI DENGAN *INTERNET OF THINGS (IoT)* DAN PEMROSESAN CITRA DIGITAL MENGGUNAKAN METODE *FASTER REGION BASED CONVOLUTIONAL NEURAL NETWORK (FASTER R-CNN)* DAN *OPTICAL CHARACTER RECOGNITION (OCR)*

Kategori : SKRIPSI

Nama Mahasiswa : ALBERT LUKAS TALUPAN PANGARIBUAN

Nomor Induk Mahasiswa : 201402150

Program Studi : SARJANA (S1) TEKNOLOGI INFORMASI

Fakultas : ILMU KOMPUTER DAN TEKNOLOGI INFORMASI  
UNIVERSITAS SUMATERA UTARA

Medan, 14 Juni 2024

Komisi Pembimbing:

Pembimbing 2,

Dr. Erna Budhiarti Nababan, M.I.T  
NIP. 196210262017042001

Pembimbing 1,

Fahrurrozi Lubis, B.I.T., M.Sc.IT  
NIP. 198610122018052001

Diketahui/disetujui oleh:  
Program Studi S1 Teknologi Informasi  
Ketua,

  
Dedy Arisandi, S.T., M.Kom  
NIP. 197908122009121002

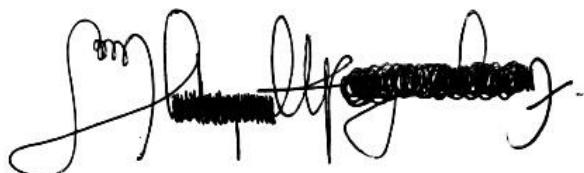
**PERNYATAAN**

SISTEM PARKIR CERDAS TERINTEGRASI DENGAN *INTERNET OF THINGS*  
(*IoT*) DAN PEMROSESAN CITRA DIGITAL MENGGUNAKAN METODE  
*FASTER REGION BASED CONVOLUTIONAL NEURAL NETWORK*  
(*FASTER R-CNN*) DAN *OPTICAL CHARACTER*  
*RECOGNITION (OCR)*

**SKRIPSI**

Saya mengakui bahwa skripsi ini merupakan hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan sumbernya.

Medan, 14 Juni 2024



ALBERT LUKAS TALUPAN PANGARIBUAN  
NIM. 201402150

## **UCAPAN TERIMA KASIH**

Segala puji syukur penulis panjatkan kepada Tuhan Yesus Kristus atas berkat dan karunia-Nya penulis dapat menyelesaikan penyusunan skripsi yang berjudul “Sistem Parkir Cerdas Terintegrasi Dengan *Internet Of Things (IoT)* Dan Pemrosesan Citra Digital Menggunakan Metode *Faster Region Based Convolutional Neural Network (Faster R-CNN)* Dan *Optical Character Recognition (OCR)*”, yang merupakan salah satu syarat untuk memperoleh gelar Sarjana pada Program Studi S1 Teknologi Informasi, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara.

Penulis menyampaikan rasa terima kasih, yang sebesar-besarnya atas dukungan luar biasa secara moral dan materi dari keluarga penulis Papi Renhard Pangaribuan, S.E, dan Mami Dra. Dormian Anita Simanungkalit, abang Samuel Tulus Pangaribuan, S.T dan kakak Paulina Theresia Pangaribuan, A.P.A.Pj. Rasa terimakasih dan bersyukur penulis ucapan kepada seluruh pihak yang turut serta terlibat dalam masa perkuliahan sampai masa penggerjaan skripsi ini:

1. Bapak Prof. Dr. Muryanto Amin, S.Sos., M.Si., selaku Rektor Universitas Sumatera Utara.
2. Ibu Dr. Maya Silvi Lydia, B.Sc., M.Sc., selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
3. Bapak Dedy Arisandi, S.T., M.Kom., selaku Ketua Program Studi S1 Teknologi Informasi Universitas Sumatera Utara.
4. Bapak Fahrurrozi Lubis, B.IT., M.Sc.IT., selaku Dosen Pembimbing 1 dan ibu Dr. Erna Budhiarti Nababan, M.IT., selaku Dosen Pembimbing 2 yang telah bersedia meluangkan waktu dan memberikan ilmu kepada penulis selama proses perkuliahan hingga penyusunan skripsi hingga selesai ke depannya.
5. Bapak Ainul Hizriadi, S.Kom., M.Sc., selaku Ketua Penguji dan bapak Seniman, S. Kom., M. Kom., selaku Dosen Penguji 2, yang turut bersedia meluangkan waktu dan memberikan ilmu kepada penulis dalam berkuliah dan penyusunan skripsi.

6. Para dosen pengajar Studi S1 Program Studi Teknologi Informasi Universitas Sumatera Utara yang telah memberikan ilmu serta dukungan selama proses perkuliahan.
7. Staf akademik yang telah memberikan fasilitas segala aspek yang dapat menunjang proses perkuliahan.
8. Opung Nyonya Simanungkalit boru Simanjuntak, Sepupu Moses Simanungkalit, S.Kom., keluarga besar Pangaribuan dan keluarga besar Simanungkalit yang juga turut mendoakan serta mendukung penulis.
9. Tulang Roling Simanungkalit Manager Divisi Telkom GMP Gatot Subroto, Pak Daniel Sihombing, S.T. selaku SM General Affair Divisi IT, pak Agung Cahyono Raharjo selaku *Manager Secretariate DIT*, ibu Neni Adiningsih selaku *PIC DIT*, dan Mas Jafar selaku mentor atas kesempatan emas dan ilmu selama magang di kantor Telkom STO Gambir, Jakarta.
10. Teman-teman Magang Telkom STO Gambir, Asad dan Azriel yang telah memberikan dedikasinya sewaktu menimba ilmu di Telkom.
11. Bapak Kepala Dinas Sosial Khoiruddin, bapak kepala bidang PFM Ronald Sihotang serta seluruh pegawai dinas sosial yang telah memberikan pelajaran yang baik bagi penulis.
12. Teman-teman Magang Dinas Sosial Kota Medan Divisi IT, Ferdinand Tumanggor, Fajar Rivaldi, Ramadhan, Tarmizi, Zacky, Arya, Ibnun, Yazid yang telah banyak membuka wawasan tentang ilmu IT.
13. Bapak Jamotan Siallagan selaku panitia acara sekaligus manajer divisi dan tim di PT. Inalum Porsea yang telah gerak cepat dan memberikan ruangan yang baik untuk sewaktu penulis melaksanakan Seminar Proposal.
14. Chindy Margaretta Sianipar yang telah membantu selama perkuliahan juga proses skripsi.
15. Teman-teman seperjuangan angkatan 20, Jethro Sihotang, Yeftha Pardede, Felix Rumahorbo, Kevin Bangun, Vicky Sinaga, Wahyu Marpaung, Rio Siregar, Yericho Natanael, Irwansyah Sarumaha yang telah bersinergi dalam masa perkuliahan.
16. Bora Sejati Siboro, Deardo Purba, Christian Tampubolon, Enrich Manalu, Sonofbel Silalahi, Juan Butarbutar, Idel Sibarani, Rachel Batubara, Katheryn Sianipar yang telah menemani dalam musik dan pertemanan yang sehat.

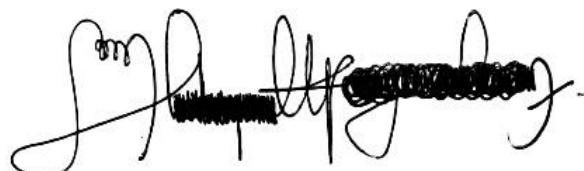
17. Teman SMA, Farhan Batubara, Alex Sembiring, Gabriel Marbun, Arief Fadhlhan, Gugun Damanik, Philip Sidabukke yang selalu mendukung penulis.
18. Gabriel Binsar Sinaga yang selalu siap dalam panggilan bantuan dan mendukung langkah-langkah penulis.
19. Bang Geylfedra Matthew Panggabean, Aulia Rahman Partomuan Sihite, M. Zikri Ihsan, Christoper Miando, Sarah Sinurat, Vania Siahaan yang telah berkontribusi terhadap penulis.
20. Teman-teman waikiki, Yehezkiel Panggabean, Renal, Jordan Simanjuntak, Prentino Barus, Christoper Simbolon yang juga turut memberikan bantuan selama perkuliahan dan pengembangan skripsi.
21. Teman-teman angkatan 20 yang tidak dapat disebutkan semua, senior angkatan 18 dan 19 tidak dapat disebutkan semua, yang turut memberikan peluang-peluang untuk penulis dapat beraktivitas positif di lingkungan kampus.

Penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan, namun penulis tetap bersinergi dengan orang-orang yang memberikan saran dan kritik untuk dapat menambah kualitas ilmu yang bersifat membangun.

Terimakasih.

Medan, 31 Mei 2024

Penulis



ALBERT LUKAS TALUPAN PANGARIBUAN  
NIM. 201402150

## ABSTRAK

Parkir merupakan bagian tak terpisahkan dari kehidupan berkendara, terutama di lingkungan perkotaan. Namun, sistem parkir yang tidak efisien menyebabkan pengendara sering menghabiskan waktu lama hanya untuk mencari tempat parkir, bahkan menghabiskan waktu 10 jam per bulan di kota besar seperti Jakarta. Selain membuang waktu, pencarian parkir yang lama juga meningkatkan konsumsi bahan bakar dan emisi  $CO_2$ . Penelitian ini bertujuan untuk mengembangkan aplikasi parkir cerdas terintegrasi dengan teknologi *Internet of things* dan *Computer Vision* untuk mengatasi masalah tersebut.

Sistem ini menggunakan sensor ultrasonik yang terhubung dengan mikrokontroler *ESP32* untuk mendeteksi ketersediaan slot parkir dengan waktu nyata, yang kemudian mengirimkan data perubahan ke *Firebase*. Selain itu, teknologi *Faster Region-Based Convolutional Neural Network (Faster R-CNN)* dan *EasyOCR* digunakan untuk mendeteksi dan mengidentifikasi plat nomor kendaraan dan kertas penanda slot parkir, memastikan kesesuaian antara pemesanan dan penggunaan aktual slot parkir.

Hasil penelitian menunjukkan bahwa sistem sensor memiliki waktu tercepat dalam pengukuran 106.3 milisekon dan pengiriman data 73.6 milisekon. Model *Faster R-CNN* menunjukkan akurasi sebesar 86% untuk deteksi plat nomor polisi dan kertas penanda slot parkir, sementara *EasyOCR* mencapai akurasi 100% dalam mengenali karakter plat nomor dan kertas penanda slot parkir. Total waktu inferensi dan ekstraksi karakter rata-rata adalah 2.7 detik.

Penelitian yang mengusung konsep integrasi ini sangat menjadi solusi bagi pengguna dan pengusaha vendor parkir.

Kata Kunci: Parkir Cerdas, *Internet of Things*, *ESP32*, Sensor Ultrasonik, *Faster R-CNN*, *EasyOCR*

**INTEGRATED SMART PARKING SYSTEM USING INTERNET OF THINGS  
(IOT) AND DIGITAL IMAGE PROCESSING WITH FASTER REGION  
BASED CONVOLUTIONAL NEURAL NETWORK (FASTER R-CNN)  
AND OPTICAL CHARACTER RECOGNITION (OCR)**

**ABSTRACT**

*Parking is an integral part of driving, especially in urban areas. However, inefficient parking systems often cause drivers to spend a significant amount of time just to find a parking spot, even up to 10 hours per month in major cities like Jakarta. Besides wasting time, prolonged parking searches also increase fuel consumption and CO<sub>2</sub> emissions. This study aims to develop a smart parking application integrated with Internet of Things and Computer Vision.*

*The system uses ultrasonic sensors connected to an ESP32 microcontroller to detect real-time parking slot availability, sending status updates to Firebase. Additionally, Faster Region-Based Convolutional Neural Network (Faster R-CNN) and EasyOCR technologies are used to detect and identify vehicle license plates and parking slot markers, ensuring consistency between reservations and actual parking slot usage.*

*The research findings indicate that the sensor system has the fastest measurement time of 106.3 milliseconds and data transmission time of 73.6 milliseconds. The Faster R-CNN model shows an accuracy of 86% for detecting license plates and parking slot markers, while EasyOCR achieves 100% accuracy in recognizing the characters of license plates and parking slot markers. The average total time for inference and character extraction is 2.7 seconds.*

*This integrated approach provides a significant solution for both users and parking vendors.*

*Keywords:* Smart Parking, Internet of Things, ESP32, Ultrasonic Sensor, Faster R-CNN, EasyOCR

**DAFTAR ISI**

PERSETUJUAN .....	iii
PERNYATAAN .....	iv
UCAPAN TERIMA KASIH.....	v
ABSTRAK.....	viii
ABSTRACT .....	ix
DAFTAR ISI.....	x
DAFTAR TABEL.....	xiii
DAFTAR GAMBAR .....	xiv
BAB 1 PENDAHULUAN .....	1
1.1. Latar Belakang .....	1
1.2. Rumusan Masalah .....	4
1.3. Tujuan Penelitian.....	4
1.4. Batasan Masalah.....	5
1.5. Manfaat Penelitian.....	5
1.6. Metodologi Penelitian .....	5
1.7. Sistematika Penulisan.....	6
BAB 2 LANDASAN TEORI.....	8
2.1. Parkir .....	8
2.2. Aplikasi Mobile.....	8
2.3. <i>Internet of Things</i> .....	8
2.3.1. Sensor pendekripsi: sensor ultrasonik HC-SR04 .....	9
2.3.2. Mikrokontroler: ESP32.....	9
2.3.3. Kabel jumper .....	10
2.3.4. Breadboard.....	11
2.3.5. Power supply .....	11
2.3.6. Arduino IDE .....	12
2.4. <i>Deep Learning</i> .....	12
2.5. <i>Computer Vision</i> .....	13
2.5.1. Object detection .....	14
2.5.2. Convolutional Neural Network (CNN) .....	14

2.5.3. Region-based Convolutional Neural Network (R-CNN) .....	15
2.5.4. Fast Region Convolutional Neural Network (Fast R-CNN).....	15
2.5.5. Faster Region Convolutional Neural Network (Faster R-CNN) .....	16
2.5.6. Perbedaan CNN, R-CNN, Fast R-CNN dan Faster R-CNN.....	17
2.5.7. PyTorch.....	18
2.5.8. Confusion Matrix.....	19
2.6. Penelitian Terdahulu .....	20
BAB 3 ANALISIS DAN PERANCANGAN SISTEM.....	27
3.1. Pengumpulan Data .....	27
3.1.1. Custom data kertas penanda slot parkir .....	27
3.1.2. Data plat nomor polisi .....	28
3.1.3. Akumulasi data citra .....	28
3.2. Arsitektur Umum.....	28
3.2.1. Arsitektur Internet of Things (IoT).....	30
3.2.2. Image processing .....	30
3.2.3. Optical character recognition.....	30
3.3. Perancangan <i>Image Processing</i> : Plat Nomor Polisi dan Kertas Penanda Slot Parkir .....	30
3.3.1. Image acquisition.....	30
3.3.2. Annotation .....	31
3.3.3. Preprocessing .....	31
3.3.4. Augmentasi .....	32
3.3.5. Eksport gambar dan anotasi.....	35
3.3.6. Feature extraction .....	35
3.3.7. Trained model.....	41
3.3.8. Output inference .....	41
3.4. Perancangan <i>Easy-Optical Character Recognition (EasyOCR)</i> .....	41
3.4.1. Pre-Processing: EasyOCR .....	42
3.4.2. Output .....	42
3.5. Perancangan Perangkat Keras .....	43
3.6. Perancangan Aplikasi Android.....	43
3.6.1. Sisi klien (frontend) .....	43
3.6.2. Sisi server (backend).....	47
BAB 4 IMPLEMENTASI DAN PENGUJIAN SISTEM.....	48

4.1. Implementasi Sistem .....	48
4.2. Implementasi Perancangan Aplikasi .....	48
4.2.1. Sisi klien (Frontend) .....	48
4.2.2. Sisi server (Backend).....	56
4.3. Implementasi Perangkat Keras .....	58
4.3.1. Rangkaian alat pendeteksi mobil pada setiap slot .....	58
4.3.2. Rangkaian penginstalan paket ESP32 .....	58
4.3.3. Rangkaian code program 3 unit perangkat keras.....	60
4.3.4. Realtime database .....	64
4.3.5. Pengujian sistem .....	64
4.4. Implementasi Pemrosesan Citra Digital .....	68
4.4.1. Implementasi data pada pemrosesan citra .....	68
4.4.2. Pelatihan sistem .....	69
4.4.3. Pengujian sistem .....	70
4.5. Implementasi <i>EasyOCR</i> .....	76
4.5.1. Implementasi data.....	76
4.5.2. Pengujian sistem .....	76
4.6. Prosedur Operasional .....	78
4.6.1. Persiapan perangkat keras dan lokasi .....	78
4.6.2. Persiapan server .....	79
4.6.3. Alur aplikasi: mobil tanpa aplikasi .....	81
4.6.4. Alur aplikasi: mobil dengan aplikasi .....	81
BAB 5 KESIMPULAN DAN SARAN .....	87
5.1. Kesimpulan.....	87
5.2. Saran.....	88
DAFTAR PUSTAKA .....	89

**DAFTAR TABEL**

Tabel 2.1 Penelitian terdahulu .....	23
Tabel 3.1 Data Gambar .....	28
Tabel 3.2 Jumlah Gambar .....	35
Tabel 3.3 Daftar Kebutuhan Perangkat Keras .....	43
Tabel 3.4 Konfigurasi Pin Perangkat Keras.....	43
Tabel 4.1 Data Pengujian Waktu Pengukuran dan Pengiriman Data .....	66
Tabel 4.2 Data Ringkasan Pengukuran dan Pengiriman Data .....	67
Tabel 4.3 Daftar Hasil Riwayat Pelatihan Model .....	69
Tabel 4.4 Data Hasil Uji Model .....	70
Tabel 4.5 Nilai <i>TP</i> , <i>TN</i> , <i>FP</i> , dan <i>FN</i> .....	73
Tabel 4.6 Nilai <i>Precision</i> , <i>Recall</i> dan <i>F1 Score</i> .....	75
Tabel 4.7 Hasil Uji <i>EasyOCR</i> Ekstrak Karakter Plat Nomor Polisi .....	76
Tabel 4.8 Hasil Uji <i>EasyOCR</i> Ekstrak Karakter Kertas Penanda Slot Parkir.....	77
Tabel 4.9 Hasil Pengujian Waktu Inferensi Gambar dan Ekstrak Karakter .....	78

## DAFTAR GAMBAR

Gambar 2.1 Sensor Ultrasonik HC-SR04 .....	9
Gambar 2.2 <i>ESP32</i> .....	10
Gambar 2.3 <i>Breadboard</i> .....	11
Gambar 2.4 Perbedaan <i>Machine Learning</i> dan <i>Deep Learning</i> .....	12
Gambar 2.5 Perbedaan Visualisasi Manusia dan Komputer Terhadap Citra Digital ..	13
Gambar 2.6 Arsitektur <i>CNN</i> .....	15
Gambar 2.7 Arsitektur <i>R-CNN</i> .....	15
Gambar 2.8 Arsitektur <i>Fast R-CNN</i> .....	16
Gambar 2.9 Arsitektur <i>Faster R-CNN</i> .....	17
Gambar 3.1 <i>Custom</i> pada <i>Canva</i> Kertas Penanda Parkir.....	27
Gambar 3.2 Arsitektur Umum .....	29
Gambar 3.3 Anotasi pada Roboflow.....	31
Gambar 3.4 <i>Resize</i> .....	32
Gambar 3.5 <i>Grayscale</i> .....	32
Gambar 3.6 <i>Brightness</i> .....	33
Gambar 3.7 <i>Exposure</i> .....	33
Gambar 3.8 <i>Blur</i> .....	34
Gambar 3.9 <i>Saturation</i> .....	34
Gambar 3.10 <i>Rotation</i> .....	35
Gambar 3.11 Matriks <i>Input</i> .....	36
Gambar 3.12 Perhitungan Matriks <i>Input</i> dengan Matriks <i>Filter</i> .....	36
Gambar 3.13 Perhitungan Pergeseran Matriks .....	37
Gambar 3.14 Matriks <i>Convolution layer</i> .....	38
Gambar 3.15 Matriks <i>Feature map</i> .....	39
Gambar 3.16 <i>Feature Map</i> .....	39
Gambar 3.17 <i>Region Proposal Network</i> .....	40
Gambar 3.18 Target <i>Region of Interest Pooling</i> .....	40
Gambar 3.19 Tahapan <i>RoI Pooling</i> .....	41
Gambar 3.20 <i>RoI Pooling</i> .....	41

Gambar 3.21 <i>Preprocess</i> Gambat Slot Sign.....	42
Gambar 3.22 <i>Preprocess</i> Gambar Plat Mobil.....	42
Gambar 3.23 Konfigurasi Perangkat Keras .....	43
Gambar 3.24 Perancangan Visualisasi Slot Parkir .....	44
Gambar 3.25 Perancangan Halaman beranda .....	44
Gambar 3.26 Perancangan Halaman Riwayat.....	45
Gambar 3.27 Perancangan Halaman <i>Input</i> Plat Nomor Polisi.....	45
Gambar 3.28 Perancangan Halaman Unggah Foto .....	46
Gambar 3.29 Perancangan Halaman Hasil .....	46
Gambar 3.30 <i>Entity Relation Diagram</i> .....	47
Gambar 4.1 <i>Splash Screen</i> .....	49
Gambar 4.2 Halaman <i>Home</i> .....	49
Gambar 4.3 Halaman Petunjuk Aplikasi .....	50
Gambar 4.4 Halaman Visualisasi Slot Parkir .....	50
Gambar 4.5 Halaman Riwayat .....	51
Gambar 4.6 Halaman <i>Input</i> Plat.....	51
Gambar 4.7 Halaman <i>Upload</i> Gambar .....	52
Gambar 4.8 Halaman Hasil Inferensi dan Hasil Karakter .....	52
Gambar 4.9 Pesan Berhasil .....	53
Gambar 4.10 Pesan Kesalahan Slot Tidak Terdeteksi .....	53
Gambar 4.11 Pesan Kesalahan Slot Tidak Sesuai.....	53
Gambar 4.12 Pesan Kesalahan Plat Tidak Terdeteksi .....	54
Gambar 4.13 Pesan Kesalahan Plat Tidak sesuai .....	54
Gambar 4.14 Pesan Kesalahan Lainnya.....	54
Gambar 4.15 Halaman Animasi <i>Loading</i> .....	55
Gambar 4.16 Pesan Konfirmasi .....	55
Gambar 4.17 Firebase: Daftar Riwayat.....	56
Gambar 4.18 Firebase: Data Detail Riwayat .....	57
Gambar 4.19 Firebase: Plat berdasarkan id .....	57
Gambar 4.20 Status Slot .....	57
Gambar 4.21 Rangkaian Perangkat Keras Setiap Slot.....	58
Gambar 4.22 Preferences <i>Arduino IDE</i> .....	58
Gambar 4.23 Preferences Tambah <i>Link</i> .....	59

Gambar 4.24 <i>Board Manager ESP32</i> .....	59
Gambar 4.25 Pemilihan Detail <i>ESP32 Dev Module</i> .....	59
Gambar 4.26 Penggunaan Paket <i>ESP32</i> .....	60
Gambar 4.27 Inisialisasi <i>Library ESP32</i> .....	60
Gambar 4.28 Inisialisasi Pin <i>ESP32</i> .....	60
Gambar 4.29 Fungsi <i>Setup Pins</i> .....	60
Gambar 4.30 Kredensial Firebase .....	61
Gambar 4.31 Kredensial <i>Wi-Fi</i> .....	61
Gambar 4.32 Inisialisasi Fungsi <i>Firebase</i> .....	61
Gambar 4.33 Konfigurasi Koneksi ke <i>Wi-Fi</i> .....	61
Gambar 4.34 Konfigurasi <i>LED Control</i> .....	61
Gambar 4.35 Konfigurasi <i>Firebase</i> .....	62
Gambar 4.36 Konfigurasi Fungsi Pengukuran Sensor.....	62
Gambar 4.37 Konfigurasi Pengiriman Data Sensor 1.....	63
Gambar 4.38 Konfigurasi Pengiriman Data Sensor 2.....	63
Gambar 4.39 Konfigurasi Pengiriman Data Sensor 3.....	63
Gambar 4.40 Memanggil Semua Fungsi untuk Setup .....	64
Gambar 4.41 Fungsi Perulangan Kerja untuk <i>ESP32</i> .....	64
Gambar 4.42 Data pada Firebase Data Sensor-sensor .....	64
Gambar 4.43 <i>Code Pengujian Waktu Sensor Membaca Jarak</i> .....	65
Gambar 4.44 <i>Code Pengujian Waktu Pengiriman Data ke Firebase</i> .....	65
Gambar 4.45 Rumus Perhitungan Waktu Pengujian .....	65
Gambar 4.46 Grafik Waktu Pengukuran Sensor-Sensor .....	65
Gambar 4.47 Grafik Waktu Pengiriman Data ke <i>Firebase</i> .....	66
Gambar 4.48 Data Plat Nomor Polisi .....	68
Gambar 4.49 Data Kertas Penanda Slot Parkir.....	69
Gambar 4.50 Grafik Pelatihan .....	70
Gambar 4.51 <i>Confusion Matrix</i> .....	74
Gambar 4.52 Data <i>Preprocess EasyOCR</i> .....	76
Gambar 4.53 Persiapan Operasional.....	79
Gambar 4.54 Server Flask.....	79
Gambar 4.55 Format Pengiriman Data <i>Server Flask JSON</i> .....	79
Gambar 4.56 Contoh Hasil <i>JSON</i> .....	80

Gambar 4.57 Server <i>Frontend</i> .....	80
Gambar 4.58 Server <i>Backend</i> .....	80
Gambar 4.59 Konfigurasi <i>IP Address</i> dan <i>Port Backend</i> pada <i>Frontend</i> .....	80
Gambar 4.60 Konfigurasi <i>IP Address</i> dan <i>Port</i> pada <i>Android</i> .....	81
Gambar 4.61 Mobil Tanpa Aplikasi <i>Unavailable</i> .....	81
Gambar 4.62 Pesan Slot Untuk Parkir .....	82
Gambar 4.63 Pengguna Mengunggah Identitas Plat Mobil .....	82
Gambar 4.64 <i>Banner</i> untuk <i>Upload</i> .....	83
Gambar 4.65 Pengguna Mengunggah Foto Bukti Parkir .....	83
Gambar 4.66 Terkonfirmasi Kecocokan Data .....	84
Gambar 4.67 Halaman Hasil Inferensi dan Ekstraksi .....	85
Gambar 4.68 Pesan Konfirmasi Selesai Parkir .....	85
Gambar 4.69 Riwayat Pemarkiran .....	86

## **BAB 1**

### **PENDAHULUAN**

#### **1.1. Latar Belakang**

Parkir merupakan suatu layanan yang tidak bisa lepas dari kehidupan berkendara. Parkir umumnya disediakan oleh tempat pusat perbelanjaan, wisata, perkantoran, maupun pinggir jalan. Sistem parkir dalam lingkungan perkotaan menjadi tantangan utama yang dihadapi oleh pengendara. Masalah ini semakin kompleks karena pertumbuhan jumlah kendaraan yang tidak sebanding dengan lahan parkir yang tersedia terlebih pada hari tertentu yang melibatkan aktivitas penggunaan gedung untuk memarkirkan kendaraanya dapat membutuhkan waktu yang sangat lama. Sering ditemukan pengunjung yang berencana untuk memarkirkan kendaraannya dapat menghabiskan waktu yang tidak sebentar hanya untuk sekedar mencari parkir. Bahkan lebih parahnya sudah mengakses pintu masuk parkir langsung keluar karena sulitnya mencari slot parkir.

Berdasarkan survei mobil merk ternama, Ford, di Amerika menunjukkan bahwa pengemudi di kota besar seperti New York menghabiskan 107 jam, Los Angeles menghabiskan 85 jam dan San Fransisco menghabiskan 83 jam dalam setahun hanya untuk mencari tempat parkir. Dengan akumulasi per bulan, setiap pengemudi mobil dapat menghabiskan rata-rata 7 jam hanya untuk mencari slot parkir mobil (Nurdin, 2018). Untuk Jakarta yang merupakan kota besar di Indonesia sendiri menembus waktu lebih dari 10 jam dalam sebulan. Hal itu berdasarkan survei pada 1000 responden berusia 18-65 tahun taksi online Uber yang digelar pada Juli hingga Agustus 2017 (Pribadi & Mali, 2017).

Selain dari aspek kewaktuan, pengendara juga menjalankan mobilnya untuk mencari slot parkir yang mana menggunakan bahan bakar minyak dan dari hal tersebut mengeluarkan gas emisi  $CO_2$  yang berbahaya. Berdasarkan penelitian (Sudarti *et al.*, 2022), rata-rata mobil berpotensi mengeluarkan emisi  $CO_2$  sebesar 63,335.30 (g.30

menit-1 km-1) termasuk didalamnya mobil yang bermesin bensin dan diesel. Kondisi tersebut sangat miris bila dihitung dengan survei kewaktuan mencari parkir pada pembahasan sebelumnya.

Berdasarkan hasil survei yang telah dilakukan, terbukti bahwa kesulitan dalam menemukan tempat parkir dapat menyebabkan sejumlah masalah, termasuk keterlambatan, pemborosan waktu dan peningkatan emisi karbon dioksida. Untuk mengatasi tantangan ini, dibutuhkan solusi untuk pemerintah, pengusaha pengelola parkir dan pengendara. Penulis akan membuat suatu aplikasi terintegrasi yang dapat memberikan informasi secara *real-time* dan pendekripsi untuk kendaraan terparkir sesuai dengan yang telah dipesan pada aplikasi. Solusi ini diharapkan dapat memecahkan masalah tersebut sehingga terciptanya keefisienan dalam hal waktu yang sangat berguna untuk pengendara dan menekan pengeluaran bahan bakar serta gas emisi yang merupakan program pemerintah dan memudahkan pihak pengusaha dalam pengelola parkir dan pada penerapannya tidak merugikan pihak pengusaha pengelola parkir karena sudah terintegrasi dengan teknologi terkini.

Sebelumnya terdapat sejumlah penelitian yang telah dilakukan dalam mencari ruang parkir mobil, diantaranya penelitian Hendy dan Zuly pada tahun 2023, membuat prototipe mencari slot parkir yang mana sistem informasi parkir ditampilkan melalui sebuah layar *LCD* dengan sistem mikrokontroler menggunakan Arduino Uno (Laksono & Budiarso, 2023). Penelitian selanjutnya dari Pratama, Wicaksono dan Pramuditha membahas pengembangan pelayanan parkir di kota Surabaya dengan menggunakan Sensor Ultrasonik dan *infrared*, yang mana data dari sensor dikirim melalui protokol *Message Queuing Telemetry Transport (MQTT)* dan ditampilkan secara *real-time* ke website dengan waktu rata-rata sekitar 6.3 detik (Pratama *et al.*, 2023). Selanjutnya, terdapat penelitian prototipe sistem parkir yang menggunakan *LCD* dan program aplikasi *Blynk* sebagai sarana informasi yang menyediakan konektivitas. Penelitian tersebut menggunakan perangkat sensor Ultrasonik, menggunakan Arduino Mega sebagai mikrokontroler dan *ESP8266* sebagai *wifi module* (Rudi *et al.*, 2017)

Sehubungan dengan aplikasi terintegrasi yang dimaksud dan akan dikembangkan oleh penulis, yaitu pengenalan citra digital yang dapat rekognisi tulisan plat nomor polisi format Indonesia dan teks penanda slot parkir, terdapat juga sejumlah penelitian dari Ivany, Harfin, dan Sepriyan mengenai pendekripsi plat nomor kendaraan menggunakan metode *template matching* hanya dengan rata-rata tingkat

keberhasilan aplikasinya sebesar 70% dengan *pre-processing* yang dilakukan yaitu *grayscale*, binerisasi dan segmentasi (Sarieff *et al.*, 2019). Kemudian terdapat penelitian (Tirtana *et al.*, 2021) mendeteksi plat nomor kendaraan bermotor menggunakan *Raspberry Pi* sebagai pengambil gambar dan yang mendeteksi plat nomor menggunakan metode *YOLO* dan *Tesseract-OCR*, menghasilkan tingkat akurasi pembacaan karakter mencapai 71,46% dan rata-rata membutuhkan waktu selama 11,55 detik. Selanjutnya, penelitian (Galahartlambang *et al.*, 2023) yang dilakukan menggunakan metode *Convolutional Neural Network* dan *Optical Character Recognition*, dengan preprocessing normalisasi memberikan hasil 98% untuk deteksi area plat nomor kendaraan, 88% untuk pembacaan karakter plat nomor kendaraan.

Berdasarkan uraian diatas, maka dalam penelitian ini dirancang sebuah program digital terintegrasi, dimana aplikasi *smart parking* ini mencakup dua bidang ilmu yakni *Internet of Things (IoT)* dan *Computer Vision* dalam hal ini *Image Processing*. Pada aplikasi ini IoT berguna untuk mendeteksi secara *real-time* menggunakan sensor ultrasonik mengenai situasi ketersediaan slot parkir tepat pada setiap ruang parkir, yang mana mikrokontroler *ESP32* akan mengirimkan informasi ke server aplikasi. Sehingga, pengendara dapat memantau ketersediaan slot parkir sebelum memarkirkan mobil. Disamping itu juga, terdapat kemungkinan kerugian pengusaha pengelola parkir, pengendara yang memesan namun tidak memarkirkan mobil sesuai dengan yang telah dipesan. Untuk mengatasi kemungkinan rugi tersebut, penulis merancang suatu kewajiban pada pengendara yaitu memberikan bukti foto yang mencakup plat nomor polisi dan kertas penanda slot parkir diatasnya. Hal ini tentu tidak menyita waktu terlalu banyak karena telah diuntungkan oleh aplikasi *real-time* yang telah disediakan. Penggunaan metode *Faster Region based Convolutional Neural Network* untuk mendeteksi kertas penanda slot parkir dan plat nomor polisi memiliki beberapa keuntungan yaitu dalam mendeteksi tingkat akurasi yang baik, dapat diadaptasi untuk berbagai jenis objek deteksi. Setelah mendapatkan hasil objek seperti yang disebutkan, metode *Optical Character Recognition* akan memproses menjadi karakter pada masing masing objek. Hal ini sebagai bentuk validasi ketersesuaian dengan data plat nomor polisi yang diberikan oleh pengendara pada aplikasi pemesanan slot parkir. Dengan adanya rancangan aplikasi seperti ini, diharapkan mendapat hasil yang sangat efisien dan adil untuk pengguna, pengusaha pengelola parkir bahkan pemerintah. Oleh karena itu, penulis mengangkat penelitian yang berjudul “**SISTEM PARKIR CERDAS**

**TERINTEGRASI DENGAN INTERNET OF THINGS (IOT) DAN PEMROSESAN CITRA DIGITAL MENGGUNAKAN METODE *FASTER REGION BASED CONVOLUTIONAL NEURAL NETWORK (FASTER R-CNN)* DAN *OPTICAL CHARACTER RECOGNITION (OCR)*”.**

### **1.2. Rumusan Masalah**

Sehubungan dengan peningkatan jumlah kendaraan di perkotaan telah menyebabkan tempat yang akan dikunjungi oleh pengendara mobil dapat memerlukan waktu yang lama. Sering pengendara mobil menghabiskan waktu hanya untuk mencari slot parkir yang sedang kosong terlebih pada jam aktivitas tinggi, bahkan pengendara mobil keluar dari lokasi dikarenakan tidak adanya slot parkir. Hal ini disebabkan karena pengunjung tidak mengetahui pasti lokasi slot parkir yang dapat di duduki. Sehingga dibutuhkan inovasi yang mana men-digitalisasi setiap slot parkir yang dapat memberikan pendekatan informasi ke pengendara mobil. Berkaitan dengan pendekatan tersebut, pengendara dapat melacak dan/atau memesan slot parkir yang sedang tersedia.

Disamping itu timbul sebuah masalah baru bagi pengusaha parkir yaitu jika pengendara mobil tidak menduduki slot parkir yang telah dipesan sebelumnya. Hal ini dapat menyebabkan kerugian bagi pengusaha vendor parkir yang seharusnya slot dapat digunakan oleh pengendara mobil lainnya. Bagi pengusaha parkir juga membutuhkan sebuah transformasi yang dapat membuktikan keberadaan mobil telah memarkirkan sesuai pada slot yang sudah dipesan yang memeriksa kesesuaian lokasi slot parkir pemesan.

### **1.3. Tujuan Penelitian**

Penelitian ini bertujuan untuk meningkatkan efisiensi dalam mencari slot parkir melalui pengoptimalan sistem parkir cerdas menggunakan sensor yang dipasang pada setiap slot parkir. Tujuan utama penelitian adalah memanfaatkan data yang diterima dari sensor-sensor tersebut untuk memberikan visualisasi yang kreatif kepada pengguna, sehingga pengguna dapat dengan cepat dan efisien memantau ketersediaan ruang parkir serta melakukan pemesanan melalui aplikasi. Dalam upaya membantu meminimalisir kerugian bagi vendor parkir, penelitian ini mengadopsi teknologi *Faster R-CNN* untuk mendeteksi plat nomor kendaraan dan kertas penanda slot parkir serta teknologi ekstraksi karakter pada masing-masing objek yang diperlukan untuk validasi

pemesanan. Sehingga memastikan kesesuaian antara penggunaan ruang parkir yang telah dipesan dengan penggunaan aktual oleh pengendara.

#### **1.4. Batasan Masalah**

Penulis melakukan pembatasan penelitian yang sesuai dengan kebutuhan penelitian, yaitu:

1. Pendekripsi mobil menggunakan Sensor Ultrasonik dan mikrokontroler ESP32.
2. Dirancang untuk tiga slot parkir.
3. Plat yang memiliki karakter jelas dan sesuai standar.
4. Tidak mempertimbangkan penerapan penalti terhadap pengendara yang tidak memarkirkan kendaraannya sesuai dengan slot yang telah dipesan.

#### **1.5. Manfaat Penelitian**

Penelitian ini bermanfaat untuk:

1. Dengan menggunakan sistem yang terintegrasi dengan sensor, pengguna aplikasi dapat dengan mudah dan cepat untuk menentukan keputusan untuk mendapatkan slot parkir. Sebagai rincian, pengguna dapat menghemat waktu, uang, dan bahan bakar.
2. Vendor parkir dapat dengan baik dalam manajemen parkir dengan adanya pencatatan riwayat parkir.
3. Dengan menggunakan sistem yang terintegrasi dengan pendekripsi citra seperti penelitian ini, dapat meningkat dan memaksimalkan pendapatan parkir.
4. Hasil dari penelitian ini dapat dikembangkan melalui pendekatan yang serupa, pada dunia bisnis parkir.

#### **1.6. Metodologi Penelitian**

Metodologi penelitian yang akan dilakukan dalam penelitian berikut, yaitu:

1. Studi Literatur

Peneliti memulai dengan mencari dan mengumpulkan berbagai referensi yang telah diteliti dari berbagai penulisan melalui publikasi penelitian sampai pakar ahli yang memiliki keterkaitan dengan Sensor Ultrasonik, Mikrokontroler ESP 32, *Faster Region based Convolutional Neural Network (Faster R-CNN)*, *Easy Optical Character Recognition (EasyOCR)* dan bisnis parkir.

2. Pengumpulan data

Peneliti mengumpulkan data plat nomor polisi dan kertas penanda slot parkir yang akan dipelajari oleh sistem.

### 3. Analisis dan Perancangan sistem

Peneliti menganalisa teori-teori perancangan sistem sensor dan pengiriman data dari sensor ke aplikasi, kebutuhan aplikasi untuk pengguna, dan pendekripsi objek menggunakan teknologi *Faster Region based Convolutional Neural Network (Faster R CNN)* serta ekstraksi karakter menggunakan *Easy Optical Character Recognition (EasyOCR)*.

### 4. Implementasi sistem

Pada tahap implementasi, penulis membangun sistem-sistem dalam yang dibutuhkan dengan menerapkan teori yang telah dirancang sebelumnya.

### 5. Pengujian sistem

Pada pengujian sistem, menganalisa dan melakukan tinjauan kembali untuk dapat menemukan kekurangan pada sistem.

### 6. Dokumentasi

Dalam tahap dokumentasi, melampirkan laporan berupa video pengujian alat sensor, aplikasi, dan sistem pemrosesan citra digital.

### 7. Penyusunan Laporan

Penulis melakukan penyusunan laporan pada tahap ini dengan menyusun laporan penelitian yang menjadi hasil akhir yaitu skripsi.

## 1.7. Sistematika Penulisan

Pada penelitian ini terdapat sistematika yang mencakup lima bagian berikut:

### Bab 1: Pendahuluan

Memuat latar belakang penelitian, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, serta metodologi dan sistematika penulisan.

### Bab 2: Landasan Teori

Bab ini akan menjelaskan teori-teori yang berkaitan dengan definisi parkir beserta penjelasan terkait solusi *IT* yang diberikan oleh penulis.

### Bab 3: Analisis dan Perancangan

Bab ini mencakup desain sistem, arsitektur umum, dan alur perancangan sistem pada perangkat keras dan *deep learning* menggunakan metode *Faster Region Based Convolutional Neural Network (Faster R-CNN)* dan metode *EasyOCR*.

### Bab 4: Implementasi dan Pengujian

Berisikan pembahasan dari pengimplementasian sistem perangkat keras, pendekripsi citra digital serta ekstraksi karakter dan juga analisis hasil evaluasi.

### **Bab 5: Kesimpulan dan Saran**

Berisi semua kesimpulan yang telah dibahas pada seluruh bab sebelumnya serta saran yang diajukan untuk penelitian kedepannya.

## **BAB 2**

### **LANDASAN TEORI**

#### **2.1. Parkir**

Pengertian dasar parkir adalah ketika suatu kendaraan berhenti untuk sementara dan tidak bergerak (Direktorat Jenderal Perhubungan Darat, 1996). Praktik ini umumnya dilakukan untuk memungkinkan pengemudi atau penumpang meninggalkan kendaraan mereka dan melakukan aktivitas di tempat tujuan yang mereka kunjungi. Tujuan parkir bisa bermacam-macam, mulai dari berbelanja di pusat perbelanjaan, mengunjungi tempat wisata, bekerja, atau sekedar beristirahat sejenak. Fasilitas parkir merupakan bagian penting dari infrastruktur transportasi dalam memenuhi kebutuhan akan tempat parkir yang aman dan nyaman bagi kendaraan bermotor.

#### **2.2. Aplikasi Mobile**

Aplikasi mobile adalah perangkat lunak yang berjalan pada perangkat *mobile* seperti *smartphone* atau tablet *PC*. Aplikasi *Mobile* juga dikenal sebagai aplikasi yang dapat diunduh dan memiliki fungsi tertentu sehingga menambah fungsionalitas dari perangkat mobile itu sendiri. Untuk mendapatkan mobile application yang diinginkan, user dapat mengunduhnya melalui situs tertentu sesuai dengan sistem operasi yang dimiliki. *Google Play* dan *iTunes* merupakan beberapa contoh dari situs yang menyediakan beragam contoh dari situs yang menyediakan beragam aplikasi bagi pengguna *Android* dan *iOS* untuk mengunduh aplikasi yang diinginkan. (Mobile Marketing Association, 2015).

#### **2.3. *Internet of Things***

*Internet of things* merupakan terobosan yang bertujuan untuk mengatasi tantangan saat ini dengan menggabungkan teknologi dan dampak sosial, seperti yang dijabarkan dalam Rekomendasi ITU-T Y.2060 ditulis pada buku. Secara teknis, *IoT* dapat disamakan dengan sebuah infrastruktur global yang memenuhi kebutuhan informasi masyarakat

dengan mengintegrasikan unsur fisik dan virtual, yang didasarkan pada kemajuan dalam teknologi informasi dan komunikasi (*ICT*).

Kevin Ashton, yang mengusulkan konsep *Internet of Things*, menjelaskan dalam bukunya “*Making Sense of IoT*” bahwa *IoT* merujuk pada sensor-sensor yang terkoneksi dengan internet dan bertindak seperti jaringan internet dengan menjaga koneksi yang terus-menerus. Sensor ini berbagi data secara bebas, memungkinkan pengembangan aplikasi yang inovatif, dan memberikan kemampuan bagi komputer untuk memahami konteks lingkungan sekitarnya serta berinteraksi dalam kehidupan manusia. Untuk mempermudah proses penyimpanan dan pertukaran informasi, teknologi semantik menjadi komponen yang penting. Karenanya, untuk mewujudkan *Internet of Things*, diperlukan tiga elemen pendukung utama: *Internet*, *Things* dan *Semantic*.

### 2.3.1. Sensor pendetksi: sensor ultrasonik HC-SR04

Sensor Ultrasonik adalah sebuah sensor yang berfungsi untuk mengubah besaran fisis (bunyi) menjadi besaran listrik dan sebaliknya. Cara kerja sensor ini didasarkan pada prinsip dari pantulan suatu gelombang suara sehingga dapat dipakai untuk menafsirkan eksistensi (jarak) suatu benda dengan frekuensi tertentu adalah piezoelektrik yang membangkitkan gelombang ultrasonik (umumnya berfrekuensi 40kHz). Sistem pendetksi ini akan mengeluarkan sonar gelombang ultrasonik menuju suatu area, setelah sampai di titik permukaan terdekat, maka memantulkan kembali yang ditangkap kembali oleh sensor. Sebagai kalkulasinya yaitu selisih antara waktu pengiriman gelombang dan waktu gelombang dan waktu gelombang pantul diterima.



**Gambar 2.1** Sensor Ultrasonik HC-SR04

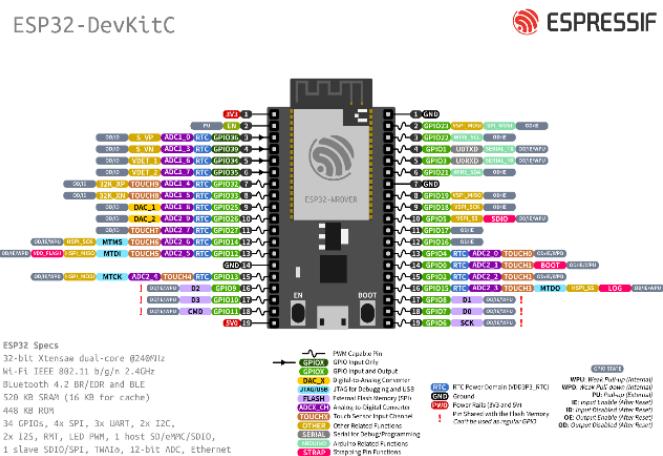
### 2.3.2. Mikrokontroler: ESP32

ESP32 merupakan mikrokontroler yang kuat dan serbaguna yang didesain khusus untuk aplikasi *Internet of Things* (*IoT*). Dikembangkan oleh perusahaan Tiongkok, Espressif Systems, ESP32 menawarkan beragam fitur yang membuatnya menjadi pilihan yang sangat sesuai untuk pengembangan proyek *IoT*. Mikrokontroler ini dilengkapi dengan kemampuan *Wi-Fi* dan *Bluetooth*, serta mendukung berbagai protokol komunikasi

lainnya yang diperlukan untuk menghubungkan perangkat *IoT* ke jaringan dan antar perangkat.

Contoh pengaplikasian ESP32 mencakup berbagai bidang seperti monitoring lingkungan, kontrol perangkat, aplikasi keamanan, sistem otomasi, dan aplikasi kesehatan. Dalam monitoring lingkungan, ESP32 dapat memantau parameter seperti suhu, kelembaban, dan kualitas udara, serta mengirimkan data ke server atau aplikasi untuk analisis lebih lanjut. Sementara itu, dalam aplikasi keamanan, ESP32 dapat digunakan dalam sistem alarm pintar, kamera pengawas, atau sistem pengunci pintu yang dapat dikontrol secara remote.

Fitur *ESP32* yang penting untuk pengembangan proyek *IoT* meliputi kemampuan *Wi-Fi* 802.11b/g/n, dan *Bluetooth* versi 4.2 yang terintegrasi, adanya dua inti prosesor untuk pemrosesan paralel, ukuran yang kecil untuk integrasi yang mudah dalam proyek-proyek berbeda, dan konsumsi energi yang rendah yang cocok untuk perangkat baterai atau dengan energi terbatas dan board ini memiliki dua versi yaitu 30 *GPIO* dan 36 *GPIO*. Selain itu, *ESP32* juga mendukung berbagai protokol komunikasi seperti *MQTT*, *CoAP*, dan *HTTP*, serta memiliki banyak pin *I/O* yang memungkinkan penghubungan dengan berbagai sensor dan perangkat eksternal tambahan.



**Gambar 2.2** *ESP32*

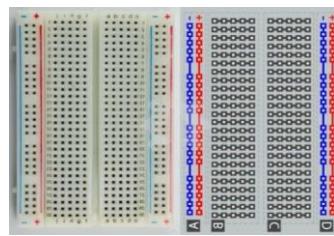
(Sumber: espressif.com)

### 2.3.3. Kabel jumper

Kabel *jumper* adalah kabel elektrik untuk menghubungkan antar komponen di breadboard tanpa memerlukan solder. Kabel jumper umumnya memiliki *connector* atau pin di masing-masing ujungnya. *Connector* untuk menusuk disebut *male connector*, dan *connector* yang ditusuk disebut *female connector*.

#### 2.3.4. Breadboard

*Breadboard* adalah dasar konstruksi sebuah sirkuit elektronik dan merupakan prototipe dari suatu rangkaian elektronik. Sifat *breadboard* tidak memerlukan solder sehingga dapat digunakan kembali dan dengan demikian sangat cocok digunakan pada tahapan proses pembuatan prototipe serta membantu dalam berkreasi dalam desain sirkuit elektronika. Sirkuit analog dan digital kecil sampai membuat unit pengolahan terpusat (*CPU*) dapat dimodelkan dengan menggunakan *breadboard*.



**Gambar 2.3 Breadboard**

#### 2.3.5. Power supply

*Power supply* pada *Internet of Things (IoT)* memainkan peran krusial dalam keberhasilan implementasi dan kinerja perangkat *IoT*. *Power supply* merujuk pada sistem atau perangkat yang menyediakan daya atau energi yang dibutuhkan oleh perangkat *IoT* untuk beroperasi. Dalam konteks ini, terdapat beberapa prinsip dasar yang perlu dipahami, antara lain efisiensi energi, pengelolaan daya, fleksibilitas, stabilitas, ketahanan, dan keamanan.

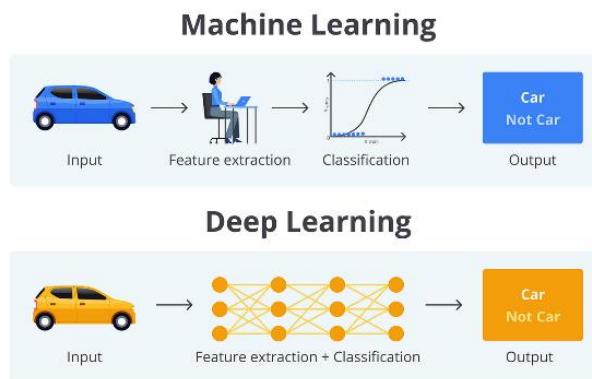
Efisiensi energi menjadi faktor penting karena perangkat *IoT* sering kali harus beroperasi dalam lingkungan dengan sumber daya terbatas, seperti baterai atau energi surya. Pengelolaan daya yang baik juga diperlukan untuk mengatur daya saat beroperasi, dalam *mode siaga*, atau saat proses *sleep* untuk menghemat daya. Fleksibilitas power supply *IoT* memungkinkan adaptasi terhadap berbagai sumber daya yang tersedia, seperti baterai, listrik AC, atau energi terbarukan. Stabilitas dan ketahanan *power supply* juga penting untuk menjaga kinerja perangkat *IoT* dalam menghadapi fluktuasi daya atau gangguan lainnya. Selain itu, keamanan *power supply* perlu diperhatikan untuk melindungi perangkat dari serangan yang dapat mempengaruhi pasokan daya dan mengganggu operasinya. Dengan memahami prinsip-prinsip dasar ini, pengembang *IoT* dapat merancang sistem *power supply* yang sesuai dengan kebutuhan dan lingkungan operasional perangkat *IoT*.

### 2.3.6. Arduino IDE

*Software Arduino IDE* adalah sebuah aplikasi yang digunakan untuk mengendalikan mikrokontroler *single-board* yang bersifat *open source*. Dirancang untuk memperlancar pengguna, aplikasi ini memungkinkan pembuatan, pengeditan, kompilasi, dan pengunggahan program ke dalam *board* Arduino. Dengan menggunakan bahasa pemrograman C/C++, *Arduino IDE* menyediakan berbagai fungsi lengkap yang memudahkan pengguna, terutama yang baru dalam mempelajari teknologi ini.

### 2.4. Deep Learning

*Deep learning* adalah salah satu cabang dari machine learning yang mengadaptasi prinsip kerja jaringan saraf tiruan yang terinspirasi dari struktur otak manusia, yang dikenal sebagai *Artificial Neural Networks (ANN)* (Setiawan, 2021). *Deep learning* terdiri dari beberapa jenis algoritma yang kompleks, yang memungkinkan komputer untuk belajar dan beradaptasi terhadap data dalam skala besar, serta menyelesaikan berbagai permasalahan yang sulit dilakukan oleh algoritma *machine learning* konvensional.



**Gambar 2.4** Perbedaan *Machine Learning* dan *Deep Learning*

(Sumber: medium.com)

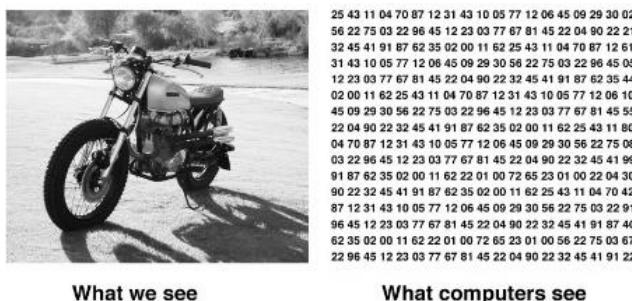
Salah satu algoritma yang populer dalam deep learning adalah *Convolutional Neural Network (CNN)*, yang digunakan untuk memproses dan mengekstrak fitur dari data gambar, sehingga berguna dalam mendeteksi objek dalam gambar atau video. Ada juga *Recurrent Neural Network (RNN)* dan *Long Short Term Memory Network (LSTM)* yang dirancang khusus untuk mengolah data berurutan seperti teks atau *time series data*, dan *Self Organizing Maps (SOM)* yang dapat membuat visualisasi data secara mandiri. Penerapan *deep learning* memiliki berbagai manfaat, termasuk kemampuan untuk memproses data tidak terstruktur seperti teks dan gambar, otomatisasi ekstraksi fitur,

hasil akhir yang berkualitas, pengurangan biaya operasional, dan manipulasi data yang efektif.

Beberapa contoh penerapan *deep learning* meliputi pengenalan gambar untuk mendeteksi objek dalam gambar atau video, pengenalan suara untuk mengenali dan merespons suara manusia, pemrosesan bahasa alami (*Natural Language Processing/NLP*) untuk menganalisis dan memahami bahasa manusia, dan deteksi anomali untuk mengidentifikasi pola yang tidak biasa atau tidak sesuai dengan perilaku yang diharapkan. Dengan penggunaan yang luas dan terus berkembang, deep learning telah menjadi teknologi penting dalam berbagai aspek kehidupan sehari-hari dan berperan dalam berbagai produk dan layanan yang kita gunakan.

### **2.5. Computer Vision**

*Computer Vision* adalah proses transformasi data dari kamera video atau gambar menjadi keputusan atau presentasi baru yang memiliki tujuan tertentu. Data yang diolah dalam proses ini dapat mencakup informasi kontekstual dari gambar, seperti keberadaan objek atau identitas seseorang. Transformasi tersebut bisa berupa pengambilan keputusan terkait gambar, misalnya untuk mendeteksi keberadaan telapak tangan atau mengidentifikasi individu dalam foto. Selain itu, proses tersebut juga dapat menghasilkan perubahan presentasi, seperti mengubah gambar menjadi *grayscale* atau memotong objek dalam gambar.



**Gambar 2.5** Perbedaan Visualisasi Manusia dan Komputer Terhadap Citra Digital

(Sumber: [freecontent.manning.com](http://freecontent.manning.com))

Meskipun manusia memiliki kemampuan untuk meneliti, memahami, dan membandingkan informasi langsung dari objek, komputer dalam sistem penglihatan mesin hanya dapat memperoleh informasi dalam bentuk angka-angka dari media *input* seperti kamera atau disket. Komputer melihat objek dalam gambar sebagai kumpulan angka tanpa interpretasi yang sama seperti manusia. Hal ini terlihat dalam gambar di mana kaca samping mobil hanya diinterpretasikan sebagai kumpulan angka oleh

komputer. Keterbatasan penglihatan komputer dalam dimensi (2D) berbanding dengan objek dalam dunia nyata yang memiliki dimensi tiga (3D) menyebabkan informasi yang diperoleh seringkali terganggu oleh noise atau gangguan, seperti cuaca, cahaya, bayangan, refleksi cahaya, dan gerakan. Oleh karena itu, masalah ini terus diteliti untuk menemukan cara atau teknik penyelesaiannya.

#### 2.5.1. *Object detection*

*Object detection* adalah salah satu cabang dari computer vision yang bertujuan untuk mendeteksi keberadaan dan lokasi objek-objek tertentu dalam sebuah gambar atau video. Tujuan utama dari *object detection* adalah untuk mengidentifikasi dan menandai objek-objek yang berbeda, serta menggambarkan kotak pembatas (*bounding box*) yang menunjukkan posisi dan ukuran relatif dari setiap objek tersebut dalam gambar.

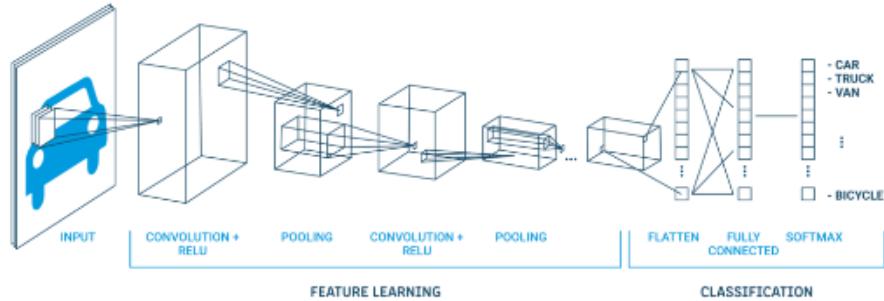
Teknik *object detection* memungkinkan komputer untuk secara otomatis mengenali objek-objek yang berbeda dalam gambar atau video, seperti mobil, manusia, hewan, atau objek lainnya. Ini adalah tugas yang sangat penting dalam berbagai aplikasi, termasuk keamanan, pengawasan video, analisis medis, kendaraan otonom, dan banyak lagi.

Metode yang umum digunakan dalam *object detection* termasuk penggunaan *Convolutional neural network (CNN)* dan teknik-teknik seperti *Region-based CNN (R-CNN)*, *YOLO (You Only Look Once)*, dan *SSD (Single Shot MultiBox Detector)*. Dengan menggunakan teknik-teknik ini, *object detection* dapat dilakukan dengan tingkat akurasi yang tinggi dan efisien, sehingga memungkinkan aplikasi yang lebih luas dalam berbagai bidang.

#### 2.5.2. *Convolutional Neural Network (CNN)*

*Convolutional Neural Network (CNN)* merupakan metode *neural network* yang dirancang khusus untuk mendeteksi, menganalisis, dan mengenali objek dalam gambar. Arsitektur *CNN* terdiri dari dua bagian utama, yaitu lapisan ekstraksi fitur dan lapisan yang terhubung penuh (*multilayer perceptron*). Tujuan utama dari *CNN* adalah untuk mengurangi jumlah parameter dan menyederhanakan arsitektur jaringan saat melakukan pengenalan objek. *CNN* memanfaatkan teknik konvolusi, di mana sebuah kernel konvolusi digerakkan ke seluruh gambar untuk menghasilkan informasi representatif baru dari citra visual, yang kemudian digunakan untuk klasifikasi objek. Algoritma *CNN* dapat melakukan pembelajaran mandiri dalam proses pengenalan objek dan dapat diterapkan pada citra dengan resolusi tinggi. Dengan memanfaatkan

parameter seperti pergeseran filter (*stride*) dan nilai nol pada piksel *input*, CNN dapat memanipulasi *feature map* pada dimensi output untuk memperoleh representasi yang optimal dari objek yang diidentifikasi.

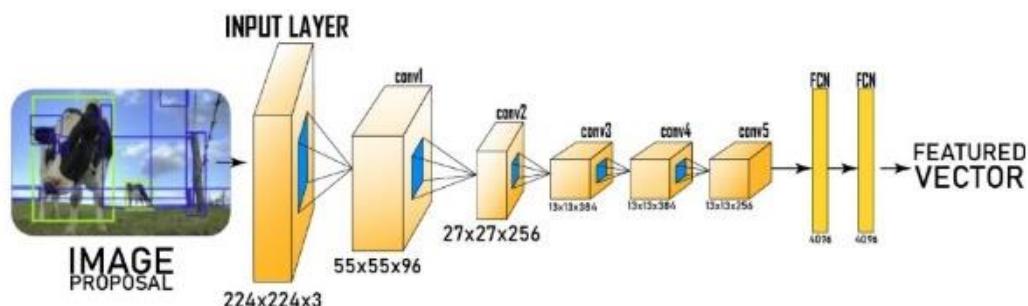


**Gambar 2.6** Arsitektur CNN

(Sumber: medium.com)

#### 2.5.3. Region-based Convolutional Neural Network (R-CNN)

R-CNN merupakan singkatan dari *Region-based Convolutional Neural Network*, jenis model *machine learning* yang digunakan untuk tugas *computer vision*, khususnya deteksi objek. Untuk memahami apa itu R-CNN, kita akan melihat arsitektur R-CNN. Konsep kerja R-CNN menggunakan kotak pembatas (*bounding boxes*) pada wilayah objek, kemudian mengevaluasi jaringan konvolusi secara mandiri pada semua Wilayah Minat (*Regions of Interest/ROI*) untuk mengklasifikasikan beberapa wilayah gambar ke dalam kelas yang diusulkan. Arsitektur R-CNN dirancang untuk menyelesaikan tugas deteksi gambar, dan menjadi dasar bagi pengembangan arsitektur *Faster R-CNN*.



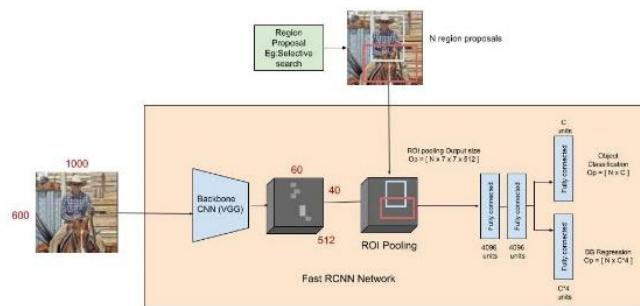
**Gambar 2.7** Arsitektur R-CNN

(Sumber: superrask.xyz)

#### 2.5.4. Fast Region Convolutional Neural Network (Fast R-CNN)

Fast R-CNN merupakan pengembangan dari metode R-CNN yang dikembangkan pada tahun 2015 oleh Ross Girshick (Girshick *et al.*, 2013). Tujuan dari Fast R-CNN adalah untuk mengatasi masalah waktu yang muncul dalam proses pelatihan dan pengujian model R-CNN. Metode ini menawarkan solusi yang lebih efisien dengan

menyederhanakan arsitektur dan meningkatkan kecepatan komputasi. Salah satu perbedaan utama antara *Fast R-CNN* dengan *R-CNN* adalah penggunaan satu model *Convolutional Neural Network (CNN)* untuk ekstraksi fitur dari setiap *region*, sehingga mengurangi redundansi komputasi. Proses kerjanya dimulai dengan ekstraksi fitur menggunakan *CNN* untuk setiap region yang dihasilkan oleh *Region Proposal Network (RPN)*, kemudian fitur-fitur tersebut dicocokkan dengan *Region of Interest (ROI)* dan digunakan untuk klasifikasi objek. *R-CNN* berperan sebagai pengklasifikasi dengan menggunakan teknik *ROI Pooling* untuk mengekstraksi fitur dari *ROI* dan lapisan terhubung penuh yang digunakan oleh *Support Vector Machine (SVM)* untuk klasifikasi.

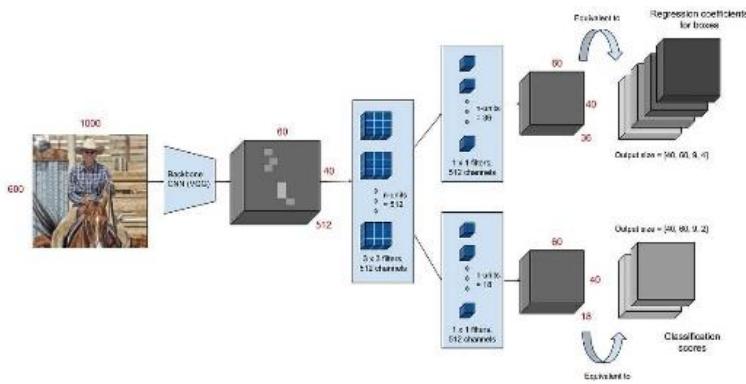


**Gambar 2.8** Arsitektur *Fast R-CNN*

(**Sumber:** towardsdatascience.com)

#### 2.5.5. Faster Region Convolutional Neural Network (*Faster R-CNN*)

*Faster R-CNN* adalah pengembangan dari *Fast R-CNN* yang merupakan salah satu metode *deep learning* yang digunakan untuk mendeteksi objek dalam citra dengan mengeksplorasi karakteristik objek melalui proses konvolusi atau *Convolutional Neural Network (CNN)*. Metode ini memiliki tiga modul utama, yakni *deep fully convolutional network* yang mengusulkan *region*, *detector* *Fast R-CNN* yang menggunakan *region* yang diusulkan, dan *Region Proposal Network (RPN)* yang mengarahkan *Fast R-CNN* untuk menganalisis citra dengan kecepatan yang signifikan. *RPN* digunakan untuk mengklasifikasikan objek dan menghasilkan *feature map* dengan *deep neural network*, menghasilkan proposal wilayah, dan melakukan regresi serta menambahkan lapisan *convolutional* untuk meningkatkan kualitas deteksi objek.



**Gambar 2.9** Arsitektur *Faster R-CNN*

(Sumber: towardsdatascience.com)

#### 2.5.6. Perbedaan CNN, R-CNN, Fast R-CNN dan Faster R-CNN

Berikut adalah perbedaan antara *CNN*, *R-CNN*, *Fast R-CNN*, dan *Faster R-CNN*:

- 1) *Convolutional Neural Network (CNN):*
    - a. *CNN* adalah arsitektur neural network yang digunakan untuk memproses data grid, seperti gambar.
    - b. *CNN* menggunakan operasi konvolusi untuk mengekstraksi fitur-fitur penting dari data *input*, yang kemudian digunakan untuk klasifikasi, deteksi objek, atau tugas lainnya.
    - c. *CNN* umumnya digunakan untuk klasifikasi gambar, di mana *input* adalah gambar dan output adalah label kelas yang mewakili objek dalam gambar.
  - 2) *Region-Based Convolutional Neural Network (R-CNN):*
    - a. *R-CNN* adalah metode deteksi objek yang menggunakan pendekatan *region-based* untuk mengidentifikasi dan memisahkan objek dalam gambar.
    - b. *R-CNN* melakukan proposisi wilayah (*region proposal*) terlebih dahulu, kemudian memproses setiap proposal menggunakan *CNN* untuk menghasilkan prediksi kelas dan kotak pembatas (*bounding box*) yang mengelilingi objek.
    - c. *R-CNN* memerlukan waktu yang lama untuk mengeksekusi karena proses ekstraksi fitur dilakukan secara terpisah untuk setiap proposal wilayah.
  - 3) *Fast R-CNN:*
    - a. *Fast R-CNN* adalah evolusi dari *R-CNN* yang memperbaiki kelemahan waktu eksekusi *R-CNN* dengan menggabungkan proses ekstraksi fitur dan deteksi objek menjadi satu langkah.

- b. Dalam *Fast R-CNN*, *CNN* digunakan untuk mengekstraksi fitur dari seluruh gambar secara keseluruhan, bukan untuk setiap proposal wilayah secara terpisah.
  - c. Dengan demikian, *Fast R-CNN* lebih cepat daripada *R-CNN* karena proses ekstraksi fitur hanya dilakukan sekali untuk seluruh gambar.
- 4) *Faster R-CNN*:
- a. *Faster R-CNN* adalah pengembangan lebih lanjut dari *Fast R-CNN* yang memperkenalkan *Region Proposal Network (RPN)* untuk menghasilkan proposal wilayah secara otomatis.
  - b. *RPN* digunakan untuk mengidentifikasi potensi lokasi objek dalam gambar dan menghasilkan proposal wilayah berdasarkan potensi tersebut.

Dengan adanya *RPN*, *Faster R-CNN* dapat menghasilkan proposal wilayah dengan lebih cepat dan secara efisien, sehingga meningkatkan kecepatan dan akurasi deteksi objek.

#### 2.5.7. PyTorch

*PyTorch* adalah kerangka kerja komputasi ilmiah sumber terbuka yang digunakan untuk aplikasi pembelajaran mesin, terutama visi komputer dan pemrosesan bahasa alami. Ini dikembangkan oleh tim di Facebook *Research* dan dirilis pada tahun 2016. *PyTorch* didasarkan pada *Python* dan berfokus pada kemudahan penggunaan, fleksibilitas, dan kinerja.

Prinsip Utama *PyTorch*:

1. *Tensor*: Unit dasar komputasi di *PyTorch* adalah *tensor*, yang merupakan struktur data multidimensi yang dapat mewakili berbagai jenis data numerik. *Tensor* dapat dioperasikan dengan berbagai cara, termasuk penjumlahan, perkalian, dan transposisi.
2. Komputasi Komponen: *PyTorch* mendukung komputasi komponen, yang memungkinkan Anda untuk membangun model pembelajaran mesin dengan menghubungkan blok-blok kode yang dapat digunakan kembali. Hal ini memungkinkan modularitas dan fleksibilitas yang lebih besar dalam pengembangan model.
3. Diferensiasi Otomatis: *PyTorch* menyediakan dukungan bawaan untuk diferensiasi otomatis, yang memungkinkan Anda untuk menghitung gradien

fungsi secara efisien. Hal ini penting untuk melatih model pembelajaran mesin menggunakan algoritma propagasi balik.

Keuntungan *PyTorch*:

1. Kemudahan Penggunaan: *PyTorch* memiliki sintaks yang intuitif dan mirip dengan *Python*, membuatnya mudah dipelajari dan digunakan.
2. Fleksibilitas: *PyTorch* memungkinkan untuk membangun model pembelajaran mesin yang kompleks dengan mudah menggunakan komputasi komponen dan dukungan Python yang luas.
3. Kinerja: *PyTorch* dioptimalkan untuk kinerja dan dapat digunakan pada berbagai platform, termasuk *CPU*, *GPU*, dan *TPU*.
4. Komunitas: *PyTorch* memiliki komunitas yang besar dan aktif yang menyediakan dukungan dan sumber daya yang bermanfaat.

Aplikasi *PyTorch*:

1. Visi Komputer: *PyTorch* banyak digunakan dalam aplikasi visi komputer seperti klasifikasi gambar, deteksi objek, dan segmentasi gambar.
2. Pemrosesan Bahasa Alami: *PyTorch* juga populer dalam aplikasi pemrosesan bahasa alami seperti terjemahan mesin, analisis sentimen, dan pemodelan bahasa.
3. Pembelajaran Penguat: *PyTorch* dapat digunakan untuk mengembangkan agen pembelajaran penguat yang dapat belajar untuk mengambil keputusan optimal dalam lingkungan yang kompleks.

#### 2.5.8. *Confusion Matrix*

Proses kinerja pembelajaran mesin memerlukan pengukuran performa menggunakan confusion matrix. *Confusion matrix* terdiri dari 4 kombinasi berbeda nilai prediksi dan nilai aktual dengan bentuk tabel. Terdapat empat istilah yang menjadi representasi hasil proses pada confusion matrix yaitu:

1. *True positive* adalah kondisi data yang diprediksi oleh model bernilai positif dan data aktual juga positif.
2. *True negative* adalah kondisi data yang diprediksi model bernilai negatif dan data actual juga bernilai negatif.
3. *False positive* adalah kondisi data yang diprediksi model bernilai positif namun data actual bernilai negatif

4. *False negative* adalah kondisi data yang diprediksi model bernilai negatif namun data actual bernilai positif.

Dengan empat kombinasi tersebut selanjutnya dapat dilakukan pengukuran performa dengan matrix turunan dari *confusion matrix*:

#### 1. *Accuracy*

*Accuracy* merupakan persentase benar dari nilai prediksi dibanding dengan nilai aktual dari data. Semakin tinggi nilai akurasi, maka dapat disimpulkan semakin akurat model akan mengklasifikasikan data dengan benar. Perhatikan persamaan *Accuracy* pada persamaan 2.1.

$$\text{Accuracy} = \frac{TP + TN}{\text{Total Data}} \quad (2.1)$$

#### 2. *Precision*

*Precision* adalah perbandingan nilai prediksi benar positif dengan keseluruhan data yang diprediksi positif. Perhatikan persamaan *Precision* pada rumus persamaan 2.2.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2.2)$$

#### 3. *Recall*

*Recall* adalah perbandingan nilai prediksi benar positif dengan keseluruhan data yang benar positif. Perhatikan persamaan *Recall* pada rumus perhitungan 2.3.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2.3)$$

#### 4. *F1-Score*

*F1-score* merupakan perbandingan rata-rata dari *precision* dan *recall*. Perhatikan persamaan 2.4.

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (2.4)$$

## 2.6. Penelitian Terdahulu

Terdapat beberapa penelitian yang berkaitan dengan sensor pendekripsi dan pengiriman data berbasis *Internet of Things*, yang pertama yaitu penelitian yang dilakukan oleh (Istiqbal *et al.*, 2023), dengan tujuan mengembangkan sebuah prototipe yang

menggunakan *NodeMCU* ESP8266 dan Arduino UNO sebagai pengolah data sensor serta pengirim data sensor ke platform Blynk. Pengembangan ini bertujuan untuk memberikan akses kepada pengguna melalui aplikasi Blynk guna memantau informasi tentang ketersediaan slot parkir, baik yang kosong maupun yang sudah terisi. Selain itu, penelitian ini juga melibatkan penggunaan web server yang berfungsi untuk menginput data pengendara atau pengguna yang akan melakukan parkir. Web server tersebut memungkinkan pengguna untuk memasukkan *ID RFID Tag* atau E-KTP dan nomor plat kendaraan mereka. Proses pengambilan *ID RFID Tag* atau E-KTP dari pengguna dilakukan dengan menggunakan *RFID reader* yang terhubung dengan *NodeMCU* ESP8266. Selanjutnya, data ID tersebut dikirimkan ke *web server* melalui jaringan *WiFi* secara lokal untuk proses lebih lanjut.

Selanjutnya, terdapat penelitian yang dilakukan oleh (Pratama *et al.*, 2023), menunjukkan bahwa implementasi teknologi *IoT* memungkinkan pemantauan *real-time* terhadap kondisi parkir serta deteksi keberadaan kendaraan. Penggunaan sensor ultrasonik dan infrared dalam sistem tersebut terbukti efektif dalam mendekripsi keberadaan kendaraan dan memonitor area parkir. Komunikasi antara sensor dan perangkat lainnya melalui protokol *MQTT* memfasilitasi pertukaran informasi secara efisien dan cepat. Metode pengembangan prototipe juga memungkinkan pengguna untuk memberikan evaluasi dan umpan balik yang digunakan untuk memperbaiki aplikasi final. Pengujian sistem menunjukkan bahwa waktu rata-rata sistem terhubung ke *MQTT* adalah sekitar 6,3 detik.

Penelitian (Anas & Sakti, 2020) memfokuskan pada pengembangan *Smart Parking System* berbasis mikrokontroler menggunakan *ESP32 NodeMCU* sebagai mikrokontroler dan sensor *infrared* untuk mengecek status pada slot parkir. Tujuan penelitian ini adalah memberikan kemudahan kepada pengendara dalam mengetahui ketersediaan slot parkir dan melakukan booking pada slot yang diinginkan melalui aplikasi *mobile*. Selain itu, penggunaan *parking lock* pada slot parkir memastikan bahwa hanya pengendara yang sudah melakukan *booking* yang dapat menggunakan slot tersebut. Hasil pengujian menunjukkan bahwa sensor *infrared* dan *parking lock* berfungsi dengan baik, sedangkan aplikasi *mobile* berhasil memberikan informasi ketersediaan slot parkir dan memungkinkan pengendara untuk melakukan booking. Kesimpulan dari penelitian ini menegaskan bahwa penggunaan aplikasi booking parkir dapat memudahkan pengendara, sensor *infrared* berfungsi dengan baik dalam

mendeteksi status slot parkir, aplikasi mobile berhasil memberikan informasi dan memfasilitasi booking slot parkir, dan mikrokontroler dapat dikontrol melalui aplikasi *mobile*.

Pada aspek pendekripsi karakter pada plat nomor polisi dan karakter pada teks penanda parkir, yang pertama yaitu dilakukan oleh (Aprilino & Amin, 2022), berbasis *You Only Look Once (YOLO)* sebagai solusi atas kendala metode manual dalam deteksi plat nomor kendaraan yang memakan waktu dan tenaga manusia yang cukup besar. Penelitian ini menggunakan *dataset pretrained model YOLO v3* sebanyak 700 data dan proses ekstraksi teks plat nomor menggunakan *library Tesseract Optical Character Recognition (OCR)*, dengan hasil yang disimpan dalam database. Sistem ini berbasis *web* dan *API*, memungkinkan penggunaan secara *online* dan lintas *platform*. Hasil pengujian menunjukkan tingkat akurasi pendekripsi plat nomor otomatis mencapai 100% dengan pencahayaan yang memadai dan *threshold* sebesar 0.5, sementara hasil deteksi karakter menggunakan *library Tesseract* mencapai 92,32%. Kesimpulan dari penelitian ini menunjukkan bahwa *YOLO v3* mampu mendekripsi plat nomor dengan baik dalam berbagai kondisi gambar, sementara *Tesseract* sangat tergantung pada tahap *pre-processing* untuk hasil deteksi yang optimal. Sistem ini dapat diakses dengan baik melalui *web browser* dan *API* secara *online*.

Selanjutnya, penelitian yang dilakukan oleh (Firswandy *et al.*, 2019), mengkaji integrasi teknik segmentasi *SCW* dalam sistem *LPR* untuk operasi *outdoor* dengan penyesuaian parameter uji yang optimal. Metode tersebut meliputi pra-pemrosesan citra (*grayscale*), segmentasi (*SCW*, *image masking (Sauvola)*, dan *Connected Component Analysis (aspect ratio, orientasi, dan bilangan Euler)*). Setelah pengujian pada dataset citra plat nomor, ditemukan bahwa 70% berhasil terdeteksi, 5% terdeteksi sebagian, dan 25% tidak terdeteksi. Faktor kegagalan utama adalah plat nomor yang buram, latar belakang yang terlalu terang, dan orientasi plat nomor yang miring. Saran untuk penelitian berikutnya termasuk peningkatan presisi nilai *threshold* serta eksplorasi metode alternatif untuk mengatasi orientasi dan kemiringan plat nomor.

Penelitian (Musyafira *et al.*, 2020) berfokus pada pengembangan sistem *License Plate Recognition (LPR)* yang dapat membaca *input* dari data video dengan pengolahan *real-time*. Metode yang digunakan meliputi lokalisasi menggunakan *Single Shot Detector*, segmentasi *Binary Image* diikuti dengan *Connected Component Labelling*, dan pengenalan karakter menggunakan *Recurrent Neural Network*. Hasil penelitian

menunjukkan akurasi sebesar 91.48% untuk lokalisasi plat, 82.69% untuk segmentasi, dan 94.94% untuk pengenalan karakter. Selain itu, berdasarkan uji coba parameter dan augmentasi anotasi, ditemukan bahwa model dengan preset *VGG300* dan *learning rate* 0.0001 memberikan akurasi yang optimal, sementara arsitektur RNN dengan *optimizer* Adam dan *learning rate* 0.0001 serta ukuran k sebesar 9 pada *K-Folds Cross Validation* memberikan hasil terbaik. Metode segmentasi karakter juga berhasil dengan akurasi rata-rata mencapai 82.69%. Sistem deteksi dan pengenalan plat nomor kendaraan berhasil diimplementasikan dengan akurasi tertinggi berturut-turut 96.99% dan 99.69%, serta akurasi pengenalan karakter rata-rata sebesar 91.48%, 84.70%, dan 94.94%.

Penelitian yang dilakukan oleh (Basthom & Utaminingrum, 2021), menitikberatkan pada pembuatan kursi roda pintar yang mampu mendeteksi dan mengenali plat nama ruangan dalam lingkungan rumah sakit, sebagai respons terhadap pandemi *COVID-19* yang mengharuskan penjagaan jarak. Metode yang digunakan mencakup pemodelan *Faster-RCNN* untuk pendekripsi plat nama ruangan dan pengenalan menggunakan *PyTesseract*. Pengujian dilakukan pada purwarupa kursi roda pintar menggunakan *Raspberry Pi 4B*, dengan hasil integrasi perangkat keras mencapai 100% untuk fungsi *buzzer* dan motor, serta integrasi perangkat lunak mencapai 95% untuk pendekripsi plat nama ruangan dan 81% untuk pengenalan nama ruangan. Selain itu, hasil akurasi pendekripsi menggunakan *Faster-RCNN* mencapai 87%, dengan waktu komputasi rata-rata 6.825 detik per gambar, sedangkan rasio prediksi pengenalan menggunakan *PyTesseract* mencapai 77.73% dengan waktu komputasi rata-rata 2.54 detik per gambar. Rangkuman seluruh penelitian terdahulu yang telah dipaparkan, disajikan pada Tabel 2.1.

**Tabel 2.1** Penelitian terdahulu

No.	Penulis	Metode	Judul	Keterangan
1.	(Istiqlal <i>et al.</i> , 2023)	-	Prototype Smart Parking Berbasis IoT	Menggunakan <i>NodeMCU</i> ESP8266 dan <i>Arduino UNO</i> sebagai pengolah data sensor dan pengirim data sensor ke <i>platform Blynk</i> .

**Tabel 2.1** Penelitian Terdahulu (Lanjutan)

No.	Penulis	Metode	Judul	Keterangan
-----	---------	--------	-------	------------

No.	Penulis	Metode	Judul	Keterangan
2.	(Pratama <i>et al.</i> , - 2023)	Perancangan Dan Implementasi Protokol <i>MQTT</i> Pada Sistem Parkir Cerdas Berbasis <i>IoT</i>	Perancangan Dan Implementasi Protokol <i>MQTT</i> Pada Sistem Parkir Cerdas Berbasis <i>IoT</i>	Penggunaan <i>NodeMCU</i> dan <i>ESP8266</i> , sensor ultrasonik, dan <i>infrared</i> dalam sistem parkir terbukti efektif dalam mendekripsi keberadaan kendaraan serta memonitor area parkir, sementara komunikasi melalui protokol <i>MQTT</i> memfasilitasi pertukaran informasi yang efisien dan cepat, serta dengan pengujian sistem menunjukkan waktu rata-rata terhubung ke <i>MQTT</i> sekitar 6,3 detik.
3.	(Anas & Sakti, - 2020)	Perancangan Sistem Aplikasi Booking Parkir Menggunakan Sensor <i>Infrared</i> Berbasis <i>Internet Of Things</i>	Perancangan Sistem Aplikasi Booking Parkir Menggunakan Sensor <i>Infrared</i> Berbasis <i>Internet Of Things</i>	Menggunakan <i>ESP32</i> sebagai mikrokontroler dan sensor <i>infrared</i> , pengiriman data menggunakan protokol <i>MQTT</i> dan ditampilkan menggunakan aplikasi berbasis <i>mobile</i> .
4.	(Aprilino & Amin, 2022)	<i>Yolo</i> dan <i>OCR</i>	Implementasi Algoritma <i>Yolo</i> Dan <i>Tesseract</i>	Hasil pengujian menunjukkan tingkat akurasi pendekripsi plat

**Tabel 2.1** Penelitian Terdahulu (Lanjutan)

No.	Penulis	Metode	Judul	Keterangan
-----	---------	--------	-------	------------

			<i>OCR</i> Pada Sistem Deteksi Plat Nomor Otomatis	nomor otomatis mencapai 100% dengan pencahayaan yang memadai dan <i>threshold</i> sebesar 0.5, sementara hasil deteksi karakter menggunakan <i>library</i> <i>Tesseract</i> mencapai 92,32%.
5.	(Firswandy <i>et al.</i> , 2019)	Sliding Concentric Windows (SCW)	Deteksi Plat Nomor Kendaraan Bermotor Menggunakan Sliding Concentric Windows (SCW) Untuk Aplikasi Sistem Transportasi Cerdas	Metode tersebut meliputi pra-pemrosesan citra (grayscale), segmentasi (SCW, image masking (Sauvola), dan Connected Component Analysis (aspect ratio, orientasi, dan bilangan Euler)). Setelah pengujian pada dataset citra plat nomor, ditemukan bahwa 70% berhasil terdeteksi, 5% terdeteksi sebagian, dan 25% tidak terdeteksi.
6.	(Musyafira <i>et al.</i> , 2020)	<i>Single Shot Detector</i> ( <i>SSD</i> ) dan <i>Recurrent Neural Network (R-NN)</i>	Deteksi Posisi Dan Pengenalan Plat Nomor Kendaraan Menggunakan <i>Single Shot Detector</i> Dan	Sistem deteksi dan pengenalan plat nomor kendaraan berhasil diimplementasikan dengan akurasi tertinggi berturut-turut 96.99% dan 99.69%, serta akurasi pengenalan

**Tabel 2.1** Penelitian Terdahulu (Lanjutan)

No.	Penulis	Metode	Judul	Keterangan
			<i>Recurrent Neural Network Pada Data Video</i>	karakter rata-rata sebesar 91.48%, 84.70%, dan 94.94%.
7.	(Basthom & Utaminingsrum, 2021)	<i>Faster R-CNN</i>	Deteksi dan Pengenalan Plat Nama Ruangan menggunakan <i>Faster-RCNN</i> dan <i>Pytesseract</i> pada Purwarupa Kursi Roda Pintar	Hasil pendekripsi menggunakan <i>Faster-RCNN</i> mencapai 87%, dengan waktu komputasi rata-rata 6.825 detik per gambar, sedangkan rasio prediksi pengenalan menggunakan <i>PyTesseract</i> mencapai 77.73% dengan waktu komputasi rata-rata 2.54 detik per gambar

## BAB 3

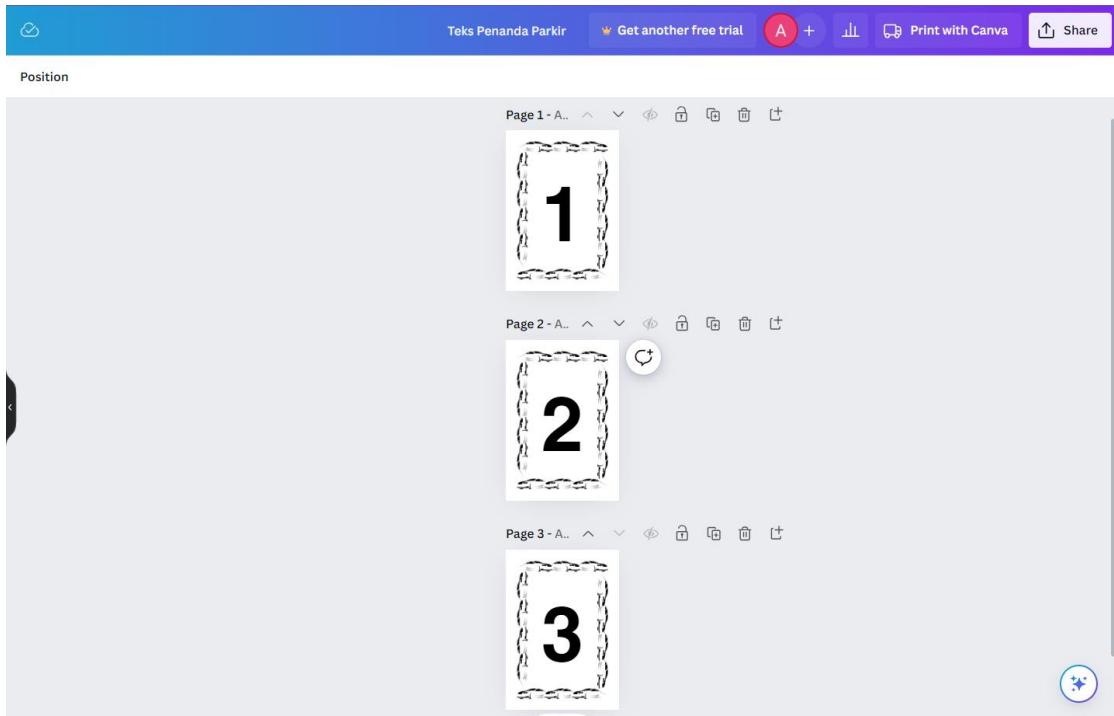
### ANALISIS DAN PERANCANGAN SISTEM

Pada bab ini membahas analisis dengan perancangan sistem terintegrasi aplikasi berbasis *mobile* antara data *realtime* dari sensor dan pemrosesan citra digital menggunakan metode *Faster Region Based Convolutional Neural Network*. Hasil dari inferensi citra digital yang terdeteksi diantaranya: kertas penanda slot parkir dan plat nomor polisi pada mobil, diekstrak menjadi karakter yang akan divalidasi kebenarannya sesuai dengan data inputan pengguna pada antarmuka berbasis *mobile*.

#### 3.1. Pengumpulan Data

##### 3.1.1. Custom data kertas penanda slot parkir

Terkhusus untuk kertas penanda parkir, penulis mendesain pada platform *Canva*. Penulis menggunakan *font Helvetica* dan menambahkan *border* unik di sekeliling kertas guna menambah keunikan yang konsisten dalam pembelajaran citra.



Gambar 3.1 Custom pada Canva Kertas Penanda Parkir

Pada penerapannya setelah di desain, penulis mencetak kertas penanda slot parkir tersebut pada kertas A4. Kemudian memotret gambar dari berbagai posisi dan pose guna memperbanyak citra yang berbeda. Data ini diakumulasikan sebanyak 1.183 data citra kertas penanda slot parkir yang kedepannya dapat digunakan untuk pengenalan citra.

### *3.1.2. Data plat nomor polisi*

Pada tahap pengumpulan data plat nomor polisi dikumpulkan sebanyak 1.782 gambar. Data citra dikumpulkan dari *Roboflow* dan data pribadi menggunakan kamera.

### *3.1.3. Akumulasi data citra*

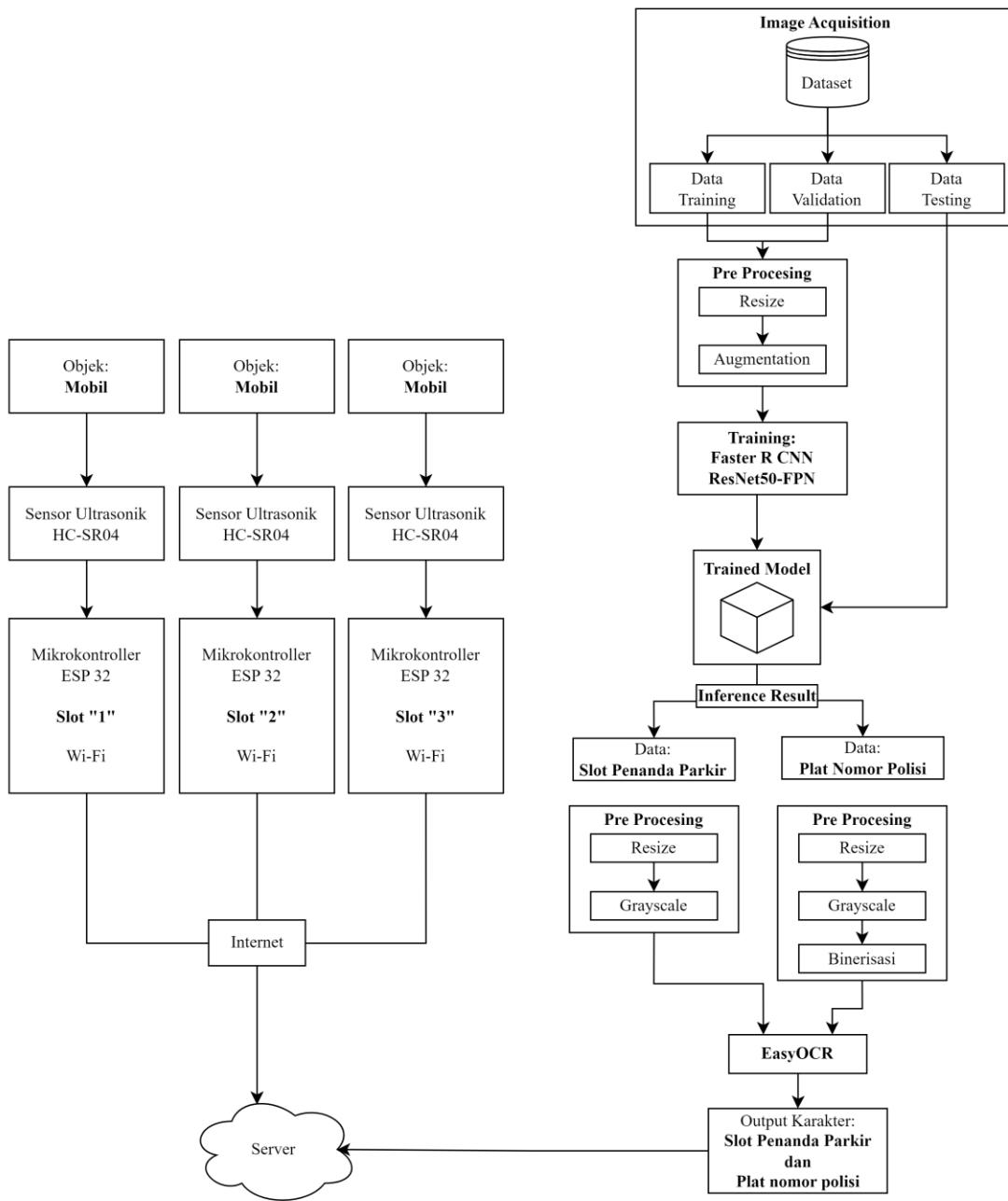
Setelah seluruh data dikumpulkan, jumlah total citra yang didapat adalah sebanyak 1.965 gambar. Rincian jumlah data adalah sebagai berikut:

**Tabel 3.1** Data Gambar

Data Gambar	Jumlah
<b>Plat Nomor Polisi</b>	1.782
<b>Kertas Penanda Parkir</b>	1.183
<b>Total</b>	2.965

## **3.2. Arsitektur Umum**

Untuk menyelesaikan aplikasi tersebut, perlu dilakukan tahapan-tahapan proses sebagai berikut dijelaskan pada Gambar 3.2.



**Gambar 3.2** Arsitektur Umum

Arsitektur umum ini berguna mengetahui rancangan proses dari cara kerja rangkaian secara garis besar. Keseluruhan arsitektur umum ini dengan gamblang menggambarkan bahwa sisi sebelah kiri merupakan tugas untuk mendeteksi keberadaan objek dalam penelitian ini, mobil. Pada bagian sebelah kanan, terdapat pemrosesan citra digital yang kemudian diekstraksi karakter pada objek hasil inferensi model.

### *3.2.1. Arsitektur Internet of Things (IoT)*

Sesuai dengan gambar, penulis akan mengembangkan suatu perangkat keras (*hardware*) yang dapat mendeteksi jarak. Sebagai pengukuran jarak, penelitian ini menggunakan sensor ultrasonik HC-SR04 yang mana sensor dapat memantulkan gelombang ultrasonik yang terpancar dari trigger, memantul menabrak objek yang ada di depannya dan kemudian diterima oleh *echo*. Penelitian ini, melakukan pengembangan dengan prototipe untuk 3 slot parkir. Masing masing slot parkir dibekali dengan sebuah mikrokontroler *ESP32* dan sensor ultrasonik yang terhubung pada pin yang tersedia pada *ESP32*. Ketiga mikrokontroler tersebut harus terhubung ke internet, guna mengirimkan data perubahan jarak ke *server Firebase Realtime Database*.

### *3.2.2. Image processing*

Hal kedua yang diperlukan untuk penelitian ini, yaitu pemrosesan citra digital yang berkaitan. Pada arsitektur umum, dijelaskan bahwa tahapan awal yaitu mengumpulkan data citra, kemudian data dibagi menjadi data latih, validasi dan pengujian, dan kemudian setelah model dapat mendeteksi kedua objek dengan baik di lapangan parkir pengujian. Model yang digunakan yaitu *Faster R-CNN*. Fungsi dari *image processing* ini, untuk mengenali citra plat nomor polisi dan kertas penanda slot parkir secara spesifik.

### *3.2.3. Optical character recognition*

Hasil inferensi dari model, dibagi menjadi 2, pemrosesan plat nomor polisi dan kertas penanda parkir. Kemudian hasil pendekripsi karakter dicetak menjadi string. Nilai hasil deteksi dapat digunakan dalam bentuk json ke server yang membutuhkannya.

## **3.3. Perancangan *Image Processing*: Plat Nomor Polisi dan Kertas Penanda Slot Parkir**

Pada bagian ini berguna untuk mendeteksi kemudian mengekstrak karakter pada teks penanda parkir dan plat nomor polisi untuk melakukan penyesuaian dengan data yang telah dipesan oleh pengguna aplikasi. Berikut merupakan arsitektur dalam menyelesaikan tahap *Image Processing*:

### *3.3.1. Image acquisition*

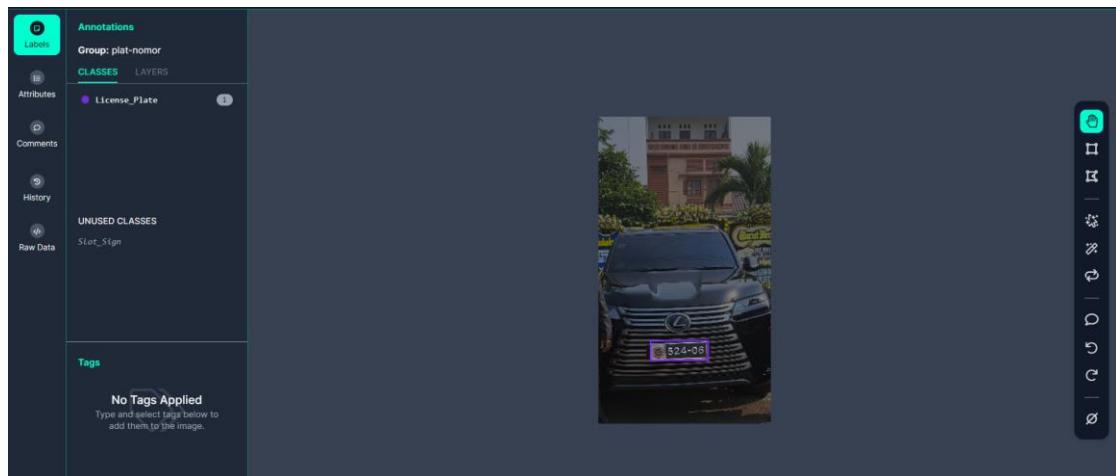
Pada tahap pertama ini dilakukan pengumpulan data yang akan dibutuhkan sesuai tujuan dari penelitian penulis. Penelitian ini akan menggunakan data plat nomor polisi dan kertas penanda parkir. Plat nomor polisi didapatkan dari laman Roboflow dan sebagian dari data pribadi. Kertas penanda parkir dibuat secara *custom* oleh penulis

kemudian dicetak menggunakan kertas A4. Setelah dicetak, penulis memotret penanda slot parkir, variasi pengambilan gambar dari banyak kemungkinan posisi.

Berdasarkan penggabungan data tersebut digunakan sebanyak 2.965 gambar yang berekstensi (.jpg). Pada tahap selanjutnya data akan dibagi menjadi 3 bagian yaitu data *train*, *validation* data *testing*.

### 3.3.2. Annotation

Anotasi gambar merupakan tahap menandai pada objek yang merupakan tujuan pelatihan atau klasifikasi guna mendapatkan hasil model deteksi plat nomor polisi dan kertas penanda slot parkir. Proses pelabelan model dilakukan pada roboflow dengan membuat *bounding box* pada setiap objek klasifikasi, seperti pada Gambar 3.3.



**Gambar 3.3** Anotasi pada Roboflow

### 3.3.3. Preprocessing

Tahap ini merupakan tahap awal pengolahan data citra yang telah dikumpulkan agar memberikan hasil yang baik pada tahap selanjutnya. Tahapan yang terdapat dalam ini meliputi:

#### 1. Resizing

Gambar-gambar dapat memiliki ukuran yang berbeda-beda. Dalam tahap ini, gambar-gambar tersebut diubah ukurannya menjadi ukuran 640 x 640 piksel yang seragam agar dapat diproses lebih lanjut dengan lebih efisien. Perbedaan bentuk sesudah dan sebelum dilakukannya dapat dilihat pada Gambar 3.4.

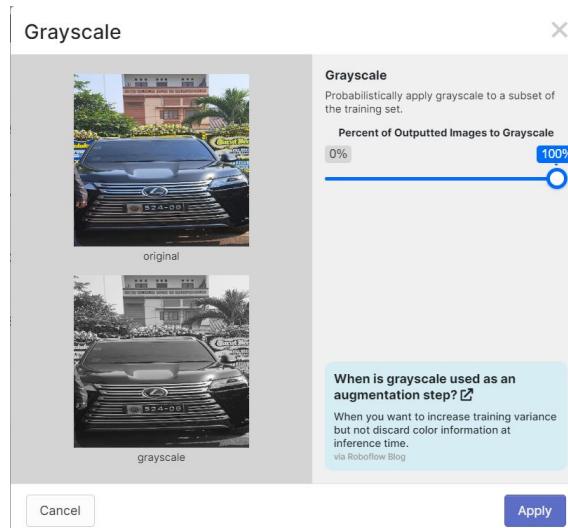


**Gambar 3.4 Resize**

#### 3.3.4. Augmentasi

Augmentasi digunakan untuk meningkatkan variasi data yang digunakan dalam pelatihan model. Ini bisa mencakup teknik seperti rotasi, pergeseran atau perubahan warna pada gambar untuk membuat model lebih *robust* terhadap variasi dalam data.

##### 1. Grayscale

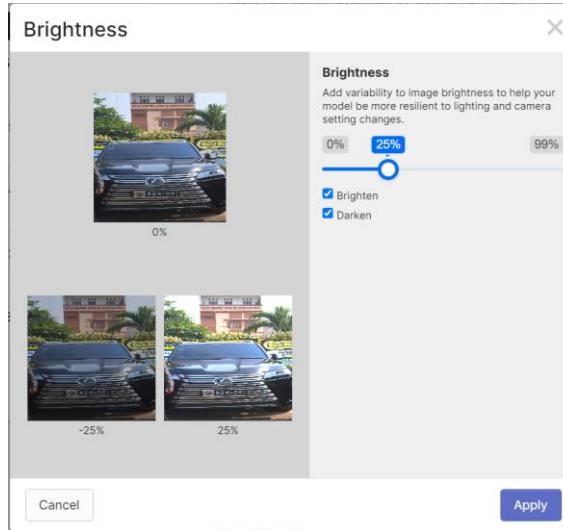


**Gambar 3.5 Grayscale**

##### 2. Brightness

Brightness merupakan teknik augmentasi untuk mengubah tingkat pencahayaan pada citra sesuai dengan rasio yang ditentukan. Penggunaan augmentasi ini didasari oleh kemungkinan model perlu mengenali citra pada kondisi perbedaan pencahayaan saat

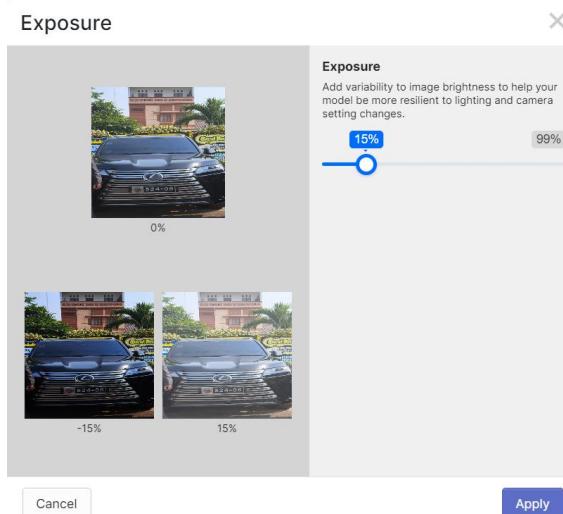
citra diambil untuk diproses. Pada Roboflow, penambahan citra kecerahan dapat dilihat pada Gambar 3.6.



**Gambar 3.6 Brightness**

### 3. Exposure

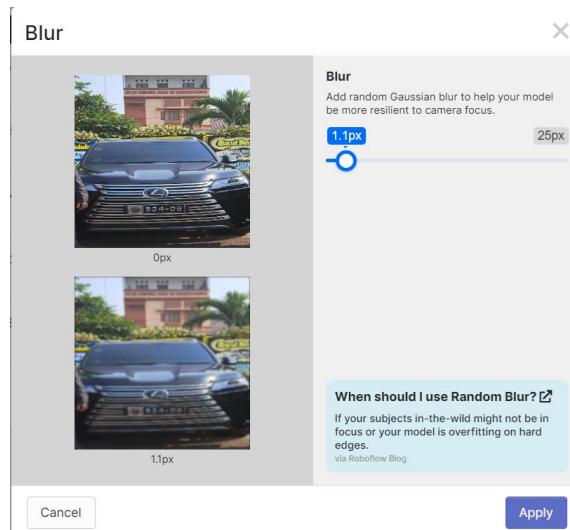
*Exposure* merupakan teknik merubah tingkat sorotan pencahayaan pada gambar sesuai dengan rasio yang diperlukan penelitian. Dengan *exposure* memungkinkan model untuk mempelajari kemungkinan terjadinya perbedaan tingkat pencahayaan pada proses pengambilan dan pengolahan data. Teknik *exposure* pada roboflow dapat dilihat pada Gambar 3.7.



**Gambar 3.7 Exposure**

#### 4. *Blur*

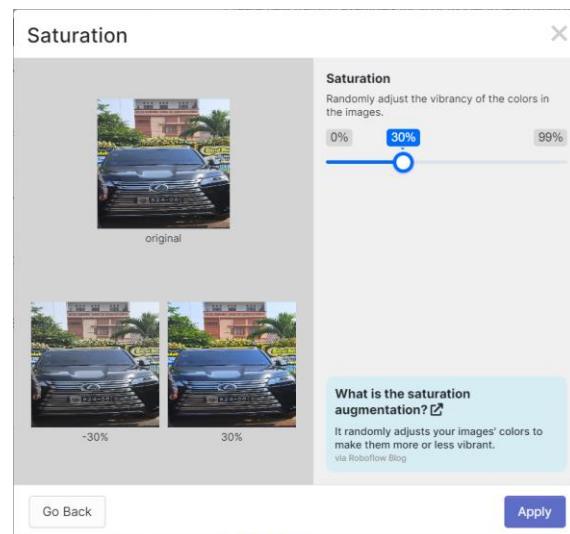
Teknik *blur* pada augmentasi merupakan teknik mengaburkan atau penyamaran citra. Teknik ini digunakan agar model dapat mendeteksi citra sekalipun pengguna mengambil gambar sedikit *blur*. Nilai *blur* dapat diatur sesuai pada Gambar 3.8.



**Gambar 3.8** *Blur*

#### 5. *Saturation*

Teknik ini mengubah intensitas warna pada gambar dengan tujuan membuat model dapat mengenali terhadap warna dan kondisi pencahayaan. Gambar 3.9 merupakan contoh teknik saturasi.

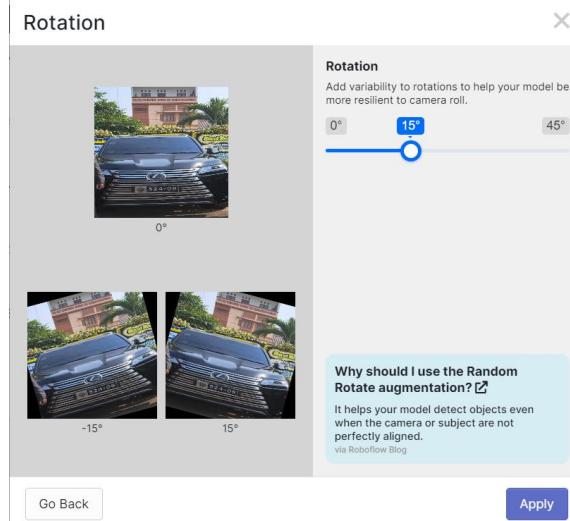


**Gambar 3.9** *Saturation*

#### 6. *Rotation*

Teknik ini melibatkan memutar gambar pada sudut tertentu. Yang bertujuan untuk membuat model lebih baik terhadap orientasi objek dalam gambar, sehingga dapat

mengenali objek meskipun posisinya berubah. Kemiringan dapat diatur sudut putarnya, pada Roboflow menyediakan kemiringan ke kiri dan kekanan seperti pada Gambar 3.10.



**Gambar 3.10 Rotation**

### 3.3.5. Eksport gambar dan anotasi

Setelah persiapan data selesai, melalui roboflow, penulis mengunduh seluruh berkas gambar dan berkas anotasi dalam bentuk (.xml) sesuai dengan kebutuhan untuk pelatihan. Tabel 3.2 merupakan tabel jumlah hasil pembagian data setelah proses anotasi gambar.

**Tabel 3.2 Jumlah Gambar**

Gambar	Train	Valid	Test
<b>Plat nomor polisi</b>	1.946	304	47
<b>Kertas penanda slot</b>	1.398	64	31

### 3.3.6. Feature extraction

Pada tahap ini, fitur-fitur penting dari gambar diekstraksi menggunakan model *Faster Region based Convolutional Neural Network (Faster R-CNN)* dengan *backbone ResNet50-FPN* menggunakan *Feature Pyramid Network* untuk menangani objek dari berbagai skala. Model ini untuk mendeteksi objek dalam gambar dan menentukan kotak pembatas (*bounding box*) yang mengelilingi objek tersebut. Tahap ini seluruh data yang telah dipisah menjadi data latih dan data validasi akan digunakan melalui proses pelatihan.

### 1. Convolution layer

*Convolutional layer* menggunakan filter untuk mengekstrak fitur yang terdapat pada objek agar dapat dikenali. Pada setiap posisi gambar, dihasilkan sebuah angka berupa *dot product* antara bagian gambar dengan filter yang digunakan. Dengan menggeser (*convolve*) filter pada gambar dihasilkan sebuah *activation map*. *Layer* ini kemudian membuat *feature map* dari objek yang sudah diambil. Adapun matriks gambar *input* pada penelitian ini adalah pada Gambar 3.11.

147	149	138	115	115
172	135	135	127	115
189	188	195	189	184
63	70	87	89	92
122	95	98	105	137

**Gambar 3.11** Matriks *Input*

Kemudian matriks dari *input* gambar dihitung dengan menggunakan *filter* yang terdapat pada *convolutional layer*. *Activation map* dihasilkan dari pengulangan proses perhitungan ini untuk setiap elemen dari gambar *input*. Adapun contoh salah satu perhitungan *filter* pada matriks gambar penelitian ini pada Gambar 3.12.

Nilai <i>Input</i>	Nilai <i>Filter</i>	Nilai Hasil																																		
<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td>147</td><td>149</td><td>138</td><td>115</td><td>115</td></tr> <tr><td>172</td><td>135</td><td>135</td><td>127</td><td>115</td></tr> <tr><td>189</td><td>188</td><td>195</td><td>189</td><td>184</td></tr> <tr><td>63</td><td>70</td><td>87</td><td>89</td><td>92</td></tr> <tr><td>122</td><td>95</td><td>98</td><td>105</td><td>137</td></tr> </table>	147	149	138	115	115	172	135	135	127	115	189	188	195	189	184	63	70	87	89	92	122	95	98	105	137	$\times \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} TL & TM & TR \\ L & M & R \\ BL & BM & BR \end{bmatrix}$	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr><td>TL</td><td>TM</td><td>TR</td></tr> <tr><td>L</td><td>M</td><td>R</td></tr> <tr><td>BL</td><td>BM</td><td>BR</td></tr> </table>	TL	TM	TR	L	M	R	BL	BM	BR
147	149	138	115	115																																
172	135	135	127	115																																
189	188	195	189	184																																
63	70	87	89	92																																
122	95	98	105	137																																
TL	TM	TR																																		
L	M	R																																		
BL	BM	BR																																		

**Gambar 3.12** Perhitungan Matriks *Input* dengan Matriks *Filter*

Pada *convolution layer* nilai matriks gambar *input* pada penelitian ini sebesar 5x5 piksel. Kemudian nilai tersebut akan dikalikan dengan nilai filter yaitu sebesar 3x3. Matriks gambar *input* bergeser sebesar 3x3 dari sudut kiri atas ke kanan dengan pergeseran 1 piksel sampai melewati semua piksel dari baris pertama. Pergeseran berlanjut sampai baris berikutnya hingga semua piksel pada gambar *input* dilalui dan

kemudian disimpan dalam matrix baru. Pergeseran matriks tersebut dapat dilihat di bawah ini.

147	149	138	115	115	147	149	138	115	115	147	149	138	115	115
172	135	135	127	115	172	135	135	127	115	172	135	135	127	115
189	188	195	189	184	189	188	195	189	184	189	188	195	189	184
63	70	87	89	92	63	70	87	89	92	63	70	87	89	92
122	95	98	105	137	122	95	98	105	137	122	95	98	105	137

147	149	138	115	115	147	149	138	115	115	147	149	138	115	115
172	135	135	127	115	172	135	135	127	115	172	135	135	127	115
189	188	195	189	184	189	188	195	189	184	189	188	195	189	184
63	70	87	89	92	63	70	87	89	92	63	70	87	89	92
122	95	98	105	137	122	95	98	105	137	122	95	98	105	137

147	149	138	115	115	147	149	138	115	115	147	149	138	115	115
172	135	135	127	115	172	135	135	127	115	172	135	135	127	115
189	188	195	189	184	189	188	195	189	184	189	188	195	189	184
63	70	87	89	92	63	70	87	89	92	63	70	87	89	92
122	95	98	105	137	122	95	98	105	137	122	95	98	105	137

**Gambar 3.13** Perhitungan Pergeseran Matriks

Kemudian hasil perkalian antara matriks deteksi fitur atau *filter* dan matriks pada input menjadi nilai dari *feature map*. Berikut merupakan perhitungan *convolutional layer* pada penelitian ini:

$$\begin{aligned}
 TL &= (147 \times 0) + (149 \times 1) + (138 \times 0) + (172 \times 1) + (135 \times 0) + (135 \times 1) + (189 \times 0) + (188 \times 1) + (195 \times 0) \\
 &= 0 + 194 + 0 + 172 + 0 + 135 + 0 + 188 + 0 \\
 &= \mathbf{689}
 \end{aligned}$$

$$\begin{aligned}
 TM &= (149 \times 0) + (138 \times 1) + (115 \times 0) + (135 \times 1) + (135 \times 0) + (127 \times 1) + (188 \times 0) + (195 \times 1) + (189 \times 0) \\
 &= 0 + 138 + 0 + 135 + 0 + 127 + 0 + 195 + 0 \\
 &= \mathbf{595}
 \end{aligned}$$

$$\begin{aligned}
 TR &= (138X0) + (115X1) + (115X0) + (135X1) + (127X0) + (115X1) + (195X0) + (189X1) + (184X0) \\
 &= 0 + 115 + 0 + 135 + 0 + 115 + 0 + 189 + 0 \\
 &= \mathbf{554} \\
 L &= (172X0) + (135X1) + (135X0) + (189X1) + (188X0) + (195X1) + (63X0) + (70X1) + (87X0) \\
 &= 0 + 135 + 0 + 189 + 0 + 195 + 0 + 70 + 0 \\
 &= \mathbf{589} \\
 M &= (135X0) + (135X1) + (127X0) + (188X1) + (195X0) + (189X1) + (70X0) + (87X1) + (89X0) \\
 &= 0 + 135 + 0 + 188 + 0 + 189 + 0 + 87 + 0 \\
 &= \mathbf{599} \\
 R &= (135X0) + (127X1) + (115X0) + (195X1) + (189X0) + (184X1) + (87X0) + (89X1) + (92X0) \\
 &= 0 + 127 + 0 + 195 + 0 + 184 + 0 + 89 + 0 \\
 &= \mathbf{595} \\
 BL &= (189X0) + (188X1) + (195X0) + (63X1) + (70X0) + (87X1) + (122X0) + (95X1) + (98X0) \\
 &= 0 + 188 + 0 + 63 + 0 + 87 + 0 + 95 + 0 \\
 &= \mathbf{433} \\
 BM &= (188X0) + (195X1) + (189X0) + (70X1) + (87X0) + (89X1) + (95X0) + (98X1) + (105X0) \\
 &= 0 + 195 + 0 + 70 + 0 + 89 + 0 + 98 + 0 \\
 &= \mathbf{452} \\
 BR &= (195X0) + (189X1) + (184X0) + (87X1) + (89X0) + (92X1) + (98X0) + (105X1) + (137X0) \\
 &= 0 + 189 + 0 + 87 + 0 + 92 + 0 + 105 + 0 \\
 &= \mathbf{473}
 \end{aligned}$$

Dengan perhitungan tersebut, maka didapatkan contoh hasil matriks penelitian ini yang sudah diproses pada *convolutional layer* yaitu sebagai berikut:

689	595	554
589	599	595
433	452	473

**Gambar 3.14** Matriks *Convolution layer*

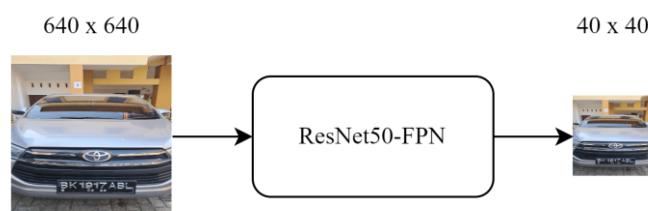
## 2. Feature map

*Feature Map* merupakan *map* yang dibuat oleh *convolution layer* yang berisi informasi tentang representasi vektor dari citra. *Convolution layer* menghasilkan dua *feature map* yang sama. *Feature map* pertama diolah *RPN* untuk menghasilkan *region proposal*, *feature map* lainnya langsung dikirim ke *pooling layer*. Gambar ditampilkan sebagai *height*, *width*, dan *depth* diteruskan ke lapisan tengah menggunakan *CNN* yang sebelumnya sudah dilatih dan berakhir sebagai *convolutional feature map*. Adapun contoh ukuran matriks yang didapat dari proses konvolusi sebelumnya adalah sebagai berikut.

689	595	554
589	599	595
433	452	473

**Gambar 3.15 Matriks Feature map**

Pada *feature map*, dilakukan perubahan ukuran dimensi gambar. Ukuran dimensi gambar *feature map* pada awalnya adalah  $640 \times 640 \times 3$ . Kemudian ukuran tersebut diubah dan diperkecil menjadi beberapa ukuran 320, 160, 80, dan 40 menggunakan model yang telah dilatih sebelumnya (*pretrained model*) dengan ResNet50-FPN. Gambar 3.16 merupakan visualisasi dari *feature map*.



**Gambar 3.16 Feature Map**

*Layer* dengan beberapa ukuran berguna agar model dapat dilatih dengan kemungkinan ukuran besar, sedang dan kecil.

### 3. Region Proposal Network (RPN)

Setelah melewati *feature map*, kemudian citra diproses oleh RPN dan dibuat dalam *convolutional layer* untuk memprediksi apa yang dianggap sebagai objek dan membuat prediksi *bounding box* objek. RPN dibagi menjadi 2 *convolutional layer* dimana 1 layer bertanggung jawab untuk mendeteksi letak objek dan 1 layer memprediksi *bounding box*. RPN akan menilai objek pada suatu citra dan membuat nilai objek menjadi kotak pembatas output. Titik 40x40 pada *feature map* sebelumnya dinilai sebagai *anchors*. Maka penentuan rasio dan ukuran secara spesifik diperlukan untuk setiap *anchors*, yaitu 1:1, 1:2, 2:1 untuk tiga rasio dan 128%, 256%, 512% untuk tiga ukuran pada citra. Ukuran anchors yaitu:  $3 \times 3 = 9$  kotak. Ukuran anchors pada citra yaitu:  $40 \times 40 \times 9 = 14.400$  kotak.

Karena ukuran *anchors* pada citra terlalu besar maka akan dipilih sebanyak 256 kotak sebagai mini batch. Gambar 3.17 merupakan visualisasi dari RPN. Pada gambar berikut, kotak berwarna hijau adalah *anchors* yang menilai citra sebagai objek dan kotak berwarna biru merupakan *anchors* yang menilai objek yang disorot bukan

merupakan objek yang akan dideteksi. Hasil dari proses deteksi ini akan mendeteksi apakah citra yang disorot merupakan objek atau bukan.



**Gambar 3.17 Region Proposal Network**

#### 4. Region of Interest Pooling (RoI Pooling)

*Region of interest* merupakan wilayah yang direkomendasikan dari gambar asli. Sebelum dideteksi, objek harus melewati proses ini. *ROI pooling* menyamakan ukuran dari *feature map* dan *region proposal* yang telah diolah oleh *RPN*. *ROI Pooling* kemudian mengirim informasi *feature map* dan *proposal* untuk dideteksi. Setelah tahap *RPN* maka selanjutnya akan terhubung ke *convolutional layer* dengan *filter ROI pooling* yaitu  $3 \times 3 \times 640$  piksel. Adapun gambar target kotak *RoI* untuk kelas plat nomor polisi terdapat pada Gambar 3.18.

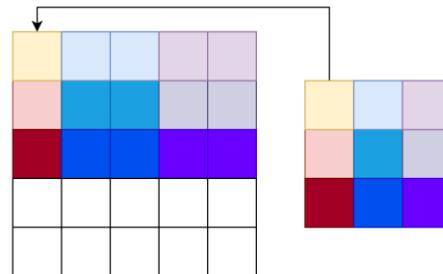


**Gambar 3.18 Target Region of Interest Pooling**

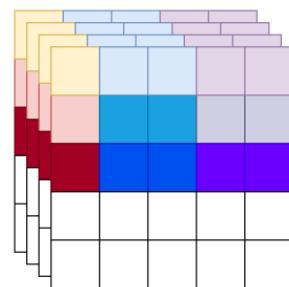
Gambar tersebut merupakan objek yang ditargetkan dengan ukuran kotak. Adapun proses *RoI Pooling* tersebut pada Gambar 3.19.

Setelah itu, ukuran *RoI* akan dipooling agar dapat masuk kedalam *fully connected layer*. Pada proses penggabungan citra, dari nilai tertinggi dari suatu matriks saja yang akan diambil sehingga ukuran awal yang beragam berubah menjadi satu

ukuran untuk keseluruhan data. Proses pengubahan data tersebut dilakukan pada seluruh lapisan matriks *RoI*. Adapun ilustrasi dari prosesnya terdapat pada Gambar 3.20.



**Gambar 3.19** Tahapan *RoI Pooling*



**Gambar 3.20** *RoI Pooling*

Setiap matriks dari hasil *RoI Pooling* akan dikirim melalui seluruh jaringan untuk masing-masing sehingga menghasilkan kotak pembatas terhadap objek.

### 3.3.7. Trained model

Model yang telah dilatih menggunakan data training digunakan untuk mendeteksi plat nomor polisi dan kertas penanda slot parkir dalam gambar. Model ini telah dipelajari untuk mengenali pola dan fitur-fitur yang mengidentifikasi plat nomor polisi dan ketas penanda slot parkir.

### 3.3.8. Output inference

Hasil dari inferensi gambar menggunakan model yang telah dilatih dibagi menjadi 2, yaitu kertas penanda slot parkir dan plat nomor polisi.

## 3.4. Perancangan *Easy-Optical Character Recognition (EasyOCR)*

Setelah plat nomor polisi dan teks penanda parkir terdeteksi, teknologi *Easy-Optical Character Recognition (EasyOCR)* digunakan untuk melakukan pengenalan karakter pada area yang terdeteksi. *EasyOCR* merupakan perangkat lunak *open-source* yang digunakan untuk mengenali dan mengekstrak teks dari gambar.

### 3.4.1. Pre-Processing: EasyOCR

Sebelum mengekstrak karakter, perlu pre-processing pada gambar hasil inferensi dari model yang telah dideteksi untuk dapat memastikan konsisten dalam pendekripsi karakter. Tahapan preprocessing untuk gambar plat nomor polisi dapat dijelaskan sebagai berikut:

#### 1. Resize

Mengubah ukuran hasil inferensi menjadi 300x600 piksel ini berguna untuk membuat konsisten dalam posisi portrait untuk kertas penanda slot parkir dan 350x150 piksel untuk plat nomor polisi untuk mempertegas posisi horizontal dan lebih kecil agar apabila plat yang terdeteksi jarak antar karakter relatif jauh, maka EasyOCR dapat mendekripsi karakter dalam 1 sub-array.

#### 2. Grayscale

*Grayscale* mengubah gambar berwarna (BGR) menjadi skala abu-abu.

#### 3. Binerisasi thresholding

Tahapan pra pemrosesan thresholding dapat mengubah gambar grayscale menjadi gambar biner (hitam-putih) menggunakan metode thresholding Otsu. Pra pemrosesan ini hanya untuk gambar plat nomor polisi agar pendekripsi karakter lebih optimal. Perbedaan pra pemrosesan ekstraksi karakter dapat dilihat pada Gambar 3.21 dan Gambar 3.22.

```
def preprocess_image_slot(image):
    image_resize = cv2.resize(image, (300, 600))
    image_gray = cv2.cvtColor(image_resize, cv2.COLOR_BGR2GRAY)

    return image_gray
```

**Gambar 3.21** Preprocess Gembat Slot Sign

```
def preprocess_image_plate(image):
    image_resize = cv2.resize(image, (300, 150))
    image_gray = cv2.cvtColor(image_resize, cv2.COLOR_BGR2GRAY)
    _, image_bw = cv2.threshold(image_gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

    return image_bw
```

**Gambar 3.22** Preprocess Gambar Plat Mobil

### 3.4.2. Output

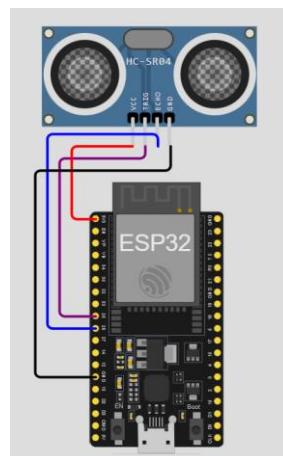
Hasil dari proses pengenalan karakter menggunakan *EasyOCR*, setelah melakukan pra pemrosesan gambar adalah karakter dari plat nomor polisi dan karakter dari kertas penanda parkir yang telah terdeteksi dan dienkripsi dalam teks.

### 3.5. Perancangan Perangkat Keras

Sistem identifikasi keberadaan mobil pada setiap slot dirancang menggunakan perangkat keras yang sesuai pada Gambar 3.23. Perancangan menunjukkan hubungan antara perangkat yang digunakan. Sumber listrik dari perangkat ini menggunakan sumber daya dari *USB*.

**Tabel 3.3** Daftar Kebutuhan Perangkat Keras

Nama Perangkat	Banyak	Fungsi
<i>ESP32</i>	3	Mikrokontroler
Sensor Ultrasonik HC-SR04	3	Mengukur jarak objek



**Gambar 3.23** Konfigurasi Perangkat Keras

Pada Tabel 3.4 merupakan rancangan konfigurasi pin dari sensor ultrasonik ke mikrokontroler *ESP32*

**Tabel 3.4** Konfigurasi Pin Perangkat Keras

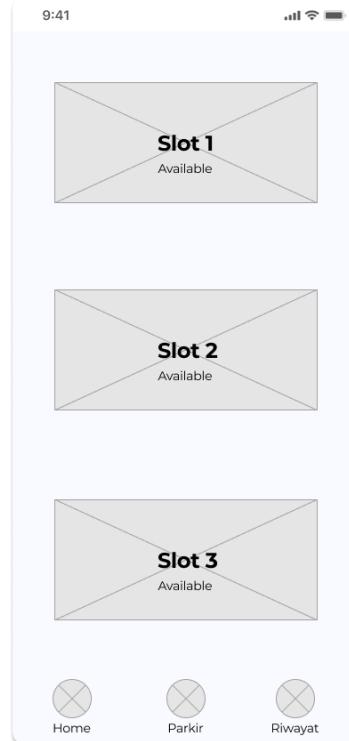
No	Sensor Ultrasonik	ESP32
1.	<i>VCC</i>	3.3V
2.	<i>GND</i>	<i>GND</i>
3.	<i>Trigger</i>	<i>GPIO25</i>
4.	<i>Echo</i>	<i>GPIO26</i>

### 3.6. Perancangan Aplikasi Android

#### 3.6.1. Sisi klien (*frontend*)

Perancangan antarmuka sebagai implementasi agar pengguna dapat berinteraksi langsung dengan fungsi yang tersedia. Sisi klien yang dibangun dengan kerangka *React Native*.

## 1. Rancangan Visualisasi Slot Parkir



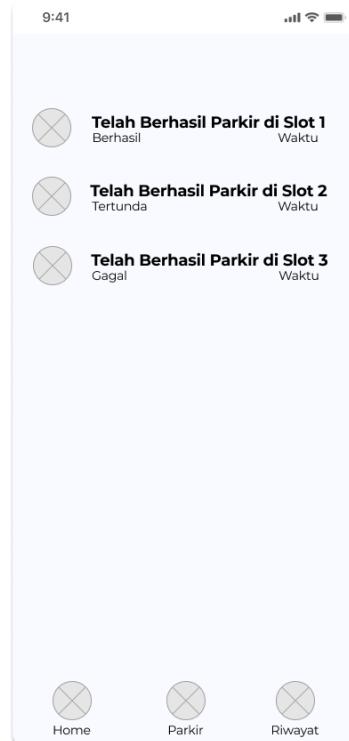
**Gambar 3.24** Perancangan Visualisasi Slot Parkir

## 2. Rancangan Halaman Beranda



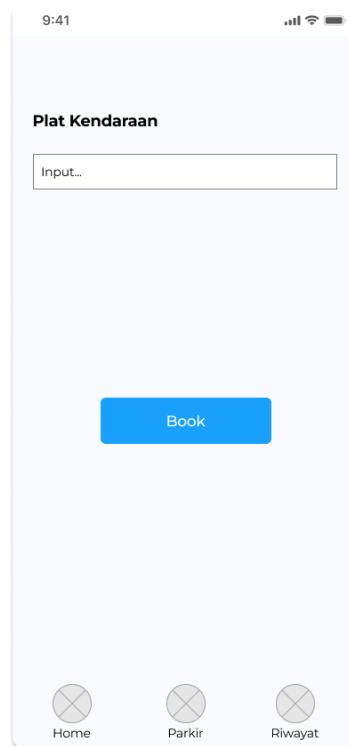
**Gambar 3.25** Perancangan Halaman beranda

### 3. Rancangan Halaman Riwayat



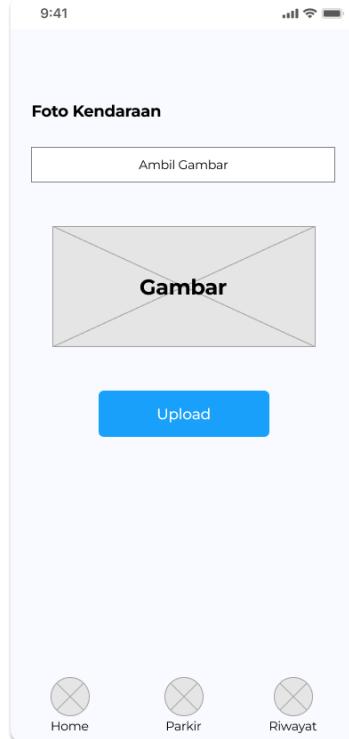
**Gambar 3.26** Perancangan Halaman Riwayat

### 4. Rancangan Halaman *Input* Plat Nomor Polisi



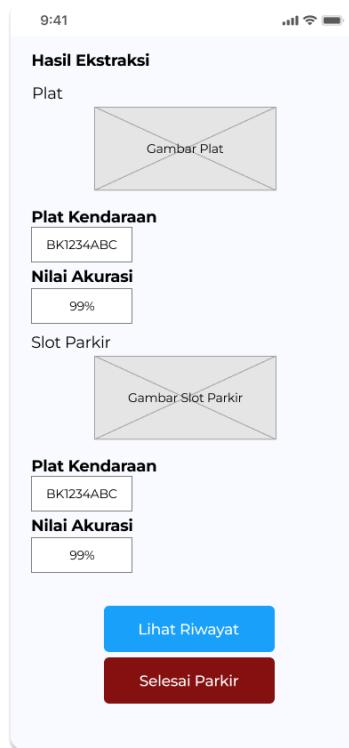
**Gambar 3.27** Perancangan Halaman *Input* Plat Nomor Polisi

## 5. Rancangan Halaman Unggah Foto



**Gambar 3.28** Perancangan Halaman Unggah Foto

## 6. Rancangan Halaman Hasil

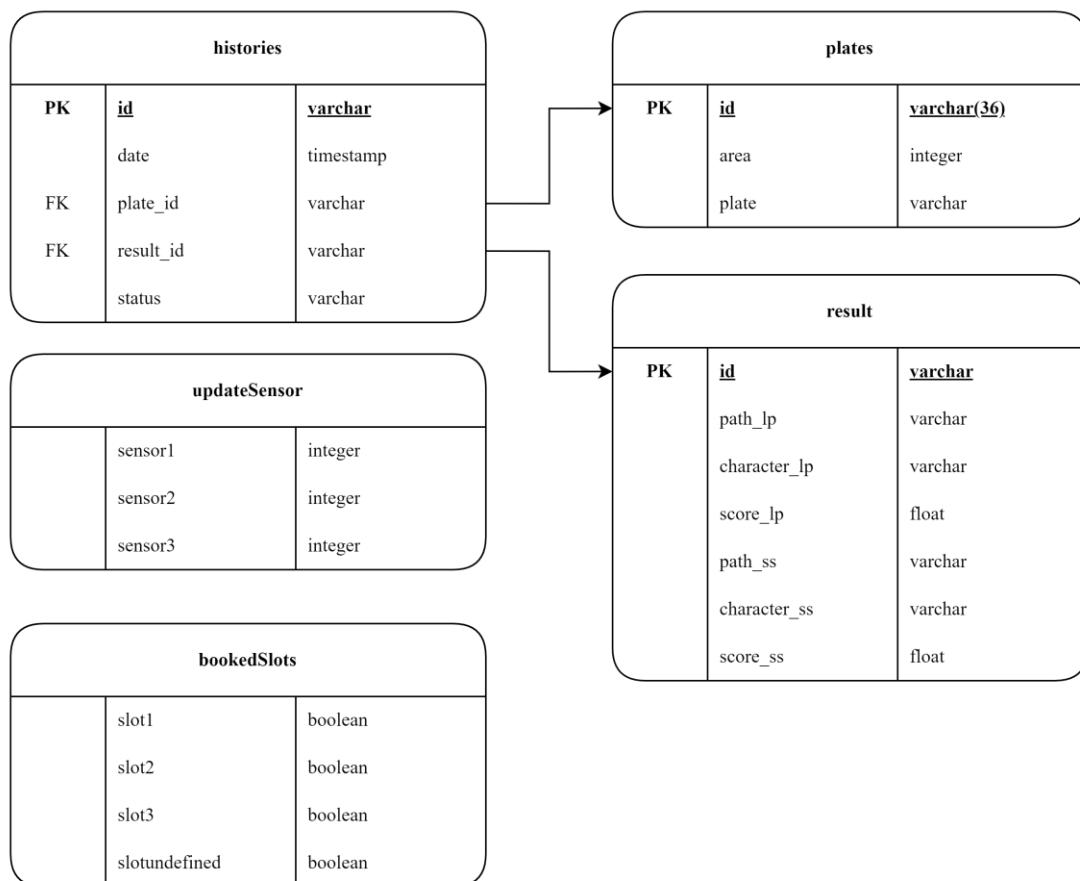


**Gambar 3.29** Perancangan Halaman Hasil

### 3.6.2. Sisi server (backend)

Sisi *server* merujuk pada bagian dari aplikasi yang berjalan pada sisi *server*. Ini mencakup logika bisnis, basis data dan *server* itu sendiri. Sisi *server* dibangun dengan menggunakan *framework ExpressJS*. Seluruh penyimpanan data yang diperlukan disimpan pada *realtime database* Firebase milik Google.

#### 1. Perancangan Entity Relation Diagram



**Gambar 3.30 Entity Relation Diagram**

Berikut merupakan daftar kebutuhan dalam proses bisnis pada aplikasi:

1. Validasi
2. Mengambil seluruh daftar riwayat.
3. Mengambil informasi detail dari salah satu riwayat
4. Menambahkan riwayat baru
5. Menambahkan data plat baru
6. Mengambil seluruh daftar plat
7. Mengambil informasi detail dari salah satu plat
8. Mengubah status dari salah satu slot.

## **BAB 4**

### **IMPLEMENTASI DAN PENGUJIAN SISTEM**

#### **4.1. Implementasi Sistem**

Komponen perangkat lunak dan perangkat keras yang ditanam dalam komputer personal (*Personal Computer*) yang digunakan dalam penelitian tugas akhir dengan spesifikasi sebagai berikut:

Nama Perangkat : Asus TUF Gaming F15 FX506HCB  
Prosesor : 11<sup>th</sup> Gen Intel® Core™ i7-11600H @ 2.90GHz  
Sistem Operasi : Windows 11 Home Single Language 64-bit  
RAM : 8 GB  
Kapasitas SSD : 512 GB

Perangkat Lunak yang digunakan dalam mengembangkan aplikasi berbasis *mobile* integrasi antara *Internet of Things* dan *Image Processing*

1. Microsoft Visual Studio Code
2. Arduino IDE
3. Google Colab
4. Jupyter Notebook
5. Python versi 3.9.2, torch versi 2.3.0, torchvision versi 0.13.0, protobuf versi 3.20.1 EasyOCR versi 1.7.1, Flask versi 3.0.3, pip versi 24.0,
6. React versi 18.2.0, react-native versi 0.72.6, socket.io versi 4.7.5, firebase versi 10.11.1, axios versi 1.6.8, android versi 0.0.8
7. ExpressJS versi 4.19.2, firebase versi 10.11.1, axios versi 1.6.8

#### **4.2. Implementasi Perancangan Aplikasi**

##### *4.2.1. Sisi klien (Frontend)*

Pada bagian ini hasil dari pengkodean menggunakan kerangka kerja *React Native*.

### 1. *Splash Screen*

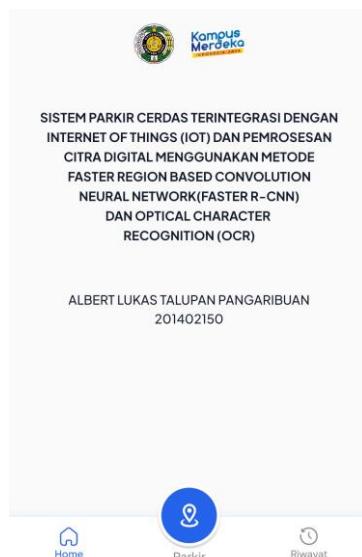
Tampilan *Splash Screen* dirancang untuk memperkenalkan nama aplikasi. Tampilan ini muncul pada awal aplikasi di-tap sebelum halaman utama ditampilkan.



**Gambar 4.1** *Splash Screen*

### 2. *Home*

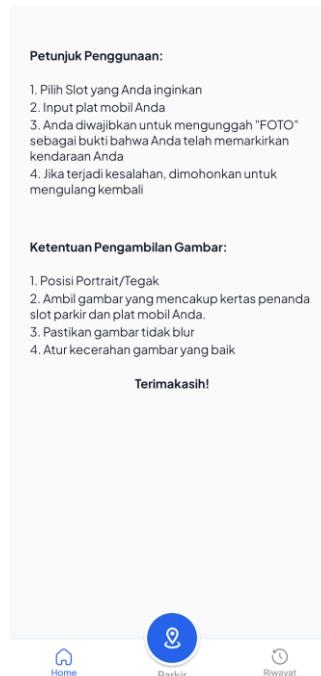
Tampilan *Home* merupakan tampilan yang menerangkan judul skripsi, identitas penulis dan logo Kampus Universitas Sumatera Utara serta Kampus Merdeka.



**Gambar 4.2** Halaman *Home*

### 3. Petunjuk aplikasi

Pada halaman ini menjelaskan petunjuk penggunaan dan ketentuan pengambilan Gambar 4.3.



**Gambar 4.3** Halaman Petunjuk Aplikasi

### 4. Visual slot parkir

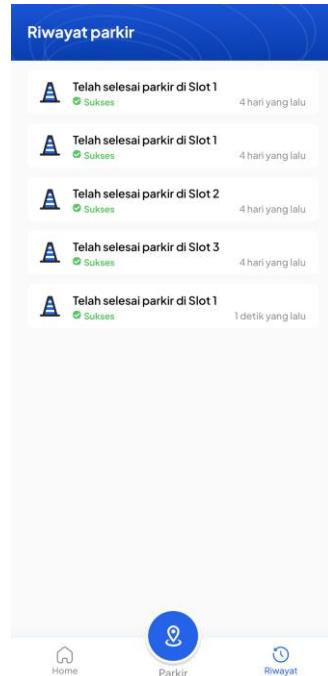
Pada halaman utama terdapat visualisasi kreatif penulis yang menggambarkan posisi situasi slot parkir di lapangan.



**Gambar 4.4** Halaman Visualisasi Slot Parkir

## 5. Riwayat

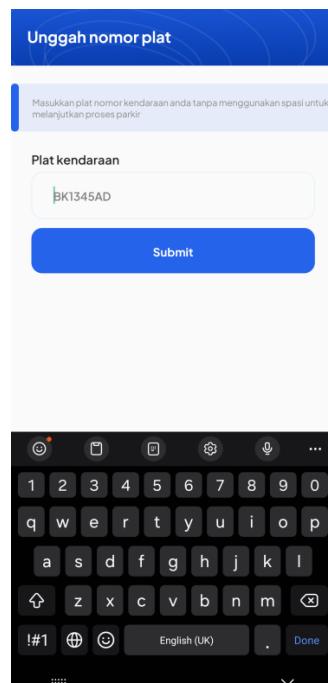
Pada tampilan riwayat, untuk menampilkan seluruh daftar penggunaan pemesanan slot parkir, status dan kewaktuan.



**Gambar 4.5** Halaman Riwayat

## 6. Input Plat Pengguna

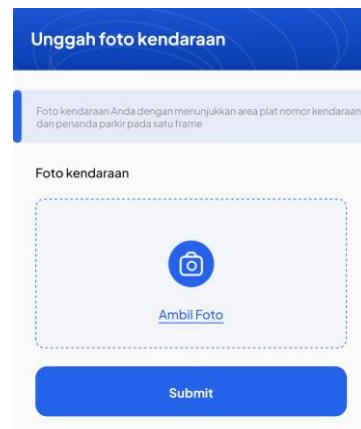
Pada halaman ini, pengguna mengisi plat nomor polisi pada formulir yang disediakan.



**Gambar 4.6** Halaman Input Plat

## 7. Unggah Gambar

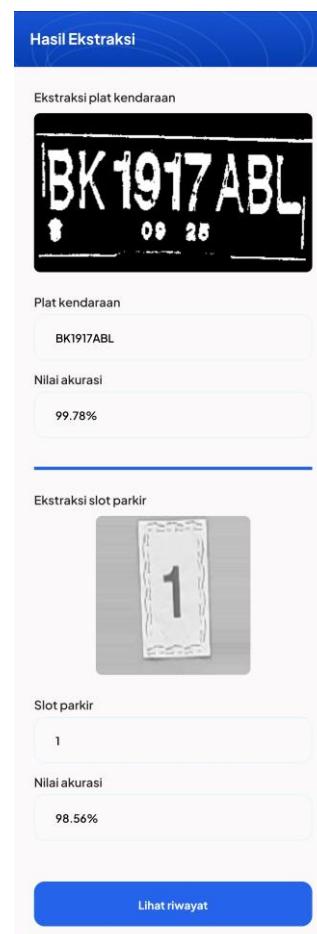
Halaman unggah foto, pengguna dapat memotret dengan menekan tombol ambil foto kemudian tekan tombol “*Submit*”.



**Gambar 4.7 Halaman *Upload* Gambar**

## 8. Hasil

Setelah pengunggahan gambar, pengguna dapat melihat hasil pada halaman Hasil Ekstraksi.



**Gambar 4.8 Halaman Hasil Inferensi dan Hasil Karakter**

## 9. Pesan Berhasil

Berikut pada Gambar 4.9, merupakan pesan berhasil



**Gambar 4.9** Pesan Berhasil

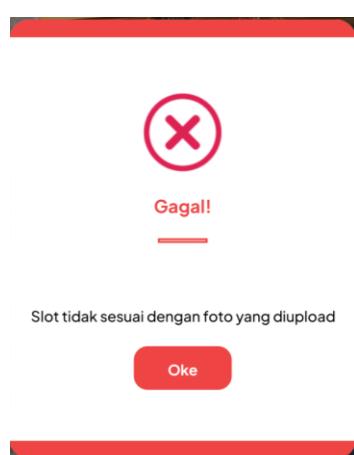
## 10. Pesan Kesalahan

1. Jika Slot tidak terdeteksi ditampilkan pada Gambar 4.10.



**Gambar 4.10** Pesan Kesalahan Slot Tidak Terdeteksi

2. Jika Slot tidak sesuai dengan foto yang diupload



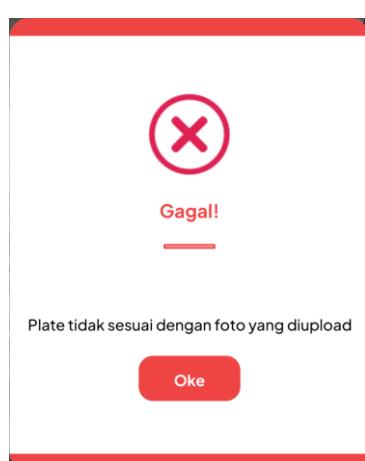
**Gambar 4.11** Pesan Kesalahan Slot Tidak Sesuai

3. Jika Plat tidak terdeteksi



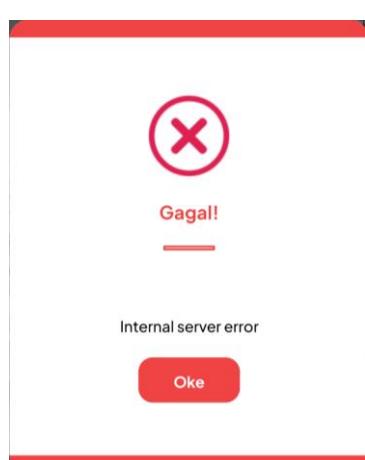
**Gambar 4.12** Pesan Kesalahan Plat Tidak Terdeteksi

4. Jika Plat tidak sesuai dengan yang diupload



**Gambar 4.13** Pesan Kesalahan Plat Tidak sesuai

5. Lainnya



**Gambar 4.14** Pesan Kesalahan Lainnya

### 11. Animasi *Loading*: Proses Validasi

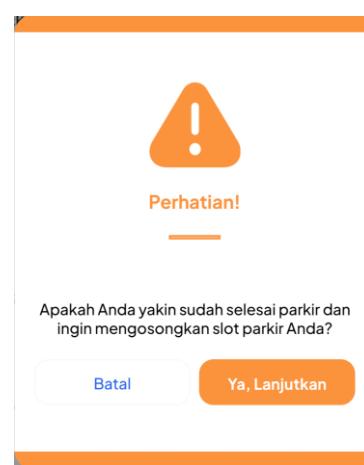
Pada halaman ini merupakan animasi .GIF untuk menunggu hasil inferensi dan ekstraksi karakter yang sedang dilakukan oleh sistem.



**Gambar 4.15** Halaman Animasi *Loading*

### 12. Pesan Konfirmasi

Ini merupakan pesan konfirmasi sebelum menyelesaikan penggunaan slot parkir.



**Gambar 4.16** Pesan Konfirmasi

#### 4.2.2. Sisi server (Backend)

Pada bagian ini merupakan implementasi pengkodean untuk pengiriman, penyimpanan, dan mengambil data dari *database*.

##### 1. Get Flask

Salah satu penggunaan untuk *backend* yaitu menangkap *server flask* pada localhost:5000.

##### 2. Validasi

Pada fungsi ini merupakan fungsi utama menangkap hasil dari ekstraksi karakter dari server flask yang memberikan data, kemudian mengambil kembali data *input-an* user dari tabel “*plates*” berdasarkan id untuk dibandingkan terdapat fungsi (*getPlatebyId*). Jika sesuai, maka fungsi ini akan mengirimkan status 200 dan data ke *frontend* untuk dapat dilihat oleh pengguna dengan antarmuka.

##### 3. Mengambil seluruh daftar riwayat (*getHistories*)

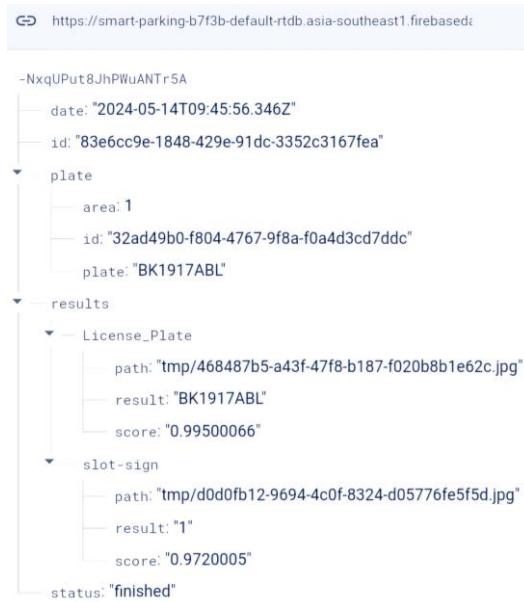
Pada fungsi ini, digunakan untuk mengambil nilai yang ada dari *database*. Berikut merupakan contoh data-data riwayat yang ada pada database.



**Gambar 4.17** Firebase: Daftar Riwayat

##### 4. Mengambil informasi detail dari salah satu *history* (*getHistory*)

Fungsi ini untuk menampilkan halaman detail. Berikut merupakan contoh data yang diperlukan untuk tampil pada halaman detail.



**Gambar 4.18** Firebase: Data Detail Riwayat

#### 5. Menambahkan data plat baru (*storePlate*)

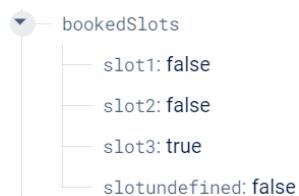
Fungsi ini digunakan untuk menyimpan data dari pengguna yang telah memilih slot dan mengisi formulir data plat nomor polisi pengguna. Setiap pengguna memesan slot parkir maka akan ditambahkan pada tabel “*plates*” data dari pengguna seperti ditunjukkan pada Gambar 4.19.



**Gambar 4.19** Firebase: Plat berdasarkan id

#### 6. Mengubah status dari salah satu slot (*updateSlotAreaFromDB*)

Penggunaan fungsi ini untuk membatasi nilai yang selalu berubah pada sensor. Dalam arti jika suatu slot telah dipesan pada aplikasi maka nilai sensor *realtime* tidak diperlukan sampai pada pengguna telah menyelesaikan pemarkiran pada aplikasi.



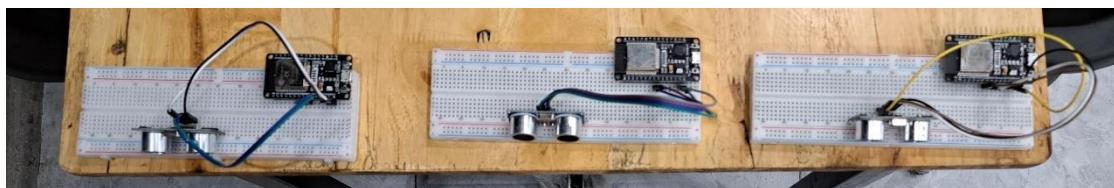
**Gambar 4.20** Status Slot

seperti pada gambar 4.20, jika bernilai true maka aplikasi menerima *realtime* dari sensor, sebaliknya jika bernilai *false* maka status pada slot tersebut sedang dipesan maka pengguna lain tidak dapat memesan slot tersebut.

### 4.3. Implementasi Perangkat Keras

#### 4.3.1. Rangkaian alat pendetksi mobil pada setiap slot

Pemantauan keberadaan mobil pada setiap slot parkir memerlukan sistem yang dibangun. Rangkaian perangkat pendetksi disusun sedemikian rupa agar dapat berfungsi dengan layak dan dapat bekerja optimal. Adapun perangkat yang digunakan yaitu *ESP32* dan sensor ultrasonik. Dalam konteks pengembangan penulis menggunakan *Breadboard*. Sebagai penghantar arus menggunakan kabel *jumper male to male*. Gambar 4.21 merupakan susunan rangkaian yang diperuntukkan tiga slot parkir.

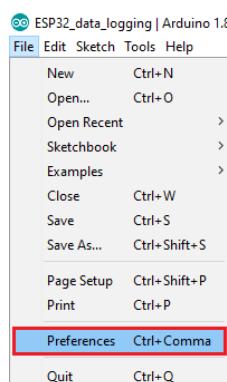


**Gambar 4.21** Rangkaian Perangkat Keras Setiap Slot

#### 4.3.2. Rangkaian penginstalan paket *ESP32*

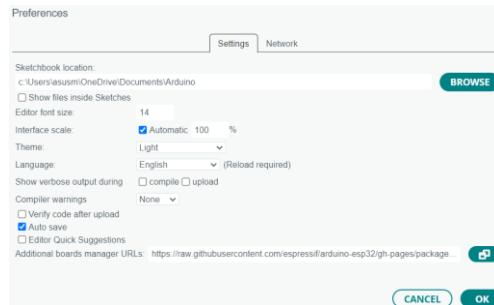
Sebelum melakukan pengkodean pada *ESP32*, perlu melakukan langkah untuk instalasi papan pengembang khusus *ESP32* membuat menu *Preference* yang ada pada *Arduino IDE* dengan menambahkan paket. Langkah instalasi untuk itu sebagai berikut.

1. Klik *File → Preferences*



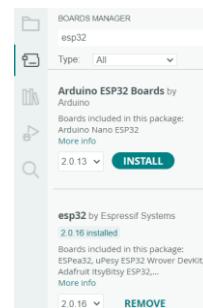
**Gambar 4.22** Preferences *Arduino IDE*

2. Tambahkan *link* pada kolom “Additional Board Manager URLs”:  
[https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package\\_esp32\\_index.json](https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json)
3. Setelah menambahkan *link* paket, klik OK



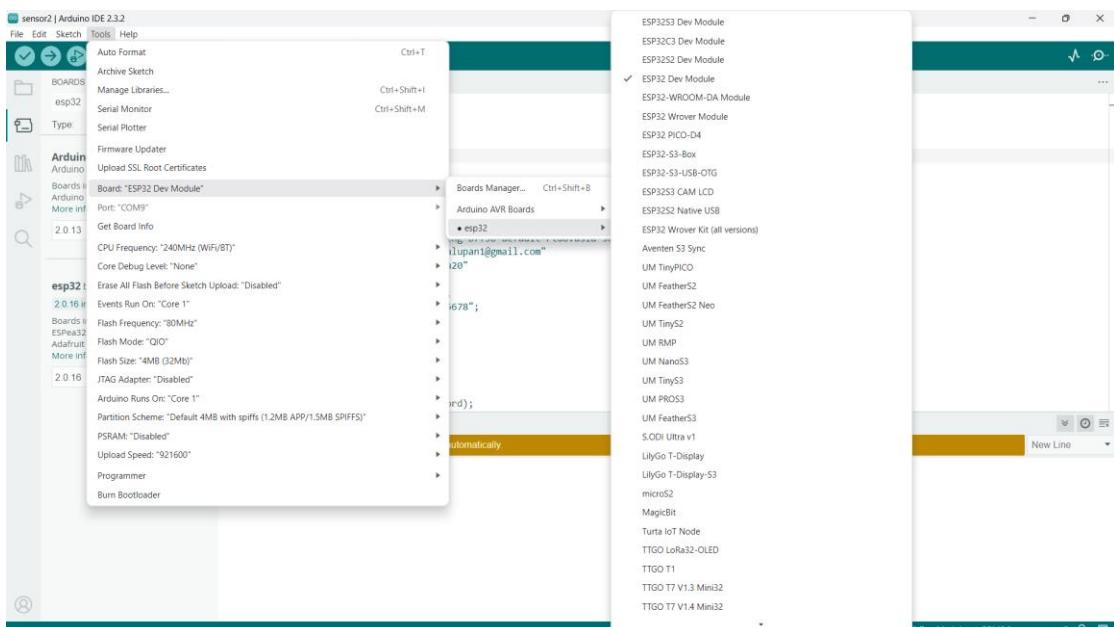
**Gambar 4.23 Preferences Tambah Link**

4. Buka *Boards Manager*, instal paket “esp32 by Espressif Systems”



**Gambar 4.24 Board Manager ESP32**

5. Pilih *ESP32 Dev Module* pada *Tools* → *Board* → *esp32* → *ESP32 Dev Module*.



**Gambar 4.25 Pemilihan Detail ESP32 Dev Module**

6. Instalasi paket untuk papan pengembang *ESP32* Selesai ditandai dengan Gambar 4.26



**Gambar 4.26** Penggunaan Paket *ESP32*

#### 4.3.3. Rangkaian code program 3 unit perangkat keras

Setelah selesai rangkaian penyematan paket papan pengembang *ESP32*, semua fitur yang diperlukan untuk menjalankan fungsi operasional perangkat dijelaskan pada bagian pengkodean berikut.

##### 1. Inisialisasi *Library*

Library yang digunakan sesuai keperluan yaitu *WiFiManager*, *Firebase*, *ESP32*, *TokenHelper* dan *RTDBHelper*.

```
#include <WiFiManager.h>
#include <FirebaseESP32.h>
#include <WiFiUdp.h>
#include <addons/TokenHelper.h>
#include <addons/RTDBHelper.h>
```

**Gambar 4.27** Inisialisasi *Library* *ESP32*

##### 2. Inisialisasi *Pin*

Terdapat pin *trigger* pada sensor ultrasonik yang mana dihubungkan pada pin 25, dan echo sensor ultrasonik dihubungkan pada pin 26. Led berwarna biru yang terdapat pada papan *ESP32* itu sendiri penulis pergunakan untuk memeriksa secara kasat mata bahwa papan terhubung dalam koneksi internet.

```
#define triggerPin 25
#define echoPin 26
#define ledPin 2
```

**Gambar 4.28** Inisialisasi Pin *ESP32*

Fungsi *setupPins* untuk membuat variabel *triggerPin* merupakan *Output*, *echoPin* merupakan *input* dan *ledPin* merupakan *output*.

```
void setupPins() {
    pinMode(triggerPin, OUTPUT);
    pinMode(echoPin, INPUT);
    pinMode(ledPin, OUTPUT);
}
```

**Gambar 4.29** Fungsi *Setup Pins*

##### 3. Kredensial *Firebase Realtime Database*

Keperluan konfigurasi Firebase didapatkan dari laman [firebase.google.com](https://firebase.google.com).

```
#define API_KEY "AIzaSyAcaYr8f5QN-Z_pc1iNsJEjK6e0sFh57gM"
#define DATABASE_URL "smart-parking-b7f3b-default-rtbd.firebaseio.com"
#define USER_EMAIL "albertlukastalupan1@gmail.com"
#define USER_PASSWORD "Konglomuda20"
```

**Gambar 4.30 Kredensial Firebase**

#### 4. Kredensial *Wifi*

Kredensial wifi disimpan pada variabel SSID dan password.

```
const char* SSID = "samsung";
const char* password = "12345678";
```

**Gambar 4.31 Kredensial *Wi-Fi***

#### 5. Inisialisasi fungsi pada *Library Firebase*

Fungsi berikut ini merupakan fungsi yang telah disediakan untuk pengiriman objek ke firebase.

```
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;
```

**Gambar 4.32 Inisialisasi Fungsi *Firebase***

#### 6. Koneksi ke *WiFi*

Fungsi berikut ini merupakan konfigurasi untuk melakukan koneksi ke jaringan internet yang tersedia berdasarkan variabel kredensial.

```
void connectToWiFi() {
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("Connected to WiFi");
}
```

**Gambar 4.33 Konfigurasi Koneksi ke *Wi-Fi***

Apabila sudah terkoneksi maka diberikan penanda lampu *LED* berwarna biru yang berkedip.

```
void wifiLedControl() {
    if (WiFi.status() == WL_CONNECTED) {
        digitalWrite(ledPin, HIGH);
        delay(500);
        digitalWrite(ledPin, LOW);
        delay(500);
    }
}
```

**Gambar 4.34 Konfigurasi *LED Control***

## 7. Konfigurasi *Firebase Realtime Database*

Fungsi Konfigurasi ini untuk dapat mengalokasikan variabel yang telah dideklarasikan untuk dapat terhubung ke *Firebase*.

```
void initializeFirebase() {
    Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);
    config.api_key = API_KEY;
    auth.user.email = USER_EMAIL;
    auth.user.password = USER_PASSWORD;
    config.database_url = DATABASE_URL;
    config.token_status_callback = tokenStatusCallback;
    Firebase.reconnectWiFi(true);
    Firebase.begin(&config, &auth);
}
```

**Gambar 4.35** Konfigurasi *Firebase*

## 8. Perhitungan jarak Sensor Ultrasonik

Fungsi berikut ini untuk dapat menghitung jarak yang terdeteksi oleh sensor.

```
float readDistance() {
    digitalWrite(triggerPin, LOW);
    delayMicroseconds(2);
    digitalWrite(triggerPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(triggerPin, LOW);

    long duration = pulseIn(echoPin, HIGH);
    float distance = duration * 0.034 / 2;
    Serial.print("Distance : ");
    Serial.print(distance);
    Serial.println(" cm");
    delay(100);
    return distance;
}
```

**Gambar 4.36** Konfigurasi Fungsi Pengukuran Sensor

## 9. Pengiriman Data *Realtime Database*

Setelah membaca nilai jarak dari fungsi ukur jarak, diperlukan logika jika jarak dibawah 1m, maka kirim nilai 1 selebihnya 2. Berikut merupakan 3 fungsi yang membedakan pengiriman data setiap mikrokontroler yang ditandai dengan *path*:

1. test/sensor1
2. test/sensor2
3. test/sensor3

Data 1 atau 2 ini berubah secara *realtime*, dikarenakan memang terdapat socket yang dapat *hit* permintaan *HTTP* berulang dan bekerja dengan sendirinya apabila ada perubahan nilai antara mikrokontroler *ESP32* dengan *Firebase*.

```

void updateFirebaseData() {
    static unsigned long lastUpdateMillis = 0;
    const unsigned long updateInterval = 200;

    if (millis() - lastUpdateMillis >= updateInterval) {
        lastUpdateMillis = millis();
        float distance = readDistance();
        int status = (distance <= 100) ? 1 : 2;

        static int lastStatus = -1;
        if (status != lastStatus) {
            lastStatus = status;
            if (Firebase.ready()) {
                if (Firebase.setInt(fbdo, F("/test/sensor1"), status)) {
                    Serial.print("Data sensor 1 sent successfully: ");
                    Serial.println(status);
                } else {
                    Serial.println("Error sending data: " + fbdo.errorReason());
                }
            }
        }
    }
}

```

**Gambar 4.37 Konfigurasi Pengiriman Data Sensor 1**

```

void updateFirebaseData() {
    static unsigned long lastUpdateMillis = 0;
    const unsigned long updateInterval = 200;

    if (millis() - lastUpdateMillis >= updateInterval) {
        lastUpdateMillis = millis();
        float distance = readDistance();
        int status = (distance <= 100) ? 1 : 2;

        static int lastStatus = -1;
        if (status != lastStatus) {
            lastStatus = status;
            if (Firebase.ready()) {
                if (Firebase.setInt(fbdo, F("/test/sensor2"), status)) {
                    Serial.print("Data sensor 2 sent successfully: ");
                    Serial.println(status);
                } else {
                    Serial.println("Error sending data: " + fbdo.errorReason());
                }
            }
        }
    }
}

```

**Gambar 4.38 Konfigurasi Pengiriman Data Sensor 2**

```

void updateFirebaseData() {
    static unsigned long lastUpdateMillis = 0;
    const unsigned long updateInterval = 200;

    if (millis() - lastUpdateMillis >= updateInterval) {
        lastUpdateMillis = millis();
        float distance = readDistance();
        int status = (distance <= 100) ? 1 : 2;

        static int lastStatus = -1;
        if (status != lastStatus) {
            lastStatus = status;
            if (Firebase.ready()) {
                if (Firebase.setInt(fbdo, F("/test/sensor3"), status)) {
                    Serial.print("Data sensor 3 sent successfully: ");
                    Serial.println(status);
                } else {
                    Serial.println("Error sending data: " + fbdo.errorReason());
                }
            }
        }
    }
}

```

**Gambar 4.39 Konfigurasi Pengiriman Data Sensor 3**

## 10. Setup

Fungsi setup merupakan fungsi ini digunakan untuk menginisiasi serial, memanggil fungsi koneksi ke internet, fungsi konfigurasi pin dan fungsi inisialisasi ke *Firebase*.

```
void setup() {
    Serial.begin(115200);
    connectToWiFi();
    setupPins();
    initializeFirebase();
}
```

**Gambar 4.40** Memanggil Semua Fungsi untuk Setup

## 11. Loop

Pada Fungsi *loop* merupakan fungsi yang dilakukan berulang secara terus menerus yaitu kontrol koneksi internet ditandai dengan lampu *LED* biru berkedip dan fungsi utama yaitu pengiriman data keberadaan mobil ke *Firebase Realtime Database*.

```
void loop() {
    wifiLedControl();
    updateFirebaseData();
}
```

**Gambar 4.41** Fungsi Perulangan Kerja untuk *ESP32*

### 4.3.4. Realtime database

Pada *path test* terdapat tiga data sensor yang dapat menerima nilai dari mikrokontroler untuk dapat dikonsumsi oleh *frontend* aplikasi *mobile*. Struktur path dapat dilihat pada Gambar 4.42.



**Gambar 4.42** Data pada Firebase Data Sensor-sensor

### 4.3.5. Pengujian sistem

Setelah seluruh rangkaian implementasi perangkat keras rampung, penulis melakukan pengujian sistem dengan mengukur waktu pengujian yang dicatat pada *serial monitor* dengan data waktu pengukuran yang dilakukan oleh sensor, dan waktu pengiriman nilai 1 atau 2 ke *Firebase Realtime Database*.

Untuk mempermudah dalam pencatatan mengukur waktu. Kode fungsi *readDistance*, digunakan variabel *measureStart* dan diapit dengan *measureEnd* rumus seperti pada Gambar 4.43.

```
unsigned long measureStart = millis();
float distance = readDistance();
unsigned long measureEnd = millis();
```

**Gambar 4.43** Code Pengujian Waktu Sensor Membaca Jarak

Selanjutnya, pengukuran pengiriman perubahan data yang dideteksi oleh perangkat keras dilakukan pada dengan mengapit fungsi pengiriman data firebase dengan *sendStart* dan *sendEnd*.

```
if (Firebase.ready()) {
    unsigned long sendStart = millis();
    if (Firebase.setInt(fbdo, F("/test/sensor3"), status)) {
        unsigned long sendEnd = millis();
```

**Gambar 4.44** Code Pengujian Waktu Pengiriman Data ke Firebase

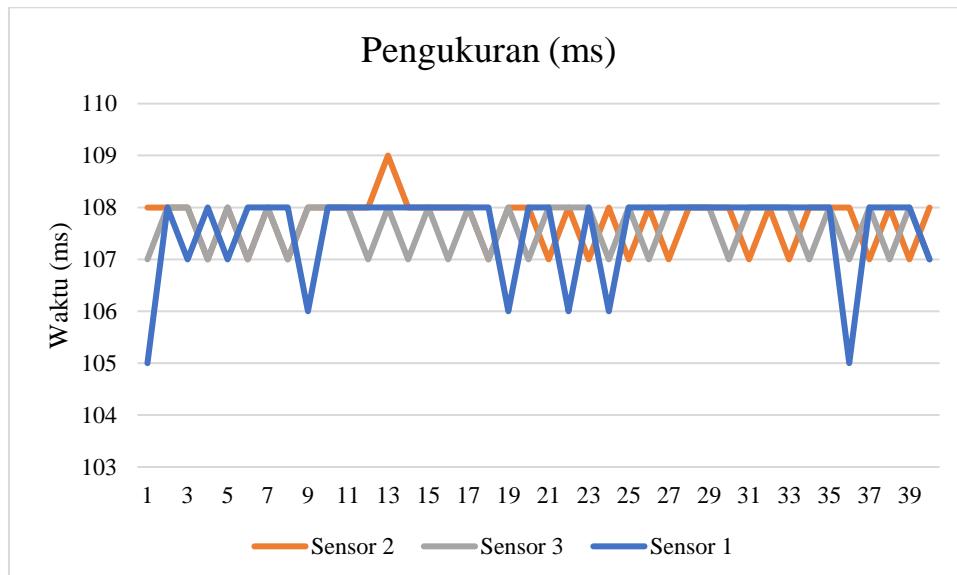
Nilai akan disimpan dalam variabel *sendTime*, *measureTime* dan *totalTime*.

```
unsigned long sendTime = sendEnd - sendStart;
unsigned long measureTime = measureEnd - measureStart;
unsigned long totalTime = sendTime + measureTime;
```

**Gambar 4.45** Rumus Perhitungan Waktu Pengujian

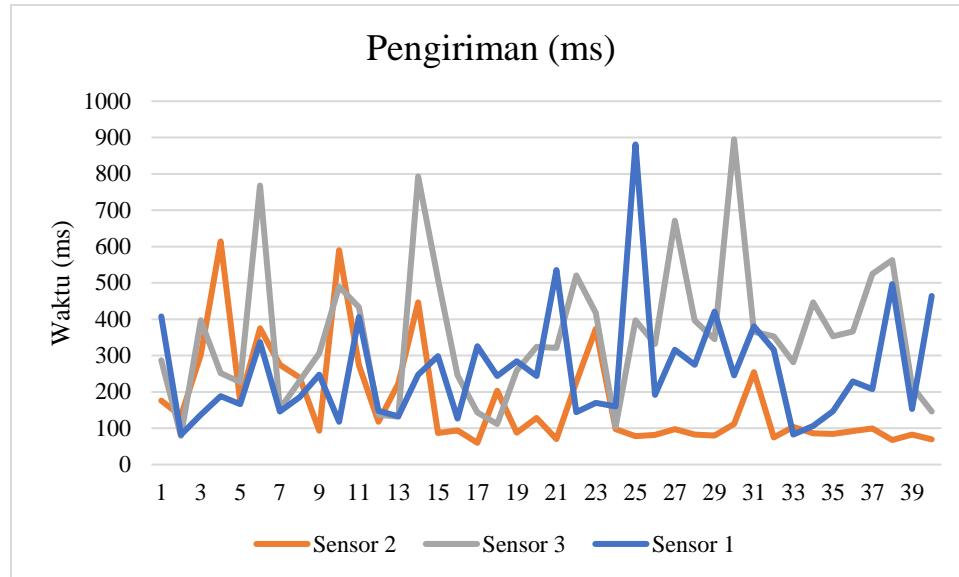
1. Nilai pengujian waktu terhadap pengukuran dan pengiriman sensor

Gambar 4.46, merupakan grafik pengukuran waktu sensor dalam mendeteksi objek di depannya.



**Gambar 4.46** Grafik Waktu Pengukuran Sensor-Sensor

Gambar 4.47, merupakan grafik pengukuran waktu sensor dalam mengirim setiap perubahan nilai 1 atau 2 ke Firebase menggunakan koneksi internet.



**Gambar 4.47** Grafik Waktu Pengiriman Data ke *Firebase*

Tabel 4.1 merupakan data konkrit dari data yang disajikan pada kedua grafik.

1. Inisial M pada tabel adalah *measure* yang merupakan pengukuran dalam milisekon.
2. Inisial S pada tabel adalah *send* yang artinya nilai waktu pengiriman ke Firebase dalam milisekon
3. Inisial T pada tabel adalah total waktu pengukuran dan pengiriman.

**Tabel 4.1** Data Pengujian Waktu Pengukuran dan Pengiriman Data

No	Status	Waktu (ms)								
		Sensor 1			Sensor 2			Sensor 3		
		M	S	T	M	S	T	M	S	T
1	1	105	408	513	108	176	284	107	287	394
2	2	108	82	190	108	134	242	108	79	187
3	1	107	137	244	108	303	411	108	397	505
4	2	108	188	296	107	614	721	107	252	359
5	1	107	166	273	108	174	282	108	227	335
6	2	108	337	445	107	375	482	107	768	875
7	1	108	146	254	108	275	383	108	153	261
8	2	108	186	294	107	239	346	107	231	338
9	1	106	247	353	108	93	201	108	306	414
10	2	108	118	226	108	590	698	108	490	598
11	1	108	406	514	108	274	382	108	433	541
12	2	108	148	256	108	118	226	107	137	244
13	1	108	133	241	109	223	332	108	132	240
14	2	108	247	355	108	446	554	107	793	900
15	1	108	298	406	108	87	195	108	510	618

**Tabel 4.1** Data Pengujian Waktu Pengukuran dan Pengiriman Data (Lanjutan)

No	Status	Waktu (ms)								
		Sensor 1			Sensor 2			Sensor 3		
		M	S	T	M	S	T	M	S	T
16	2	108	127	235	108	94	202	107	246	353
17	1	108	326	434	108	60	168	108	143	251
18	2	108	244	352	107	203	310	107	112	219
19	1	106	284	390	108	88	196	108	259	367
20	2	108	244	352	108	128	236	107	324	431
21	1	108	535	643	107	70	177	108	321	429
22	2	106	144	250	108	227	335	108	520	628
23	1	108	170	278	107	373	480	108	417	525
24	2	106	160	266	108	98	206	107	105	212
25	1	108	881	989	107	78	185	108	397	505
26	2	108	192	300	108	82	190	107	332	439
27	1	108	316	424	107	98	205	108	671	779
28	2	108	275	383	108	83	191	108	396	504
29	1	108	421	529	108	80	188	108	345	453
30	2	108	246	354	108	112	220	107	895	1002
31	1	108	380	488	107	254	361	108	365	473
32	2	108	316	424	108	75	183	108	353	461
33	1	108	83	191	107	104	211	108	282	390
34	2	108	106	214	108	86	194	107	446	553
35	1	108	147	255	108	84	192	108	353	461
36	2	105	229	334	108	92	200	107	366	473
37	1	108	208	316	107	99	206	108	525	633
38	2	108	497	605	108	68	176	107	563	670
39	1	108	153	261	107	83	190	108	216	324
40	2	107	464	571	108	69	177	107	146	253
<b>Rata-rata</b>		<b>107.5</b>	<b>259.8</b>	<b>367.4</b>	<b>107.7</b>	<b>175.2</b>	<b>282.9</b>	<b>107.6</b>	<b>357.3</b>	<b>714.9</b>
<b>Tercepat</b>		<b>105</b>	<b>82</b>	<b>190</b>	<b>107</b>	<b>60</b>	<b>168</b>	<b>107</b>	<b>79</b>	<b>187</b>
<b>Terlama</b>		<b>108</b>	<b>881</b>	<b>989</b>	<b>109</b>	<b>614</b>	<b>721</b>	<b>108</b>	<b>895</b>	<b>1002</b>

Dari kumpulan data pada Tabel 4.1, didapati kesimpulan pada Tabel 4.2.

**Tabel 4.2** Data Ringkasan Pengukuran dan Pengiriman Data

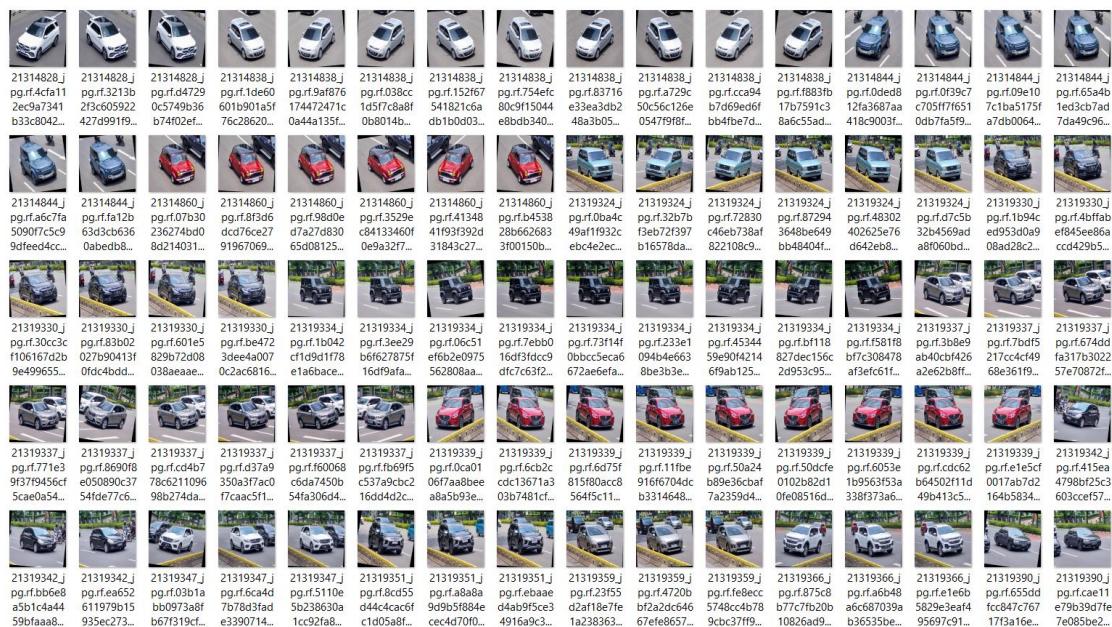
	Pengukuran (ms)		Pengiriman (ms)	
	Tercepat	Terlama	Tercepat	Terlama
<b>Sensor 1</b>	105	108	82	881
<b>Sensor 2</b>	107	109	60	614
<b>Sensor 3</b>	107	108	79	895
<b>Rata-rata</b>	<b>106.3</b>	<b>108.3</b>	<b>73.6</b>	<b>796.6</b>

Berdasarkan Tabel 4.2, didapatkan hasil rata-rata dari ketiga perangkat keras dalam mengukur dan mengirimkan data ke Firebase. Dapat diamati bahwa rata-rata tercepat pengukuran yaitu 106.3 ms dan tercepat untuk pengiriman sebesar 73.6 ms. Namun disamping itu terdapat kekurangan yaitu waktu terlama dalam pengukuran sebesar 108.3 ms dan terlama dalam pengiriman 796.6 ms. Berdasarkan waktu terlama pada saat pengiriman terlama dari *ESP32* ke *Firebase* dikarenakan jaringan yang kurang stabil.

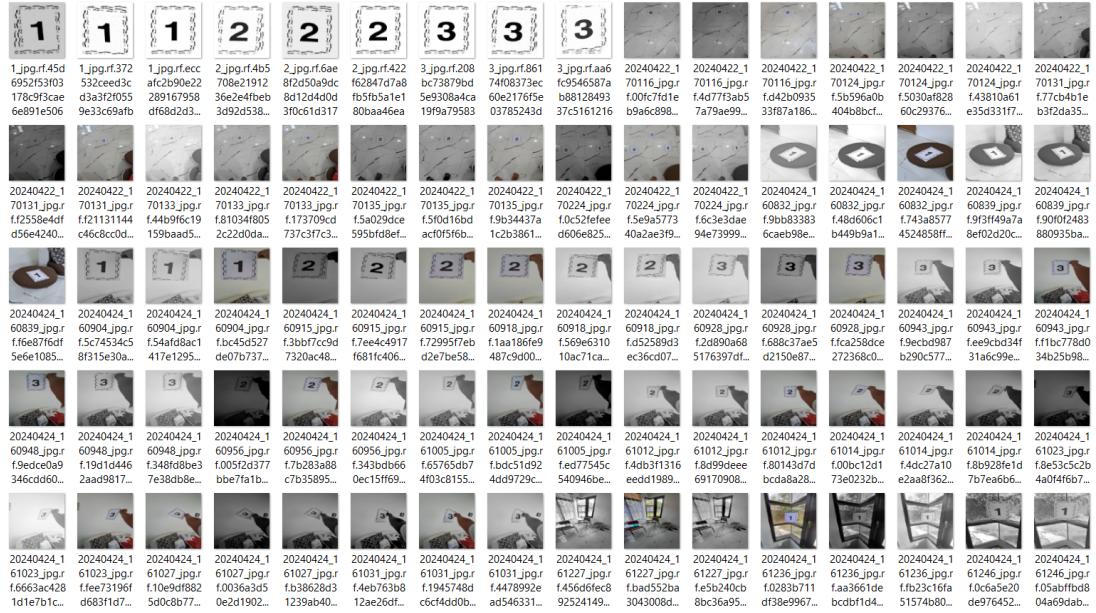
#### 4.4. Implementasi Pemrosesan Citra Digital

##### 4.4.1. Implementasi data pada pemrosesan citra

Pada implementasi data citra ini, digunakan dataset yang telah dikumpulkan sebelumnya. Dataset ini meliputi sumber dari platform roboflow sebagai sumber utama citra plat nomor polisi dan kertas penanda parkir. Implementasi data ini, telah melalui proses penambahan data augmentasi. Sampel dapat dilihat pada Gambar 4.48 dan Gambar 4.49.



Gambar 4.48 Data Plat Nomor Polisi



**Gambar 4.49 Data Kertas Penanda Slot Parkir**

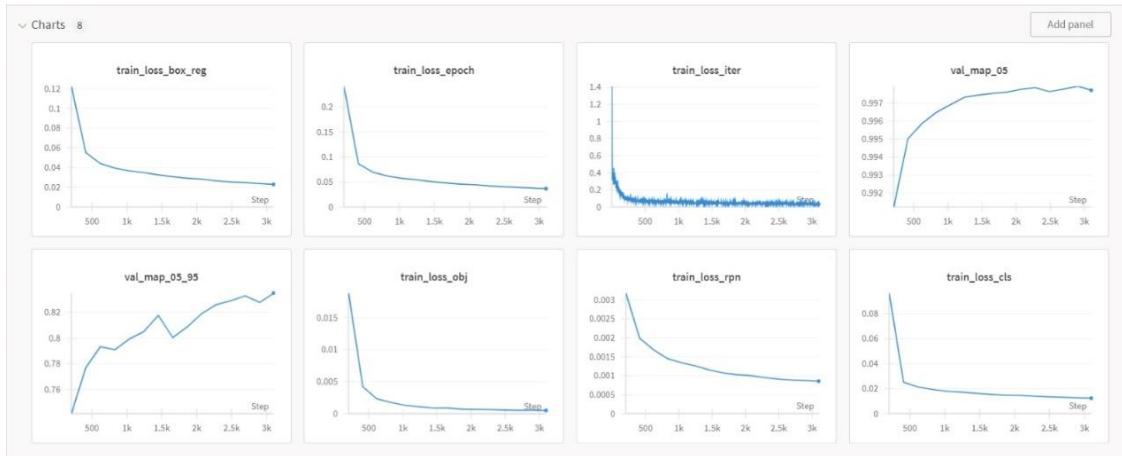
#### 4.4.2. Pelatihan sistem

Pelatihan sistem ini merupakan langkah pelatihan model untuk mengenali objek. Semua data yang digunakan dalam pelatihan model telah melalui tahap pra-pemrosesan. Proses pelatihan sistem dilakukan menggunakan *Google Colab* dan/atau *Kaggle Jupyter Notebook* sebagai alat bantu yang populer, serta *model pre-trained fasterrcnn\_resnet50\_fpn\_v2\_coco-dd69338a.pth* digunakan sebagai dasar untuk proses pelatihan. *Model* pada proses pelatihan dilatih sebanyak 4 kali dengan beberapa konfigurasi berbeda pada nilai *epoch* dan *batch*, dapat dirincikan pada Tabel 4.3.

**Tabel 4.3** Daftar Hasil Riwayat Pelatihan Model

<i>Epoch</i>	<i>Batch</i>	<i>mAP</i>	<i>mAP 0.5</i>	<i>mAP 0.5:0.95</i>	<i>Precision</i>	<i>Recall</i>	<i>Status</i>
<b>5</b>	11	0.808	0.996	0.808	0.805	0.85	-
<b>12</b>	12	0.817	0.997	0.817	0.817	0.644	-
<b>15</b>	13	0.835	0.997	0.835	0.998	0.878	Digunakan
<b>20</b>	16	0.832	0.998	0.832	0.998	0.876	-

Setelah melalui seluruh proses pelatihan seperti pada Tabel 4.3. maka didapatkan model terbaik dengan nilai *MAP* 0.835, *MAP 0.5* sebesar 0.997, precision sebesar 0.998 dan recall 0.878. Model dengan hasil terbaik pada *Faster R-CNN* disimpan dengan nama *best\_model.pt*. grafik dari hasil proses pelatihan model terbaik dapat dilihat pada Gambar 4.50.

**Gambar 4.50** Grafik Pelatihan

#### 4.4.3. Pengujian sistem

##### 1. Evaluasi Kemampuan Model

Pengujian sistem merupakan proses evaluasi kemampuan sistem yang telah didapat sebelumnya dari proses pelatihan.

**Tabel 4.4** Data Hasil Uji Model

No.	<i>Actual class</i>	<i>Predicted class</i>
1	Slot_Sign	Slot_Sign
2	Slot_Sign	Slot_Sign
3	Slot_Sign	Slot_Sign
4	Slot_Sign	Slot_Sign
5	Slot_Sign	Slot_Sign
6	Slot_Sign	Slot_Sign
7	Slot_Sign	Slot_Sign
8	Slot_Sign	Slot_Sign
9	Slot_Sign	Slot_Sign
10	Slot_Sign	Slot_Sign
11	Slot_Sign	Slot_Sign
12	Slot_Sign	Slot_Sign
13	Slot_Sign	Slot_Sign
14	Slot_Sign	Slot_Sign
15	Slot_Sign	Slot_Sign
16	Slot_Sign	Slot_Sign
17	Slot_Sign	Slot_Sign

No.	<i>Actual class</i>	<i>Predicted class</i>
1	Slot_Sign	Slot_Sign
2	Slot_Sign	Slot_Sign
3	Slot_Sign	Slot_Sign
4	Slot_Sign	Slot_Sign
5	Slot_Sign	Slot_Sign
6	Slot_Sign	Slot_Sign
7	Slot_Sign	Slot_Sign
8	Slot_Sign	Slot_Sign
9	Slot_Sign	Slot_Sign
10	Slot_Sign	Slot_Sign
11	Slot_Sign	Slot_Sign
12	Slot_Sign	Slot_Sign
13	Slot_Sign	Slot_Sign
14	Slot_Sign	Slot_Sign
15	Slot_Sign	Slot_Sign
16	Slot_Sign	Slot_Sign
17	Slot_Sign	Slot_Sign

**Tabel 4.4** Data Hasil Uji Model (Lanjutan)

No.	Actual Class	Predicted Class
18	Slot_Sign	Slot_Sign
19	Slot_Sign	Slot_Sign
20	Slot_Sign	Slot_Sign
21	Slot_Sign	Slot_Sign
22	Slot_Sign	Slot_Sign
23	Slot_Sign	Slot_Sign
24	Slot_Sign	Slot_Sign
25	Slot_Sign	Slot_Sign
26	Slot_Sign	Slot_Sign
27	Slot_Sign	Slot_Sign
28	Slot_Sign	Slot_Sign
29	Slot_Sign	Slot_Sign
30	Slot_Sign	Slot_Sign
31	Slot_Sign	Slot_Sign
32	Slot_Sign	Slot_Sign
33	Slot_Sign	Slot_Sign
34	Slot_Sign	Slot_Sign
35	Slot_Sign	Slot_Sign
36	Slot_Sign	Slot_Sign
37	Slot_Sign	Slot_Sign
38	Slot_Sign	Slot_Sign
39	Slot_Sign	Slot_Sign
40	Slot_Sign	License_Plate
41	License_Plate	License_Plate
42	License_Plate	Slot_Sign
43	Slot_Sign	Slot_Sign
44	Slot_Sign	License_Plate
45	Slot_Sign	License_Plate
46	Slot_Sign	Slot_Sign
47	Slot_Sign	Slot_Sign
48	Slot_Sign	License_Plate

**Tabel 4.4** Data Hasil Uji Model (Lanjutan)

No.	Actual Class	Predicted Class
49	Slot_Sign	License_Plate
50	Slot_Sign	Slot_Sign
51	License_Plate	License_Plate
52	License_Plate	Slot_Sign
53	Slot_Sign	License_Plate
54	Slot_Sign	Slot_Sign
55	License_Plate	License_Plate
56	License_Plate	Slot_Sign
57	Slot_Sign	License_Plate
58	Slot_Sign	Slot_Sign
59	Slot_Sign	Slot_Sign
60	Slot_Sign	License_Plate
61	Slot_Sign	License_Plate
62	Slot_Sign	Slot_Sign
63	Slot_Sign	Slot_Sign
64	Slot_Sign	License_Plate
65	License_Plate	License_Plate
66	License_Plate	License_Plate
67	License_Plate	License_Plate
68	License_Plate	License_Plate
69	License_Plate	Slot_Sign
70	License_Plate	License_Plate
71	License_Plate	License_Plate
72	License_Plate	License_Plate
73	License_Plate	License_Plate
74	License_Plate	License_Plate
75	License_Plate	License_Plate
76	License_Plate	License_Plate
77	License_Plate	License_Plate
78	License_Plate	License_Plate
79	License_Plate	License_Plate

**Tabel 4.4** Data Hasil Uji Model (Lanjutan)

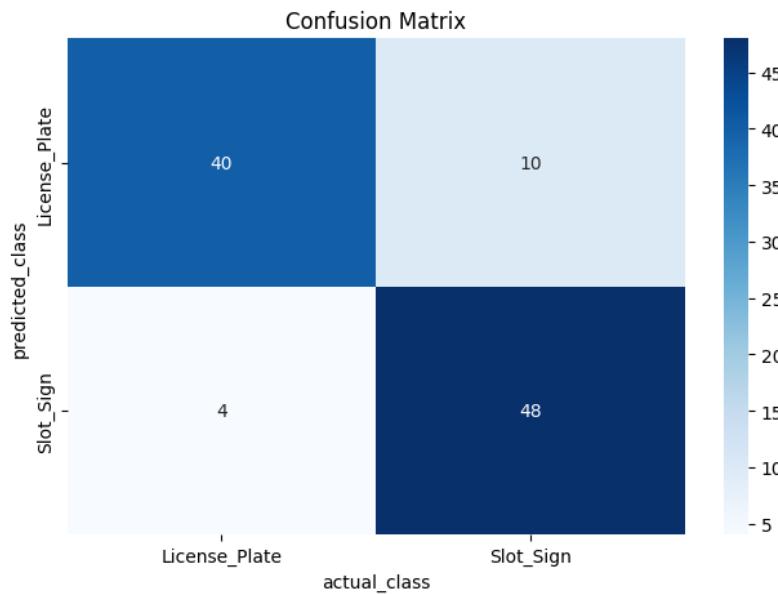
No.	Actual Class	Predicted Class
80	License_Plate	License_Plate
81	License_Plate	License_Plate
82	License_Plate	License_Plate
83	License_Plate	License_Plate
84	License_Plate	License_Plate
85	License_Plate	License_Plate
86	License_Plate	License_Plate
87	License_Plate	License_Plate
88	License_Plate	License_Plate
89	License_Plate	License_Plate
90	License_Plate	License_Plate
91	License_Plate	License_Plate
92	License_Plate	License_Plate
93	License_Plate	License_Plate
94	License_Plate	License_Plate
95	License_Plate	License_Plate
96	License_Plate	License_Plate
97	License_Plate	License_Plate
98	License_Plate	License_Plate
99	License_Plate	License_Plate
100	License_Plate	License_Plate
101	License_Plate	License_Plate
102	License_Plate	License_Plate

Berdasarkan data pada Tabel 4.4, disajikan pada Tabel 4.5 nilai *TP*, *TN*, *FP*, dan *FN*

**Tabel 4.5** Nilai *TP*, *TN*, *FP*, dan *FN*

Class	TP	TN	FP	FN
<i>Slot Sign</i>	48	0	10	4
<i>License Plate</i>	40	0	4	10

Tabel 4.5 merupakan pengujian yang merupakan representasi keseluruhan data yang digunakan pada proses uji sistem. Selanjutnya dilakukan proses evaluasi dari hasil pengujian sistem menggunakan *confusion matrix*. Hasil *confusion matrix* dapat dilihat pada Gambar 4.51.



**Gambar 4.51 Confusion Matrix**

Berdasarkan Gambar 4.51, terdapat beberapa kesalahan dalam proses deteksi objek pada setiap kelas. Pada proses pendekripsi *License\_plate* terdapat kesalahan dimana sistem 10 kali mendekripsi sebagai *Slot\_Sign*. Pada pendekripsi yang seharusnya *Slot\_Sign* terdapat 4 kali mendekripsi sebagai *License\_Plate*. Hal ini dikarenakan kemiripan objek hitam diatas putih pada *License\_Plate* dan *Slot\_Sign*.

### 1. Persamaan *precision*

$$Precision = \frac{TP}{TP + FP}$$

$$Precision_{Slot\_Sign} = \frac{48}{48 + 10}$$

$$Precision_{Slot\_Sign} = 0.82$$

$$Precision_{License\_Plate} = \frac{40}{40 + 4}$$

$$Precision_{License\_Plate} = 0.90$$

### 2. Persamaan *recall*

$$Recall = \frac{TP}{TP + FN}$$

$$Recall\_Slot\_Sign = \frac{48}{48 + 4}$$

$$Recall\_Slot\_Sign = 0.92$$

$$Recall\_License\_Plate = \frac{40}{40 + 10}$$

$$Recall\_License\_Plate = 0.8$$

### 3. Persamaan *F1 Score*

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

$$F1\_Slot\_Sign = 2 \times \frac{0.82 \times 0.92}{0.82 + 0.92}$$

$$F1\_Slot\_Sign = 0.86$$

$$F1\_License\_Plate = 2 \times \frac{0.9 \times 0.8}{0.9 + 0.8}$$

$$F1\_License\_Plate = 0.84$$

Pada Tabel 4.6, ditampilkan ringkasan nilai berdasarkan persamaan *precision*, *recall* dan *F1 Score*.

**Tabel 4.6** Nilai *Precision*, *Recall* dan *F1 Score*

	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>
<b>Slot Sign</b>	82%	92%	86%
<b>License Plate</b>	90%	80%	84%

Berdasarkan seluruh uji sistem, didapati nilai akurasi sebagai berikut:

$$Accuracy = \frac{TP + TN}{Total\ Data}$$

$$Accuracy = \frac{88}{102}$$

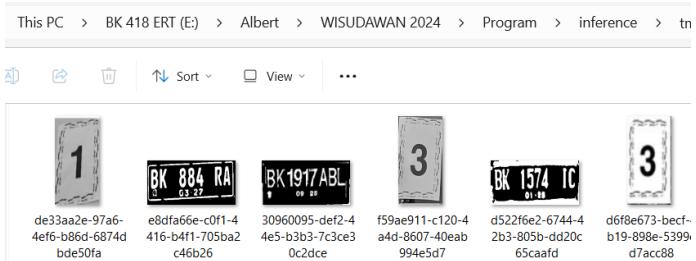
$$Accuracy = 0.86$$

Peneliti mendapatkan hasil pelatihan yang baik dengan nilai *precision* dan *recall* masing-masing pada kertas penanda parkir sebesar 82% dan 92%, dan *precision* dan *recall* pada plat nomor polisi sebesar 90% dan 80%. Penulis menguji mendapat kan nilai *F1 Score* sebesar 86% dan 84% pada masing masing tugas klasifikasi. Dengan nilai akurasi 86%, dapat diambil kesimpulan bahwa model yang telah dilatih bekerja dengan baik dalam melakukan inferensi namun belum cukup sempurna.

## 4.5. Implementasi EasyOCR

### 4.5.1. Implementasi data

Objek hasil inferensi dari Model *Faster R-CNN*, kemudian digunakan untuk dapat ekstraksi karakter dan disimpan pada folder “/tmp”. Berikut merupakan sampel hasil dari *preprocessing* data objek yang terdeteksi.



**Gambar 4.52** Data Preprocess EasyOCR

### 4.5.2. Pengujian sistem

#### 1. Plat Nomor Polisi

Berikut merupakan hasil pengujian ekstrak karakter yang dilakukan oleh *EasyOCR* pada objek hasil inferensi, plat nomor polisi.

**Tabel 4.7** Hasil Uji EasyOCR Ekstrak Karakter Plat Nomor Polisi

No	Sebelum	Sesudah	Plat	Hasil	Persentase
1			BK1917ABL	BK1917ABL	100%
2			BK884RA	BK884RA	100%
3			BK1787AV	BK1787AV	100%
4			BK8541VV	BK8541VV	100%

Berdasarkan Tabel 4.7, didapati hasil kumulatif 100% pada 4 objek yang mewakili jenis plat yang ada di Indonesia. Seperti pada Tabel 4.7 pada nomor satu, jenis plat variasi. Pada nomor dua dan tiga, plat putih diatas hitam. Dan yang terakhir plat jenis baru yang ada di Indonesia, plat hitam diatas putih.

## 2. Kertas Penanda Slot Parkir

Pada Tabel 4.8, disajikan hasil *encode* karakter pada teks penanda slot parkir yang dikerjakan oleh teknologi *EasyOCR* pada hasil deteksi, kertas penanda parkir.

**Tabel 4.8** Hasil Uji *EasyOCR* Ekstrak Karakter Kertas Penanda Slot Parkir

No	Sebelum	Sesudah	Slot	Hasil	Persentase
1			1	1	100%
2			2	2	100%
3			3	3	100%

Evaluasi pengujian pada kertas penanda parkir juga memiliki nilai akhir yang baik sebesar 100% dalam mengenkode karakter.

## 3. Evaluasi Waktu Inferensi dan Ekstrak Karakter

Pada Tabel 4.9, disajikan informasi detil mengenai waktu inferensi dan *EasyOCR* bekerja.

**Tabel 4.9** Hasil Pengujian Waktu Inferensi Gambar dan Ekstrak Karakter

<b>Percobaan</b>	<b>Inferensi (s)</b>	<b>EasyOCR</b>		<b>Total (s)</b>
		<b>Plat (s)</b>	<b>Slot (s)</b>	
<b>1</b>	1.84	0.65	0.23	2.72
<b>2</b>	1.81	0.65	0.22	2.68
<b>3</b>	2.35	0.87	0.28	3.5
<b>4</b>	1.78	0.59	0.21	2.58
<b>5</b>	1.81	0.64	0.22	2.67
<b>6</b>	1.85	0.59	0.21	2.65
<b>7</b>	1.85	0.6	0.2	2.65
<b>8</b>	2.02	0.66	0.2	2.88
<b>9</b>	1.72	0.64	0.25	2.61

**Tabel 4.9** Hasil Pengujian Waktu Inferensi Gambar dan Ekstrak Karakter  
(Lanjutan)

<b>Percobaan</b>	<b>Inferensi (s)</b>	<b>EasyOCR</b>		<b>Total (s)</b>
		<b>Plat (s)</b>	<b>Slot (s)</b>	
<b>10</b>	1.89	0.67	0.2	2.76
<b>Rata-Rata</b>	1.892	0.656	0.222	2.77
<b>Tercepat</b>	1.72	0.59	0.2	2.58
<b>Terlama</b>	<b>2.35</b>	<b>0.87</b>	<b>0.28</b>	<b>3.5</b>

Berdasarkan Tabel 4.9, didapatkan hasil inferensi tercepat sebesar 1.72, ekstrak karakter pada plat 0.59 detik serta ekstrak karakter pada kertas sebesar 0.28 detik dan ekstraksi dengan rata-rata tercepat sebesar 2.58 detik. angka ini cukup baik untuk nilai akurasi yang tinggi.

#### 4.6. Prosedur Operasional

##### 4.6.1. Persiapan perangkat keras dan lokasi

Pemasangan perangkat keras yang sudah dapat bekerja mengirimkan data diletakkan pada lokasi uji program pada tempat parkir halaman. Sekaligus menempelkan kertas penanda parkir pada tiang yang sejajar dengan sensor.



**Gambar 4.53 Persiapan Operasional**

#### 4.6.2. Persiapan server

Untuk dapat melakukan simulasi dan setiap fungsi yang telah *di-coding* berfungsi, setiap *server* harus dalam kondisi hidup. Terdapat 3 server yang telah dipersiapkan yaitu *server flask*, *server Frontend* dan *server Backend*.

##### 1. Menyalakan *server flask*

Fungsi utama dalam *server flask* pada Gambar 4.54, berguna untuk dapat menginferensi gambar yang dikirim oleh pengguna. Setelah program menginferensi, program mengekstraksi karakter yang berjalan untuk memberikan hasil. *Server flask* memiliki *HTTP Post* untuk mengirimkan data dalam bentuk *JSON* dengan format pada Gambar 4.55, dan contoh hasil data dicetak pada *console* di Gambar 4.56.

```
E:\Albert\WISUDAWAN 2024\Program\inference>python main.py
 * Serving Flask app 'main'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 762-862-163
```

**Gambar 4.54 Server Flask**

```
result_dict = {
    "License_Plate": {
        "result": karakter_license_Plate,
        "score": str(scores[pred_classes.index('License_Plate')]),
        "path": foto_License_Plate
    },
    "slot-sign": {
        "result": karakter_slot_sign,
        "score": str(scores[pred_classes.index('slot-sign')]),
        "path": foto_slot_sign
    }
}
```

**Gambar 4.55 Format Pengiriman Data *Server Flask JSON***

```
{'License_Plate': {'result': 'BK884RA', 'score': '0.9954776', 'path': 'tmp/e8dfaf66e-c0f1-4416-b4f1-705ba2c46b26.jpg'}, 'slot-sign': {'result': '1', 'score': '0.9908375', 'path': 'tmp/de33aa2e-97a6-4fe6-b86d-6874dbde50fa.jpg'}}
```

#### Gambar 4.56 Contoh Hasil JSON

## 2. Menyalakan *server* Antarmuka

*Server Frontend* berguna agar antarmuka aplikasi dapat digunakan pada *android* atau *iOS*, namun dalam pengujian ini penulis menggunakan *android*.

A screenshot of a terminal window. At the top, the path 'E:\Albert\WISUDAWAN 2024\Program\Smart Parking\fe-smart-parking>npm start' is shown. Below it, two commands are listed: '> smart-parking@0.0.1 start' and '> react-native start'. The main content of the window is a large, blue, pixelated version of the Mario Goomba enemy from Super Mario Bros. Below the logo, the text 'Welcome to Metro v0.76.8' is displayed, followed by 'Fast - Scalable - Integrated'. At the bottom, a series of command keys are listed: 'r - reload the app', 'd - open developer menu', 'i - run on iOS', and 'a - run on Android'.

**Gambar 4.57** Server *Frontend*

### 3. Menyalakan *server backend*

Server backend ini mayoritas berfungsi untuk mengolah data yang diterima atau dapat disebut sebagai perantara serta menyimpan, mengambil, menghapus dan memperbarui data pada *Firebase*.

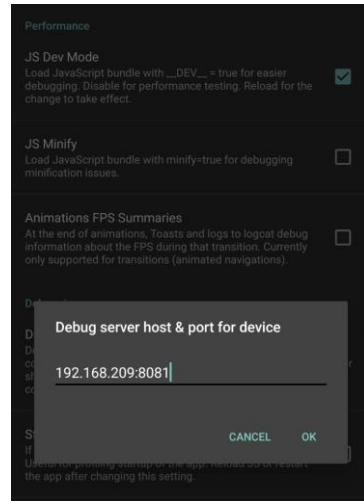
```
E:\Albert\WISUDAWAN 2024\Program\Smart Parking\be-smart-parking>npm start  
> be-smart-parking@1.0.0 start  
> node index.js  
  
[01:48:51.000] INFO: Server is running at http://localhost:8000
```

**Gambar 4.58** Server Backend

Percobaan dilakukan pada gawai *Android* Samsung dan *server* lokal pada Laptop, keduanya harus tergabung dalam *WiFi* yang sama agar memiliki *IP Address* yang sama. Sintaks yang digunakan yaitu “`http://IPAddress:PortBackend`”. Pada Gambar 4.58 merupakan konfigurasi pada instansi axios pada *frontend*, sementara pada Gambar 4.59 merupakan konfigurasi pada android “`http://IPAddress:PortFrontend`”, untuk dapat terhubung dan bertukar data.

```
export const API_URI = 'http://192.168.2.209:8000'
```

**Gambar 4.59** Konfigurasi *IP Address* dan *Port Backend* pada *Frontend*



**Gambar 4.60** Konfigurasi *IP Address* dan *Port* pada *Android*

#### 4.6.3. Alur aplikasi: mobil tanpa aplikasi

Mobil yang terdeteksi oleh sensor namun tidak memesan, maka ditandai dengan slot “Unavailable”. Disajikan pada Gambar 4.61.



**Gambar 4.61** Mobil Tanpa Aplikasi *Unavailable*

#### 4.6.4. Alur aplikasi: mobil dengan aplikasi

Pengguna yang memakai aplikasi dapat memesan lokasi slot parkir pada aplikasi “*Smart Parking*”.

##### 1. Pesan Slot

Seperti pada Gambar 4.62, terdapat slot 1 dan slot 2 kosong yang artinya slot tersebut dapat dipesan.



**Gambar 4.62** Pesan Slot Untuk Parkir

## 2. Input Plat Nomor Mobil

Dengan mengisi formulir plat kendaraan seperti pada Gambar 4.63, pengguna sudah dapat memesan slot parkir yang dapat untuk dipakai.

The screenshot shows a mobile application interface titled 'Unggah nomor plat' (Upload license plate number). It features a text input field with placeholder text 'Masukkan plat nomor kendaraan anda tanpa menggunakan spasi untuk melanjutkan proses parkir'. Below the input field is a 'Plat kendaraan' label and a text input field containing the plate number 'BK1345AD'. A large blue 'Submit' button is positioned below the input fields. At the bottom of the screen, a virtual keyboard is visible.

**Gambar 4.63** Pengguna Mengunggah Identitas Plat Mobil

### 3. Unggah Foto

Setelah pengguna berhasil mendapatkan slot parkir yang sudah dipesan, pengguna dibutuhkan untuk mengupload gambar dengan *tap* tombol *banner*, seperti pada Gambar 4.64. Yang hanya memuat satu plat nomor polisi pada mobil dan yang kedua kertas penanda parkir. Setelah itu klik tombol *Submit*.



**Gambar 4.64** Banner untuk Upload

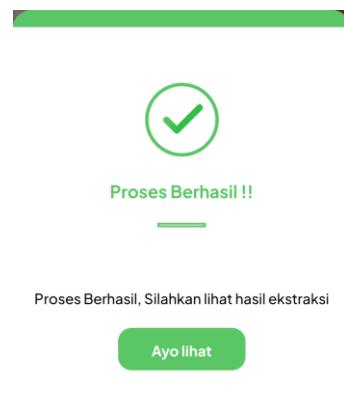


**Gambar 4.65** Pengguna Mengunggah Foto Bukti Parkir

### 4. Validasi

Sistem pada *main.py* akan melaksanakan deteksi kedua objek, kertas penanda slot parkir dan plat nomor polisi pada mobil. File *main.py* juga digunakan sekaligus untuk mengekstrak karakter pada hasil inferensi citra yang sudah diolah melalui proses

*preprocessing*. Jika benar maka akan muncul pesan *pop-up* seperti pada Gambar 4.66. dan diarahkan ke halaman Hasil Ekstraksi, pada halaman itu terdapat Gambar plat nomor polisi mobil dan kertas penanda slot parkir yang disajikan pada Gambar 4.67.

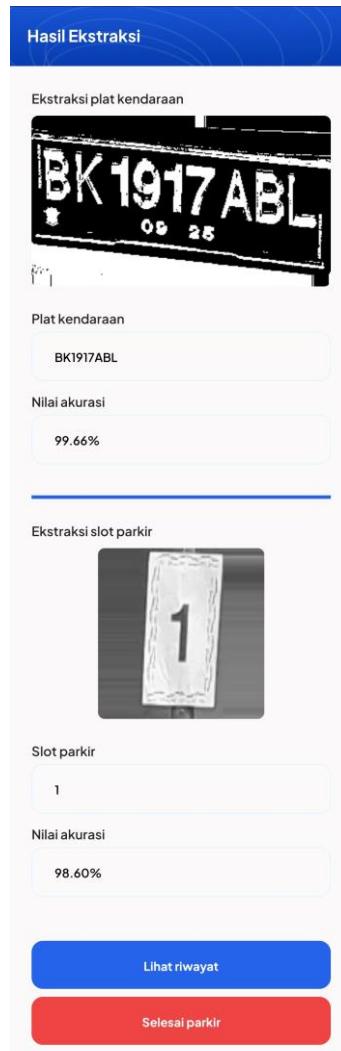


**Gambar 4.66** Terkonfirmasi Kecocokan Data

Jika salah satu tidak terdeteksi atau tidak sesuai maka akan muncul pesan error.

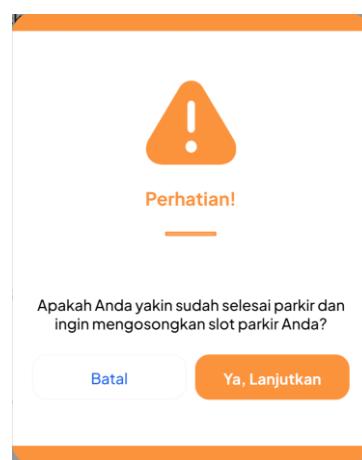
##### 5. Selesai Parkir

Pengguna klik tombol Selesai Parkir untuk dapat lanjut meninggalkan lokasi parkir



**Gambar 4.67** Halaman Hasil Inferensi dan Ekstraksi

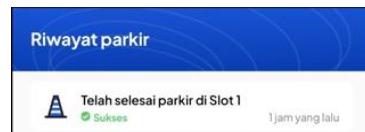
Setelah *tap* tombol selesai parkir, maka akan muncul pesan konfirmasi *pop-up* dan klik “Ya, Lanjutkan”.



**Gambar 4.68** Pesan Konfirmasi Selesai Parkir

Dengan *tap* selesai, akan diberikan pesan konfirmasi terlebih dahulu. Setelah selesai maka sensor, akan bekerja normal kembali karena sebelumnya karena pada fungsi *backend*, jika pengguna telah memesan maka dari aplikasi dapat menampilkan situasi.

## 6. Riwayat



**Gambar 4.69** Riwayat Pemarkiran

Pengguna dapat memeriksa status pemesanan pada riwayat transaksi pada Gambar 4.69.

## **BAB 5**

### **KESIMPULAN DAN SARAN**

#### **5.1. Kesimpulan**

Pada akhir penelitian, penulis menyimpulkan beberapa hal terkait integrasi aplikasi berbasis mobile yang dirancang menggunakan sensor Ultrasonik yang terhubung dengan mikrokontroler *ESP32*. Aplikasi ini bertujuan untuk mendeteksi keberadaan mobil pada setiap slot parkir dan mengirimkan data secara real-time ke *Firebase Realtime Database*. Selain itu, penelitian ini juga mencakup perancangan deep learning menggunakan metode *Faster Region-Based Convolutional Neural Network (Faster R-CNN)*. Tujuan inferensi model ini adalah untuk mendeteksi nomor polisi dan kertas penanda slot parkir sehingga karakter pada masing-masing objek yang terdeteksi dapat diekstraksi. Hal ini berfungsi sebagai bukti bahwa pengguna yang telah memesan slot parkir benar-benar memarkirkan mobil pada slot yang sesuai. Berdasarkan hasil evaluasi dapat disimpulkan bahwa:

1. Kinerja perangkat deteksi keberadaan mobil pada ke 3 nya sudah didapatkan hasil yang sangat baik, dengan rata-rata pengukuran sensor tercepat 106.3 milisekon dan pengiriman nilai perubahan ke *Firebase* sebesar 73.6 milisekon.
2. Kinerja pemrosesan citra menggunakan metode *Faster Region Based Convolutional Neural Network* baik dalam melakukan tugas klasifikasi plat nomor polisi dan kertas penanda parkir ditandai dengan *F1 Score* sebesar 86%, dengan nilai rata-rata *precision* 86%, dan nilai *recall* sebesar 86% serta akurasi sebesar 86%.
3. Waktu inferensi gambar mendapatkan 2 objek yang diperlukan, cenderung stabil dengan variasi kecil di sekitar rata-rata 1.89 detik.
4. Pada tugas ekstraksi karakter dengan *EasyOCR*, hasil yang sangat baik sebesar 100% pada 4 gambar yang dapat mewakili plat nomor polisi yang ada di

Indonesia, meliputi plat hitam diatas putih, putih diatas hitam dan juga plat variasi.

5. Pemrosesan plat dan slot oleh *EasyOCR* juga menunjukkan stabilitas dengan rata-rata masing-masing 0.6 detik dan 0.2 detik.
6. Performa terbaik inferensi sampai ekstraksi yang dicatat adalah total waktu 2.56 detik sedangkan performa terburuk adalah 3.5 detik.
7. Total waktu proses bervariasi namun tetap dalam rentang yang cukup konsisten dengan rata-rata 2.7 detik.

## 5.2. Saran

Adapun saran yang dapat diberikan pada penelitian selanjutnya adalah:

1. Meningkatkan kemampuan sistem pemrosesan citra yang dapat mengenali objek dengan lebih cepat dan akurat.
2. Pada tahap produksi sebaiknya menggunakan perangkat yang dapat menghemat biaya produksi.
3. Untuk tahap produksi, sebaiknya ada lampu penanda status slot agar yang bukan pengguna aplikasi dapat melihat status slot parkir.
4. Untuk dapat diproduksi, aplikasi perlu menggunakan akun multi user dan *database* yang lebih baik agar manajerial penyimpanan data lebih teratur serta meningkatkan keamanan baik dalam pengiriman dan pertukaran informasi dari perangkat ke *database*.

## DAFTAR PUSTAKA

- Anas, M. J., & Sakti, D. V. S. Y. (2020). Perancangan Sistem Aplikasi Booking Parkir Menggunakan Sensor Infrared Berbasis Internet Of Things. *Konferensi Nasional Ilmu Komputer (KONIK)*.
- Aprilino, A., & Amin, I. H. Al. (2022). Implementasi Algoritma Yolo Dan Tesseract OCR Pada Sistem Deteksi Plat Nomor Otomatis. *TEKNOINFO*, 16(2615-224X).
- Basthomi, M. S. Y., & Utaminingrum, F. (2021). Deteksi dan Pengenalan Plat Nama Ruangan menggunakan Faster-RCNN dan Pytesseract pada Purwarupa Kursi Roda Pintar. *Pengembangan Teknologi Informasi Dan Ilmu Komputer*, 5, 820–826. <http://j-ptiik.ub.ac.id>
- Direktorat Jenderal Perhubungan Darat. (1996). *Pedoman Teknis Penyelenggaraan Fasilitas Parkir*.
- Galahartlambang, Y., Khotiah, T., Fanani, Z., & Aprilia Yani Solekhah, A. (2023). Deteksi Plat Nomor Kendaraan Otomatis Dengan Convolutional Neural Network Dan OCR Pada Tempat Parkir ITB Ahmad Dahlan Lamongan. *Jurnal Manajemen Informatika & Sistem Informasi (MISI)*. doi.org/10.36595/misi.v5i2
- Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2013). *Rich feature hierarchies for accurate object detection and semantic segmentation*. arxiv.org/abs/1311.2524
- Firswandy, R. I., Rahayu, S., & Affandi, A. (2019). *Deteksi Plat Nomor Kendaraan Bermotor Menggunakan Sliding Concentric Windows (SCW) Untuk Aplikasi Sistem Transportasi Cerdas*. Institut Teknologi Sepuluh Nopember.
- Istiqlal, A., Zaen, M. T. A., & Pratama, W. W. (2023). Prototype Smart Parking Berbasis IoT. *Jurnal JURTIE*, 5(2), 73–86. doi.org/10.55542/jurtie.v5i2
- Laksono, H. T., & Budiarso, Z. (2023). Rancang Bangun Sistem Smart Parkir Berbasis Arduino. *STRING (Satuan Tulisan Riset Dan Inovasi Teknologi)*, 3, 339–343.
- Musyafira, N., Navastara, D. A., & Faticah, C. (2020). *Deteksi Posisi Dan Pengenalan Plat Nomor Kendaraan Menggunakan Single Shot Detector Dan Recurrent Neural Network Pada Data Video*. Institut Teknologi Sepuluh Nopember.
- Nurdin, P. (2018). *Susahnya Parkir di Kota Besar, 107 Jam Per Tahun Hanya Untuk Mencari Tempat Parkir*. Mobilmo. [mobilmo.com/pasar-mobil/susahnya-parkir-di-kota-besar-107-jam-per-tahun-hanya-untuk-mencari-tempat-parkir-aid3619](https://mobilmo.com/pasar-mobil/susahnya-parkir-di-kota-besar-107-jam-per-tahun-hanya-untuk-mencari-tempat-parkir-aid3619)
- Pratama, R. F., Wicaksono, R. S. R., & Pramudhita, A. N. (2023). Perancangan Dan Implementasi Protokol MQTT Pada Sistem Parkir Cerdas Berbasis IoT. *Jurnal Informatika Dan Teknik Elektro Terapan*. doi.org/10.23960/jitet.v11i3.3191

- Pribadi, T., & Mali, P. Y. (2017). *Survei: Butuh 21 Menit Mencari Tempat Parkir di Jakarta*. Viva. <https://www.viva.co.id/otomotif/mobil/971400-survei-butuh-21-menit-mencari-tempat-parkir-di-jakarta>
- Rudi, R., Dinata, I., & Kurniawan, R. (2017). Rancang Bangun Prototype Sistem Smart Parking Berbasis Arduino Dan Pemantauan Melalui Smartphone. *Jurnal ECOTIPE*, 4(2), 14–20. doi.org/10.33019/ecotipe.v4i2.7
- Sarieff, I., Biu, H. Y., Harisman, F., & Chandra, S. I. (2019). Detection of Vehicles Number Plate Using Image Processing with Template matching Method. *Telekontran : Jurnal Ilmiah Telekomunikasi, Kendali Dan Elektronika Terapan*, 7(1), 14–24. doi.org/10.34010/telekontran.v7i1.1634
- Setiawan, R. (2021, October 9). *Mengenal Deep Learning Lebih Jelas*. Dicoding. [www.dicoding.com/blog/mengenal-deep-learning/](http://www.dicoding.com/blog/mengenal-deep-learning/)
- Sudarti, S., Yushardi, Y., & Kasanah, N. (2022). Analisis Potensi Emisi CO<sub>2</sub> Oleh Berbagai Jenis Kendaraan Bermotor di Jalan Raya Kemantren Kabupaten Sidoarjo. *Jurnal Sumberdaya Alam Dan Lingkungan*, 9(2), 70–75. doi.org/10.21776/ub.jsal.2022.009.02.4
- Tirtana, E., Gunadi, K., & Sugianto, I. (2021). Penerapan Metode YOLO dan Tesseract-OCR untuk Pendataan Plat Nomor Kendaraan Bermotor Umum di Indonesia Menggunakan Raspberry Pi. *Jurnal Infra*, 9.