

**PENENTUAN PARAMETER TERBAIK MODEL SSD
MOBILENET-V2 UNTUK DETEKSI RAMBU LALU LINTAS
PADA RASPBERRY PI 4**



SKRIPSI

**Disusun Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Komputer
pada Departemen Ilmu Komputer/ Informatika**

**Disusun oleh:
Angga Dharma Iswara
24060117130051**

**DEPARTEMEN ILMU KOMPUTER/INFORMATIKA
FAKULTAS SAINS DAN MATEMATIKA
UNIVERSITAS DIPONEGORO**

2022

HALAMAN PERNYATAAN KEASLIAN SKRIPSI

Saya yang bertanda tangan di bawah ini:

Nama : Angga Dharma Iswara

NIM : 24060117130051

Judul : Penentuan Parameter Terbaik Model SSD MobileNet-V2 Untuk Deteksi Rambu Lalu Lintas Pada Raspberry Pi 4

Dengan ini saya menyatakan bahwa dalam skripsi ini tidak terdapat karya yang pernah diajukan untuk memperoleh gelar kesarjanaan di suatu Perguruan Tinggi, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang tertulis diacu dalam naskah ini dan disebutkan di dalam daftar pustaka.

Semarang, 22 Juni 2022



Angga Dharma Iswara

NIM. 24060117130051

HALAMAN PENGESAHAN

Saya yang bertanda tangan di bawah ini:

Nama : Angga Dharma Iswara

NIM : 24060117130051

Judul : Penentuan Parameter Terbaik Model SSD MobileNet-V2 Untuk Deteksi Rambu Lalu Lintas Pada Raspberry Pi 4

Telah diujikan pada sidang skripsi pada tanggal 12 Mei 2022 dan dinyatakan lulus pada tanggal 22 Juni 2022.

Semarang, 22 Juni 2022

Mengetahui,

Ketua Departemen Ilmu Komputer/ Informatika

FSM UNDIP



Dr. Retno Kusumaningrum, S.Si., M.Kom.

NIP. 198104202005012001

Panitia Penguji Skripsi

Ketua,



Guruh Aryotojo, S.Kom., M.Sc.

NIP. 198912272015041002

HALAMAN PENGESAHAN

Judul : Penentuan Parameter Terbaik Model SSD MobileNet-V2 Untuk Deteksi
Rambu Lalu Lintas Pada Raspberry Pi 4
Nama : Angga Dharma Iswara
NIM : 24060117130051

Telah diujikan pada sidang skripsi pada tanggal 12 Mei 2022 dan dinyatakan lulus pada tanggal 22 Juni 2022.

Semarang, 22 Juni 2022

Pembimbing I

Pembimbing II



Prof. Dr. Kusworo Adi, S.Si., M.T.

NIP. 197203171998021001



Aris Sugiharto, S.Si., M.Kom.

NIP. 197108111997021004

ABSTRAK

Salah satu penyebab kecelakaan lalu lintas adalah faktor kesalahan manusia. Faktor ini diawali dengan pelanggaran rambu lalu lintas yang terjadi karena kesengajaan, ketidaktahuan atau pengabaian oleh pengendara. Permasalahan ini dapat diatasi dengan teknologi sistem deteksi rambu lalu lintas secara otomatis untuk memberi peringatan bagi pengendara pada lampu indikator dan tampilan. Sistem ini dikembangkan dengan model deep learning yaitu SSD MobileNet-V2. Model ini memiliki keunggulan pada kecepatan waktu komputasi, tetapi kecepatan ini ditukar dengan tingkat akurasi. Peningkatan akurasi dapat dilakukan dengan augmentasi data dan penentuan *hyperparameter*. Dataset yang digunakan adalah *German Traffic Sign Detection Benchmark* (GTSDB) dengan 4 kelas dasar yaitu *prohibitory*, *danger*, *mandatory*, dan *other*. *Hyperparameter* yang diamati pada penelitian ini adalah *learning rate*, *batch*, dan *epoch*. Nilai *learning rate* yang digunakan yaitu 0.0001, 0.0005, dan 0.005. Nilai *batch* yang digunakan yaitu 2, 4, dan 8. Nilai *epoch* ditentukan dari hasil pelatihan model yang menerapkan *early stopping*. Hasil evaluasi pada CPU memiliki tingkat akurasi tertinggi sebesar 81% dan rata-rata waktu komputasi 0.103s dengan penambahan augmentasi data, *learning rate* 0.0005, *batch* 8, dan *epoch* 26087. Di sisi lain, pengujian pada TPU menggunakan *hyperparameter* yang sama terjadi perubahan rata-rata waktu komputasi menjadi 0.011 s atau 8 kali lipat lebih cepat. Model ini diimplementasikan pada perangkat Raspberry Pi 4 dengan webcam dan tampilan. Skema pengujian secara *realtime* pada CPU memiliki tingkat akurasi sebesar 89.6%, dan rata-rata waktu komputasi 0.107 s. Di sisi lain, pengujian pada TPU terjadi perubahan rata-rata waktu komputasi menjadi 0.018 s atau 6 kali lipat lebih cepat.

Kata kunci : Deteksi Objek, Rambu Lalu Lintas, SSD Mobilenet-V2, Raspberry Pi.

ABSTRACT

One of the causes of traffic accidents is the human error factor. This factor begins with violations of traffic signs that occur due to intentional, ignorance or neglect by the driver. This problem can be overcome with automatic traffic sign detection system technology to warn motorists on indicator lights and displays. This system was developed with a deep learning model, namely SSD MobileNet-V2. This model has the advantage of speed computing time, but this speed is traded for accuracy. Accuracy improvements can be made by augmenting data and hyperparameters tuning. The dataset used is the German Traffic Sign Detection Benchmark (GTSDB) with 4 basic classes, namely prohibitory, danger, mandatory, and other. The hyperparameters observed in this study were learning rate, batch, and epoch. The learning rate values used are 0.0001, 0.0005, and 0.005. The batch values used are 2, 4, and 8. The epoch value is determined from the results of model training that applies early stopping. The evaluation results on the CPU have the highest accuracy rate of 81% and the average computation time is 0.103s with the addition of data augmentation, learning rate 0.0005, batch 8, and epoch 26087. On the other hand, testing on the TPU using the same hyperparameters changes the average the average computing time becomes 0.011 s or 8 times faster. This model is implemented by Raspberry Pi 4 devices with webcams and displays. The realtime testing scheme on the CPU has an accuracy rate of 89.6%, and an average computation time of 0.107 s. On the other hand, the TPU test showed a change in the average computing time to 0.018 s or 6 times faster.

Keyword : Object Detection, Traffic Sign, SSD Mobilenet-V2, Raspberry Pi.

KATA PENGANTAR

Penulis memanjatkan puji syukur kepada Tuhan Yang Maha Esa atas berkat dan rahmat-Nya sehingga penulis dapat menyelesaikan penulisan laporan Skripsi yang berjudul “Penentuan Parameter Terbaik Model SSD MobileNet-V2 Untuk Deteksi Rambu Lalu Lintas Pada Raspberry Pi 4” dengan baik dan lancar. Laporan skripsi ini disusun sebagai salah satu syarat untuk memperoleh gelar sarjana strata satu pada Departemen Ilmu Komputer / Informatika Fakultas Sains dan Matematika Universitas Diponegoro Semarang.

Dalam penyusunan laporan skripsi ini, penulis mendapatkan banyak bimbingan, dan bantuan dari berbagai pihak. Oleh karena itu, penulis ingin menyampaikan rasa terima kasih kepada:

1. Prof. Dr. Widowati, M.Si. selaku Dekan FSM Universitas Diponegoro.
2. Dr. Retno Kusumaningrum, S.Si, M.Kom. selaku Ketua Departemen Ilmu Komputer/ Informatika Universitas Diponegoro.
3. Panji Wisnu Wirawan, S.T, M.T. selaku Koordinator Skripsi.
4. Prof. Dr. Kusworo Adi, S.Si, M.T. selaku dosen pembimbing 1 yang telah menyediakan waktu, tenaga dan pikiran untuk membimbing penulis.
5. Aris Sugiharto, S.Si., M.Kom. selaku dosen pembimbing 2 yang telah menyediakan waktu, tenaga dan pikiran untuk membimbing penulis.
6. Kedua orang tua penulis, Bapak Nurofiq dan Ibu Kundariyah yang senantiasa membimbing, mendukung dan mendoakan penulis.
7. Semua pihak yang telah membantu kelancaran dalam pengerjaan skripsi, yang tidak dapat penulis sebutkan satu persatu.

Penulis menyadari bahwa dalam laporan ini terdapat banyak kekurangan, baik isi materi maupun dalam penyampaian materi. Hal ini disebabkan oleh keterbatasan kemampuan dan pengetahuan dari penulis. Oleh karena itu, penulis sangat menerima kritik dan saran yang bersifat membangun. Penulis berharap laporan ini dapat bermanfaat bagi penulis sendiri dan pembaca pada umumnya.

Semarang, 26 Januari 2022

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI SKRIPSI UNTUK KEPENTINGAN AKADEMIS

Sebagai civitas akademik Universitas Diponegoro, saya yang bertanda tangan di bawah ini:

Nama : Angga Dharma Iswara

NIM : 24060117130051

Program Studi: Informatika

Departemen : Ilmu Komputer/Informatika

Fakultas : Sains dan Matematika

Jenis Karya : Skripsi

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan **Hak Bebas Royalti Noneksklusif** (*Non-exclusive Royalty Free Right*) kepada Universitas Diponegoro atas karya ilmiah saya yang berjudul:

“Penentuan Parameter Terbaik Model SSD MobileNet-V2 Untuk Deteksi Rambu Lalu Lintas Pada Raspberry Pi 4”

Beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Non-Eksklusif ini Universitas Diponegoro berhak menyimpan, mengalihmedia/ formatkan, mengelola dalam meminta izin dari saya selama tetapi mencantumkan nama saya sebagai penulis/ pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Semarang, 22 Juni 2022



Angga Dharma Iswara

NIM. 24060117130051

DAFTAR ISI

HALAMAN PERNYATAAN KEASLIAN SKRIPSI.....	ii
HALAMAN PENGESAHAN	iii
HALAMAN PENGESAHAN	iv
ABSTRAK	v
ABSTRACT	vi
KATA PENGANTAR.....	vii
HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI SKRIPSI UNTUK KEPENTINGAN AKADEMIS	viii
DAFTAR ISI	ix
DAFTAR GAMBAR.....	xii
DAFTAR TABEL	xiv
BAB I PENDAHULUAN	1
1.1. Latar Belakang	1
1.2. Rumusan Masalah	2
1.3. Tujuan dan Manfaat	2
1.4. Ruang Lingkup.....	3
1.5. Sistematika Penulisan.....	3
BAB II TINJAUAN PUSTAKA	5
2.1. Deteksi Objek.....	5
2.2. Rambu Lalu Lintas	6
2.3. State of Art	6
2.4. <i>Convolutional Neural Network (CNN)</i>	8
2.4.1. Operasi Konvolusi	8
2.4.2. <i>Batch Normalization</i>	9
2.4.3. Fungsi Aktivasi ReLU (<i>Rectified Linear Unit</i>)	10
2.4.4. Fungsi Aktivasi Sigmoid.....	11
2.4.5. RMSProp	11
2.5. MobileNet-V2	12
2.5.1. <i>Depthwise Separable Convolutions</i>	12
2.5.2. <i>Linear Bottlenecks dan Inverted Residuals</i>	13

2.6. <i>Single Shot Detector (SSD)</i>	15
2.7. <i>Augmentasi Data</i>	22
2.7.1. <i>Translation</i>	23
2.7.2. <i>Rotation</i>	23
2.7.3. <i>Scaling</i>	24
2.7.4. <i>Reflection / Flipping</i>	24
2.7.5. <i>Shearing</i>	25
2.7.6. <i>Arithmetic</i>	26
2.7.7. <i>Filtering</i>	27
2.8. <i>Evaluation Metrics</i>	28
2.8.1. <i>Intersection Over Union (IOU)</i>	28
2.8.2. <i>Accuracy</i>	29
2.8.3. <i>Precision</i>	29
2.8.4. <i>Recall</i>	29
2.9. <i>Transfer Learning</i>	30
2.10. <i>Python</i>	30
2.11. <i>Tensorflow</i>	30
2.12. <i>OpenCV</i>	31
2.13. <i>Raspberry Pi</i>	31
2.14. <i>Google Coral USB Accelerator</i>	33
BAB III METODOLOGI PENELITIAN	34
3.1. <i>Gambaran Umum Penyelesaian Masalah</i>	34
3.2. <i>Pengumpulan Data</i>	35
3.3. <i>Pra-Pemrosesan Data</i>	36
3.3.1. <i>Konversi Data</i>	36
3.3.2. <i>Augmentasi Data</i>	36
3.3.3. <i>Pembagian Data</i>	48
3.4. <i>Pelatihan Model</i>	48
3.4.1. <i>Persiapan Data</i>	48
3.4.2. <i>Skema Pelatihan Model</i>	49
3.4.3. <i>Pelatihan Model SSD MobileNet-V2</i>	51
3.5. <i>Pengujian Model</i>	75
3.5.1. <i>Skema Pengujian Jupyter Notebook</i>	75

3.5.2. Skema Pengujian Raspberry Pi	76
3.6. Evaluasi Model.....	77
BAB IV HASIL DAN PEMBAHASAN	79
4.1. Implementasi	79
4.1.1. Spesifikasi Perangkat.....	79
4.1.2. Implementasi Penelitian	80
4.1.3. Implementasi Perangkat Raspberry Pi.....	80
4.2. Pengujian Model Melalui Jupyter Notebook Perangkat Raspberry Pi.....	85
4.2.1. Kasus Uji I.....	86
4.2.2. Kasus Uji II.....	86
4.2.3. Kasus Uji III	87
4.2.4. Kasus Uji IV	87
4.2.5. Kasus Uji V	88
4.2.6. Kasus Uji VI.....	88
4.2.7. Kasus Uji VII.....	89
4.2.8. Kasus Uji VIII	90
4.2.9. Kasus Uji IX.....	90
4.2.10. Penentuan <i>Hyperparameter</i> Terbaik	91
4.3. Pengujian Alat Deteksi Rambu Lalu Lintas Secara <i>Realtime</i>	92
BAB V KESIMPULAN DAN SARAN	97
5.1. Kesimpulan.....	97
5.2. Saran.....	98
DAFTAR PUSTAKA.....	99
LAMPIRAN-LAMPIRAN	103

DAFTAR GAMBAR

Gambar 2.1. Perbandingan Area Studi (Adhikari, 2018)	5
Gambar 2.2. Jenis Rambu Lalu Lintas (Houben et al., 2013)	6
Gambar 2.3. <i>Depthwise Separable Convolutions</i> (Dang et al., 2020).....	12
Gambar 2.4. <i>Linear Bottleneck</i> dan <i>Inverted Residual</i> (Sandler et al., 2018)	14
Gambar 2.5. Arsitektur SSD MobileNet-V2	17
Gambar 2.6. <i>Timeline</i> Perkembangan Raspberry Pi (Google, 2022)	32
Gambar 2.7. Susunan GPIO Raspberry Pi 4 (Google, 2022)	32
Gambar 2.8. Goole Coral USB Accelerator (Pinto de Aguiar et al., 2020)	33
Gambar 3.1. Gambaran Umum Penyelesaian Umum	34
Gambar 3.2. Contoh Citra Rambu Lalu Lintas.....	35
Gambar 3.3. Struktur Berkas CSV Awal.....	36
Gambar 3.4. Jumlah Kelas Awal	37
Gambar 3.5. Proses <i>Translasi</i> Citra.....	38
Gambar 3.6. Proses <i>Rotation</i> Citra	39
Gambar 3.7. Proses <i>Scaling</i> Citra.....	40
Gambar 3.8. Proses <i>Reflection / Flipping</i> Citra.....	41
Gambar 3.9. Proses <i>Shearing</i> Citra	42
Gambar 3.10. Proses Penjumlahan Nilai Piksel	43
Gambar 3.11. Proses Pengurangan Nilai Piksel	44
Gambar 3.12. Proses Perkalian Nilai Piksel	45
Gambar 3.13. Proses <i>Average Filter</i>	46
Gambar 3.14. Proses <i>Median Filter</i>	47
Gambar 3.15. Jumlah Kelas Akhir	48
Gambar 3.16. Diagram Alir Persiapan Data.....	49
Gambar 3.17. Struktur Berkas CSV Akhir	49
Gambar 3.18. Diagram Alir Pelatihan Model.....	50
Gambar 3.19. Skema Pengujian Jupyter Notebook	76
Gambar 3.20. Skema Pengujian Jupyter Notebook	77
Gambar 4.1. Susunan Alat	81
Gambar 4.2. Implementasi Tampilan Sistem	85

Gambar 4.3. Hasil Deteksi Rambu Lalu Lintas Secara Tunggal.....	94
Gambar 4.4. Hasil Deteksi Rambu Lalu Lintas Secara Gabungan.....	95
Gambar 4.5. Kondisi Lampu LED Saat Proses Deteksi	96

DAFTAR TABEL

Tabel 2.1. Daftar Penelitian Deteksi Rambu Lalu Lintas Menggunakan <i>Deep Learning</i>	7
Tabel 2.2. Model MobileNet-V2	14
Tabel 2.3. Arsitektur SSD MobileNet-V2	17
Tabel 2.4. <i>Feature Maps</i> SSD MobileNet-V2.....	22
Tabel 3.1. Kasus Uji <i>HyperParameter</i>	78
Tabel 4.1. Susunan GPIO Modul LED DOT Matrix.....	82
Tabel 4.2. Susunan GPIO Modul OLED 0.96 Inch.....	83
Tabel 4.3. Susunan GPIO Tombol Switch	84
Tabel 4.4. Susunan GPIO Kipas.....	84
Tabel 4.5. Hasil Pengujian Kasus Uji I Menggunakan CPU.....	86
Tabel 4.6. Hasil Pengujian Kasus Uji II Menggunakan CPU	86
Tabel 4.7. Hasil Pengujian Kasus Uji III Menggunakan CPU	87
Tabel 4.8. Hasil Pengujian Kasus Uji IV Menggunakan CPU	87
Tabel 4.9. Hasil Pengujian Kasus Uji V Menggunakan CPU	88
Tabel 4.10. Hasil Pengujian Kasus Uji VI Menggunakan CPU	89
Tabel 4.11. Hasil Pengujian Kasus Uji VII Menggunakan CPU.....	89
Tabel 4.12. Hasil Pengujian Kasus Uji VIII Menggunakan CPU	90
Tabel 4.13. Hasil Pengujian Kasus Uji IX Menggunakan CPU	90
Tabel 4.14. Perbandingan Pengujian <i>Hyperparameter</i> Terbaik Menggunakan	91
Tabel 4.15. Hasil Pengujian Secara <i>Realtime</i> Menggunakan CPU dan TPU.....	92

BAB I

PENDAHULUAN

Bab ini membahas mengenai latar belakang, rumusan masalah, tujuan dan manfaat, ruang lingkup dan sistematika penulisan laporan skripsi.

1.1. Latar Belakang

Kecelakaan lalu lintas adalah sebuah peristiwa di jalan yang terjadi secara tidak diduga dan tidak disengaja. Peristiwa ini melibatkan pengguna kendaraan dengan pengguna jalan atau tanpa pengguna jalan yang mengakibatkan korban manusia atau kerugian harta benda (Undang-Undang Republik Indonesia No. 22 Tahun 2009 Tentang Lalu Lintas Dan Angkutan Jalan, 2009). Studi yang dilakukan oleh Komite Nasional Keselamatan Transportasi (KNKT) menyebutkan bahwa salah satu penyebab kecelakaan lalu lintas yaitu faktor manusia. Pada umumnya, kecelakaan lalu lintas didahului dengan pelanggaran rambu-rambu lalu lintas yang dikarenakan kesengajaan, ketidaktahuan atau pengabaian oleh pengemudi (Badan Pengembangan Sumber Daya Manusia Kementrian PUPR, 2016; Herawati, 2019). Salah satu cara untuk mengatasi hal ini dapat dilakukan dengan pendekatan teknologi melalui sebuah sistem.

Adanya perkembangan di dunia teknologi yang begitu pesat sehingga bermunculan berbagai perangkat teknologi yang bersifat *mobile* seperti *smartphone*, laptop, raspberry pi, dan sebagainya. Hal ini tidak menutup kemungkinan untuk mengembangkan sebuah sistem deteksi rambu lalu lintas yang dapat membantu pengemudi. Sistem ini dapat mengambil citra dari sudut pandang pengemudi (menggunakan kamera) kemudian citra diolah oleh sistem untuk mengetahui rambu lalu lintas yang ada di hadapan pengemudi. Apabila rambu lalu lintas terdeteksi, sistem akan memberikan pemberitahuan kepada pengemudi. Sistem ini dapat dikembangkan dengan salah satu metode yaitu *deep learning* untuk pengembangan model deteksi rambu lalu lintas.

Deep learning adalah bagian dari *machine learning* yang merupakan subbagian dari *Artificial Intelligence*. Metode ini mengacu pada arsitektur yang memiliki banyak lapisan tersembunyi untuk mempelajari berbagai fitur dengan berbagai tingkat abstraksi (Patterson & Gibson, 2005; Wani et al., 2019). *Deep learning* dapat digunakan untuk beberapa permasalahan terkait dengan deteksi objek seperti rambu lalu lintas dan lampu lalu lintas.

Metode ini memiliki kinerja yang cukup baik untuk menangani jumlah data yang besar dan memiliki akurasi yang cukup tinggi, walaupun memerlukan waktu dan sumber daya yang begitu besar (Zhang et al., 2017).

Beberapa penelitian terkait deteksi rambu lalu lintas menggunakan *deep learning* dilakukan oleh (Arcos-García et al., 2018) dengan menggunakan arsitektur Faster R-CNN Inception Resnet-V2 yang memiliki akurasi sebesar 95.77% dan menggunakan *dataset* GTSDb (German Traffic Sign Detection Benchmark). Adapun penelitian lain yang dilakukan oleh (Rajendran et al., 2019) menggunakan arsitektur YOLO-V3 yang memiliki akurasi sebesar 92.2% dan penelitian yang dilakukan oleh (Zhao et al., 2018) menggunakan Faster R-CNN yang memiliki akurasi sebesar 65% . Berdasarkan dari ketiga penelitian yang telah dilakukan, arsitektur Faster R-CNN Inception Resnet-V2 memiliki akurasi tertinggi sebesar 95.77% dengan waktu komputasi sebesar 442ms (Arcos-García et al., 2018).

Salah satu arsitektur *deep learning* yang dikembangkan untuk deteksi objek pada *mobile device* yaitu SSD (*Single Shot Detector*) MobileNet-V2. Arsitektur ini memiliki keuntungan dalam kecepatan waktu komputasi karena ukuran *layer* dan parameter yang lebih sedikit sehingga model yang dihasilkan lebih ringan (Sandler et al., 2018). Pada penelitian ini menggunakan arsitektur SSD MobileNet-V2 dengan pemilihan parameter yang sesuai yaitu nilai *learning rate*, nilai *batch*, dan jumlah *epoch* untuk menghasilkan model terbaik. Setelah itu, model terbaik diimplementasikan ke dalam perangkat Raspberry Pi 4.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, maka dapat dirumuskan masalah yaitu pengembangan sistem deteksi rambu lalu lintas menggunakan model SSD MobileNet-V2. Alasan dalam pemilihan model ditinjau dari segi kebutuhan sumber daya komputasi dan kecepatan waktu komputasi. Pengembangan model dilakukan dengan penentuan *hyperparameter* berupa nilai *learning rate*, nilai *batch*, dan jumlah *epoch* terhadap akurasi model. Hasil terbaik dari pengujian model diimplementasikan ke dalam perangkat Raspberry Pi 4 menggunakan tampilan.

1.3. Tujuan dan Manfaat

Tujuan dari penelitian ini adalah pengembangan sistem deteksi rambu lalu lintas menggunakan model SSD MobileNet-V2 dengan pemilihan *hyperparameter* berupa

learning rate, nilai *batch*, dan jumlah *epoch* untuk diimplementasikan ke dalam Raspberry Pi 4.

Manfaat dari penelitian ini adalah sistem deteksi rambu lalu lintas menggunakan model SSD MobileNet-V2 dengan *hyperparameter* terbaik yang diimplementasikan pada perangkat Raspberry Pi 4 sebagai sistem untuk membantu pengendara dalam memberi peringatan rambu lalu lintas.

1.4. Ruang Lingkup

Ruang lingkup penelitian bertujuan untuk memberikan batasan dari penelitian agar penelitian tidak menyimpang dan sesuai dengan target yang diinginkan. Ruang lingkup skripsi adalah sebagai berikut.

1. *Dataset* yang digunakan yaitu *German Traffic Sign Detection Benchmark* (GTSDB) dari website: https://benchmark.ini.rub.de/gtsdb_dataset.html. *Dataset* ini digunakan sebagai tolok ukur / *benchmark* penelitian melalui hasil evaluasi.
2. Hasil deteksi rambu lalu lintas ditandai dengan lokasi dan kelas identifikasi pada objek yang terpilih.
3. Penentuan *hyperparameter* terbaik yang akan dipilih berdasarkan nilai *learning rate*, nilai *batch*, dan jumlah *epoch*.
4. Proses pengujian dan implementasi tampilan dilakukan pada perangkat Raspberry Pi 4 dengan sistem operasi berbasis Linux (Debian 11 / Raspbian OS versi 11).
5. Pengujian secara *realtime* menggunakan citra tambahan yang sesuai dengan *dataset*. Hal ini dilakukan karena rambu lalu lintas pada GTSDB berukuran kecil. *Dataset* dapat diunduh pada <https://www.bast.de/DE/Verkehrstechnik/Fachthemen/v1-verkehrszeichen/vz-download.html>.
6. Evaluasi penelitian ditunjukkan dengan beberapa nilai yaitu akurasi, presisi, *recall*, dan kecepatan waktu komputasi.

1.5. Sistematika Penulisan

Sistematika penulisan memberikan gambaran umum dari laporan skripsi secara urut dan jelas. Berikut ini adalah sistematika penulisan skripsi:

BAB I PENDAHULUAN

Bab ini membahas mengenai latar belakang, rumusan masalah, tujuan dan manfaat, ruang lingkup dan sistematika penulisan laporan skripsi.

BAB II TINJAUAN PUSTAKA

Bab ini membahas mengenai teori-teori yang berhubungan dan mendukung topik atau masalah yang dibahas pada skripsi.

BAB III METODOLOGI PENELITIAN

Bab ini membahas mengenai metodologi penelitian yang dilakukan dalam penyelesaian masalah deteksi rambu lalu lintas.

BAB IV HASIL DAN PEMBAHASAN

Bab ini membahas mengenai implementasi dan hasil yang dilakukan dalam penyelesaian masalah deteksi rambu lalu lintas

BAB V PENUTUP

Bab ini membahas kesimpulan dari penelitian sistem deteksi rambu lalu lintas dan saran untuk pengembangan pada penelitian selanjutnya.

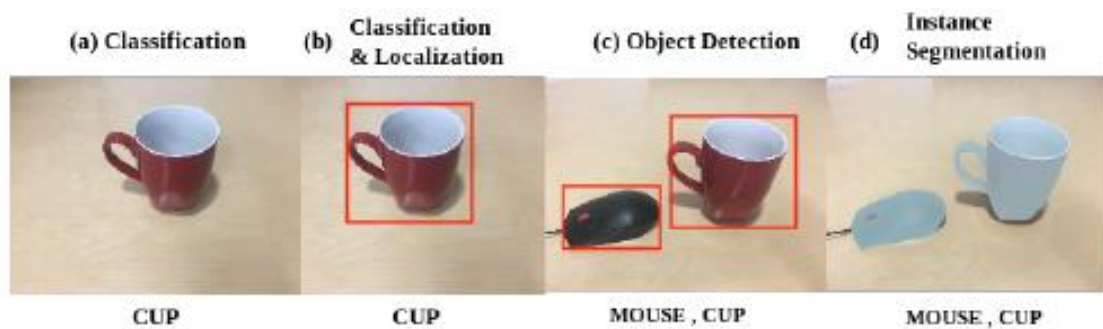
BAB II

TINJAUAN PUSTAKA

Bab ini membahas mengenai teori-teori yang berhubungan dan mendukung topik atau masalah yang dibahas pada skripsi.

2.1. Deteksi Objek

Perkembangan dunia visi komputer menghasilkan beberapa area studi yang berkaitan dengan objek yaitu deteksi (*detection*), klasifikasi (*classification*), lokalisasi (*localization*), dan segmentasi (*segmentation*) (Adhikari, 2018; A. I. Khan & Al-Habsi, 2020). Gambar 2.1. menunjukkan perbandingan antara keempat area studi, (a) klasifikasi kelas objek pada gambar, (b) lokalisasi kelas objek (lokasi dan kelas) pada gambar, (c) kelas dan lokasi masing-masing objek pada gambar, (d) batas piksel masing-masing objek pada gambar.



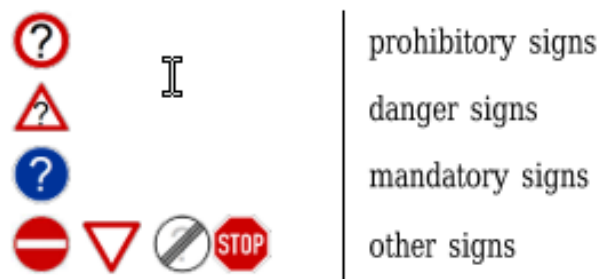
Gambar 2.1. Perbandingan Area Studi (Adhikari, 2018)

Deteksi objek adalah algoritma yang menghasilkan daftar kategori objek dan kotak pembatas pada gambar sebagai indikasi posisi dari masing-masing objek (Russakovsky et al., 2015). Area studi ini banyak digunakan dalam teknologi modern yang digunakan untuk membantu pekerjaan manusia seperti pengembangan aplikasi atau sistem di bidang keamanan, industri, medis, pendidikan, transportasi, dan sebagainya.

Perkembangan studi tentang deteksi objek menghasilkan berbagai ranah riset yang berbasis *machine learning* (pembelajaran mesin) dan *computer vision* (visi komputer). Penerapan studi ini diterapkan pada sebuah *automated system* (sistem otomatis) pada *traffic management* (manajemen lalu lintas) seperti deteksi, pelacakan, klasifikasi dan perhitungan pada lalu lintas (A. I. Khan & Al-Habsi, 2020). Hal ini memungkinkan untuk mengembangkan sebuah sistem yang berbasis deteksi objek guna menjaga keselamatan bagi pengguna jalan, khususnya para pengemudi kendaraan bermotor.

2.2. Rambu Lalu Lintas

Rambu lalu lintas adalah bagian dari perlengkapan jalan berupa lambang, huruf, kalimat, atau perpaduan yang berfungsi sebagai peringatan, larangan, perintah, dan petunjuk bagi pengguna jalan (*Undang-Undang Republik Indonesia No. 22 Tahun 2009 Tentang Lalu Lintas Dan Angkutan Jalan*, 2009). Pada umumnya rambu lalu lintas terbagi menjadi beberapa jenis yaitu peringatan, larangan, perintah, dan petunjuk yang terpasang pada ruas jalan (Perhubungan & Indonesia, 2014). Adapun jenis rambu lalu lintas yang digunakan pada penelitian ini mengacu pada penelitian yang dilakukan oleh (Houben et al., 2013), yaitu *prohibitory* (larangan), *danger* (peringatan/bahaya), *mandatory* (perintah), *other* (lain-lain). Contoh gambar dari masing-masing jenis rambu lintas ditunjukkan pada Gambar 2.2.



Gambar 2.2. Jenis Rambu Lalu Lintas (Houben et al., 2013)

Rambu larangan memiliki ciri-ciri berbentuk lingkaran dengan perpaduan warna merah dan putih, rambu peringatan/bahaya memiliki ciri-ciri segitiga dengan perpaduan warna merah dan putih, rambu perintah memiliki ciri-ciri berbentuk lingkaran dengan perpaduan warna biru dan putih, dan rambu lain-lain memiliki ciri-ciri yang hampir mirip dengan ketiga jenis rambu tetapi memiliki maksud yang berbeda (Houben et al., 2013). Representasi rambu lalu lintas dengan sesuai dengan Gambar 2.2 dapat dilihat pada Lampiran 1.

2.3. State of Art

Deep Learning (DL) adalah salah satu teknologi yang memiliki peranan penting di bidang *Artificial Intelligence* (AI), pembelajaran mesin, dan data besar untuk performa yang luar biasa, DL memiliki kelebihan dalam kinerja yang sangat mengesankan pada visi komputer, pemrosesan sinyal, pengenalan suara, dan pengolahan bahasa alami (Adhikari, 2018). Namun, DL memiliki kekurangan yaitu membutuhkan jumlah *dataset* yang besar dan kemampuan komputasi yang tinggi untuk menghasilkan performa yang baik (Adhikari,

2018). Tabel 2.1 menjabarkan informasi mengenai penelitian deteksi rambu lalu lintas menggunakan DL dan model SSD MobileNet-V2.

Tabel 2.1. Daftar Penelitian Deteksi Rambu Lalu Lintas Menggunakan *Deep Learning*

No.	Penulis	Judul	Objek Penelitian	Hasil
1.	(Arcos-García et al., 2018)	<i>Evaluation of Deep Neural Networks for Traffic Sign Detection Systems</i>	Membandingkan beberapa model <i>deep learning</i> untuk deteksi rambu lalu lintas yaitu Faster R-CNN, R-FCN, SSD (Single Shot Detector), dan YOLO (You Only Look Once) dengan dataset GTSDDB (German Traffic Sign Detection Benchmark)	Model Faster R-CNN Inception Resnet-V2 menghasilkan akurasi terbaik sebesar 95.77%.
2.	(Rajendran et al., 2019)	<i>Real Time Traffic Sign Recognition Using YOLOv3 Based Detector</i>	Pengenalan rambu lalu lintas menggunakan model YOLO-V3 dan lapisan CNN Classifier dengan dataset GTSDDB.	Tingkat akurasi deteksi menggunakan model YOLO-V3 sebesar 92.2% dan akurasi klasifikasi menggunakan lapisan CNN sebesar 96.46%.
3.	(Zhao et al., 2018)	<i>Traffic Sign Detection Based on Faster R-CNN in Scene Graph</i>	Deteksi rambu lalu lintas menggunakan model Faster R-CNN dengan dataset GTSDDB.	Tingkat akurasi deteksi menggunakan model Faster R-CNN sebesar 65%

2.4. Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) atau CovNet atau jaringan saraf konvolusi adalah salah satu teknik *deep learning* yang terinspirasi oleh mekanisme persepsi penglihatan makhluk hidup (Gu et al., 2017). Teknik ini dapat diterapkan pada berbagai kasus seperti klasifikasi gambar, deteksi objek, pengenalan suara, pelacakan objek, dan pengolahan bahasa alami (Gu et al., 2017). Pada studi visi komputer, penerapan CNN dapat diimplementasikan pada robotika, *self-driving*, dan berbagai macam studi yang berkaitan dengan gambar (Reithaug, 2018). CNN terdiri dari 3 buah lapisan, yaitu lapisan masukan (*input*), lapisan tersembunyi (*hidden*), lapisan keluaran (*output*) (Ayi, 2020). Lapisan *input* berupa data dengan dimensi tinggi seperti gambar atau video, lapisan *hidden* berupa operasi konvolusi dan fungsi aktivasi untuk mengolah data input, lapisan *output* berupa hasil keluaran dari pemrosesan oleh lapisan *hidden* (Reithaug, 2018). Berikut ini adalah beberapa operasi yang digunakan dalam CNN.

2.4.1. Operasi Konvolusi

Konvolusi adalah operasi untuk mengekstraksi fitur dari gambar masukan yang dilakukan dengan cara menggabungkan dua set informasi dengan menggunakan perkalian *element-wise product* atau sering dikenal dengan perkalian *dot product* (Reithaug, 2018). Operasi konvolusi terdiri dari beberapa parameter tambahan seperti *kernel (filter)*, *stride*, dan *padding*. *Kernel* atau filter digunakan sebagai bidang pandang dari operasi konvolusi. *Stride* digunakan sebagai ukuran langkah dalam operasi konvolusi. *Padding* digunakan sebagai penambahan batas pada suatu citra (Reithaug, 2018). Persamaan operasi konvolusi dapat dituliskan pada persamaan 2.1 (Wani et al., 2019).

$$F(x, y) = (A * K)(i, j) = \sum_m \sum_n A(m, n)K(i - m, j - n) \dots\dots\dots (2.1)$$

Keterangan:

$F(x, y)$ = Hasil konvolusi

A = Gambar masukan

K = *Kernel / filter*

x, y = Indeks pada hasil konvolusi

m, n = Indeks pada gambar masukan

i, j = Indeks pada *kernel / filter*

2.4.2. Batch Normalization

Batch normalization adalah metode yang digunakan untuk memperbaiki nilai rata-rata dan varian dari sebuah input. Metode ini digunakan pada *neural network* (jaringan saraf) untuk mempercepat waktu pelatihan dan menghasilkan akurasi yang optimal (Ioffe & Szegedy, 2015). *Batch normalization* dapat diterapkan pada jaringan konvolusi sebagai serangkaian fungsi aktivasi yang digabungkan dengan fungsi aktivasi seperti sigmoid atau ReLU (Ioffe & Szegedy, 2015). Persamaan *batch normalization* dapat dituliskan pada persamaan 2.2 sampai 2.5 (Ioffe & Szegedy, 2015).

a. Menghitung nilai rata-rata / *mean*

$$\mu = \frac{1}{m} \sum_{i=1}^m x_i \dots\dots\dots (2.2)$$

Keterangan:

μ = Nilai rata-rata / *mean*
 x_i = Nilai pada indeks ke-i
 i = Indeks
 m = Banyak data

b. Menghitung nilai varian

$$\sigma^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu)^2 \dots\dots\dots (2.3)$$

Keterangan:

σ^2 = Nilai varian
 μ = Nilai rata-rata / *mean*
 x_i = Nilai pada indeks ke-i
 i = Indeks
 m = Banyak data

c. Normalisasi nilai

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \varepsilon}} \dots\dots\dots (2.4)$$

Keterangan:

\hat{x}_i = Hasil normalisasi data pada indeks ke-i
 σ^2 = Nilai varian
 μ = Nilai rata-rata / *mean*
 ε = Nilai konstanta
 x_i = Nilai pada indeks ke-i

i = Indeks

d. Menghitung penskalaan (*scaling*) dan pergeseran (*shifting*)

$$y_i = \gamma \hat{x}_i + \beta \dots\dots\dots (2.5)$$

Keterangan:

y_i = Hasil *batch normalization* pada indeks ke-i

\hat{x}_i = Hasil normalisasi data pada indeks ke-i

γ = Parameter untuk penskalaan

β = Parameter untuk pergeseran

i = Indeks

2.4.3. Fungsi Aktivasi ReLU (*Rectified Linear Unit*)

Fungsi aktivasi ReLU merupakan fungsi aktivasi yang diusulkan oleh Nair dan Hinton pada tahun 2010, fungsi ini melakukan operasi penentuan nilai ambang batas (*threshold*) untuk setiap nilai masukan dengan cara memaksa sebuah nilai yang kurang dari nol diatur sebagai nilai nol dan nilai yang lebih besar sama dengan nol akan memiliki nilai tetap (Nwankpa et al., 2018). Keuntungan dari penggunaan fungsi ini yaitu memiliki waktu komputasi yang cepat karena tidak memerlukan perhitungan eksponensial dan pembagian (Nwankpa et al., 2018). Persamaan fungsi aktivasi ReLU dapat dituliskan pada persamaan 2.6 (Nwankpa et al., 2018).

$$f(x) = \max(0, x) \begin{cases} x, & \text{jika } x \geq 0 \\ 0, & \text{jika } x < 0 \end{cases} \dots\dots\dots (2.6)$$

Keterangan:

$f(x)$ = Hasil ReLU

x = Nilai masukan

Fungsi aktivasi ReLU6 merupakan varian lain dari fungsi aktivasi ReLU, fungsi ini memiliki hasil yang lebih *robust* pada komputasi presisi rendah dibandingkan dengan ReLU (Sandler et al., 2018). Persamaan fungsi aktivasi ReLU6 dapat dituliskan pada persamaan 2.7 (Sandler et al., 2018).

$$f(x) = \min (\max(0, x), 6) \dots\dots\dots (2.7)$$

Keterangan:

$f(x)$ = Hasil ReLU6

x = Nilai masukan

2.4.4. Fungsi Aktivasi Sigmoid

Fungsi aktivasi *Sigmoid* atau logistik adalah fungsi nonlinear yang digunakan pada lapisan keluaran dari arsitektur DL, fungsi ini digunakan untuk menentukan prediksi probabilitas, klasifikasi biner maupun regresi logistik (Nwankpa et al., 2018). Persamaan fungsi aktivasi *sigmoid* dapat dituliskan pada persamaan 2.8 (Nwankpa et al., 2018).

$$f(x) = \frac{1}{(1+e^{-x})} \dots\dots\dots (2.8)$$

Keterangan:

$f(x)$ = Hasil *sigmoid*

e = Nilai *euler* ($e=2,71828$)

x = Nilai masukan

2.4.5. RMSProp

RMSProp merupakan metode pembelajaran adaptif yang diusulkan oleh Geoffrey Hinton pada tahun 2012. Metode ini digunakan untuk menormalisasi nilai gradien supaya membantu proses pengoptimalan pada setiap parameter (Aggarwal, 2018; S. Khan et al., 2018; Tieleman & Hinton, 2012). Persamaan RMSProp dapat dituliskan pada persamaan 2.9 sampai 2.10 (S. Khan et al., 2018).

$$E[\delta^2]_t = \gamma E[\delta^2]_{t-1} + (1 - \gamma) \delta_t^2 \dots\dots\dots (2.9)$$

Keterangan:

$E[\delta^2]$ = Rata-rata perubahan eksponensial terhadap gradien

γ = Nilai peluruhan / *decays* (bernilai konstanta = 0.9)

t = Indeks

δ = Nilai gradien/perubahan

Setelah mencari nilai rata-rata perubahan eksponensial terhadap gradien. dilakukan proses pembaruan parameter yang dituliskan pada persamaan 2.10.

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{E[\delta^2]_t + \epsilon}} \delta_t \dots\dots\dots (2.10)$$

Keterangan:

θ_t = Parameter

t = Indeks

$E[\delta^2]$ = Rata-rata perubahan eksponensial terhadap gradien

α = Laju pembelajaran / *learning rate*

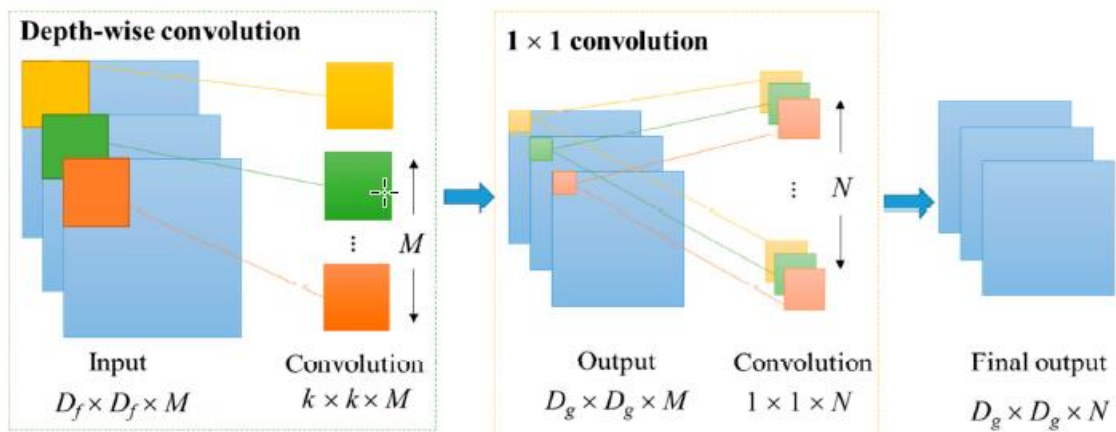
δ = Nilai gradien/perubahan pada waktu t

2.5. MobileNet-V2

MobileNet-V2 adalah model atau arsitektur CNN yang dikembangkan oleh Mark Sandler pada tahun 2018. Model ini dapat diimplementasikan pada perangkat *mobile* atau sistem tertanam yang digunakan sebagai model untuk klasifikasi maupun deteksi (Sandler et al., 2018). MobileNet-V2 dikembangkan untuk meningkatkan kinerja dari versi terdahulu yaitu MobileNet-V1 yang dikenalkan pada tahun 2017. Keunggulan dari MobileNet-V2 terdiri dari beberapa aspek seperti akurasi, ukuran memori dan waktu komputasi dibandingkan dengan Mobilenet-V1 (Sandler et al., 2018). bahkan model ini tetap mempertahankan kesederhanaan dan tidak memerlukan operasi khusus (Sandler et al., 2018). Secara umum operasi Mobilenet-V2 menggunakan operasi dasar yang digunakan MobileNet-V1 yaitu *depthwise separable convolutions* kemudian ditambahkan operasi *linear bottleneck* dan *inverted residuals* (Howard et al., 2017; Sandler et al., 2018).

2.5.1. Depthwise Separable Convolutions

Depthwise Separable Convolutions adalah operasi konvolusi pada model MobileNet yang terdiri dari dua buah lapisan terpisah yaitu *depthwise convolution* dan *pointwise convolution* (Howard et al., 2017). *Depthwise convolution* menerapkan operasi *filtering* untuk setiap *input channel* (nilai kedalaman pada masukan) dan *pointwise convolution* menerapkan operasi *combining* dari hasil keluaran pada lapisan *depthwise convolution*. Kedua lapisan yang digunakan pada model MobileNet menghasilkan pengurangan biaya komputasi dan ukuran model (Howard et al., 2017). Operasi *depthwise separable convolutions* dapat ditunjukkan pada Gambar 2.3.



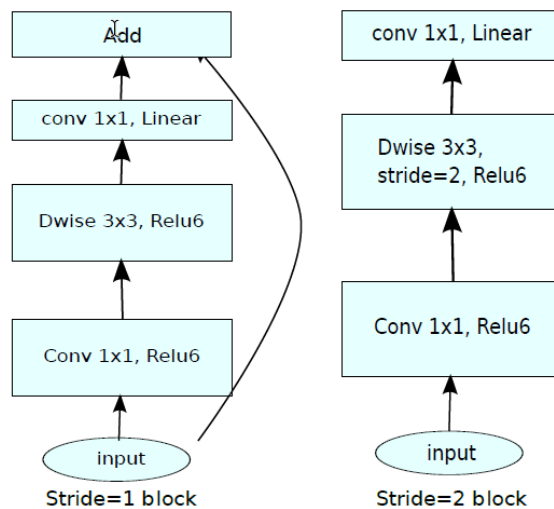
Gambar 2.3. *Depthwise Separable Convolutions* (Dang et al., 2020)

Depthwise convolutions melakukan operasi konvolusi dengan memisahkan semua *channel* pada masukan kemudian setiap *channel* dilakukan operasi konvolusi dengan *kernel* $k \times k \times M$ dan hasil keluaran disusun menjadi satu dengan urutan konvolusi. Contoh sederhana ketika masukan memiliki tiga buah *channel* kemudian dilakukan pemisahan *channel* menjadi tiga buah bagian. Setelah itu, dilakukan operasi konvolusi dengan *kernel* $3 \times 3 \times 1$ pada masing-masing bagian dan hasil keluaran disusun ulang sesuai urutan konvolusi *channel* RGB (Dang et al., 2020).

Pointwise convolutions melakukan operasi konvolusi dari hasil keluaran *depthwise convolutions* dengan *kernel* $1 \times 1 \times N$ dan melibatkan ukuran *channel* pada masukan. Contoh sederhana ketika masukan memiliki tiga buah *channel* kemudian dilakukan operasi konvolusi dengan *kernel* $1 \times 1 \times 3$ maka hasil yang didapatkan yaitu *block* berukuran sama dengan masukan dan jumlah *channel* berubah menjadi satu (Dang et al., 2020).

2.5.2. Linear Bottlenecks dan Inverted Residuals

Model MobileNet-V2 mempertahankan operasi *depthwise separable convolutions* dengan perubahan pada blok penyusun arsitektur, blok ini terdiri dari dua buah bagian yaitu *linear bottlenecks* dan *inverted residuals* (Sandler et al., 2018). Lapisan *linear bottleneck* terdiri dari tiga buah operasi konvolusi yaitu konvolusi dengan *kernel* 1×1 (*expansion layer*), *depthwise convolution* dengan *kernel* 3×3 , dan *pointwise convolution* dengan *kernel* 1×1 (*projection layer*). Operasi pertama dilakukan dengan konvolusi 1×1 yang bertujuan untuk memperbesar jumlah *channel* supaya memperoleh data lebih banyak dengan faktor ekspansi sebesar 6. Hasil operasi ini akan memperbesar jumlah *channel* sebesar 6 kali lipat terhadap ukuran *channel* yang dimiliki oleh masukan. Operasi kedua dilakukan dengan *depthwise convolutions* yang bertujuan untuk memfilter data. Operasi ketiga dilakukan dengan *pointwise convolutions* yang bertujuan mengurangi jumlah *channel*. Lapisan *inverted residual* merupakan lapisan yang terdiri dari lapisan *bottleneck* yang digabungkan dengan *residual connection*. Lapisan ini melakukan proses penjumlahan antara data masukan dengan data hasil *bottleneck* dengan syarat ukuran masing-masing data harus sama. Peran *inverted residual block* digunakan untuk menyamakan ukuran data saat operasi *bottleneck* (Sandler et al., 2018). Gambaran umum mengenai lapisan *linear bottleneck* (ukuran *stride* 2) dan *inverted residual* (ukuran *stride* 1) dapat dilihat pada Gambar 2.4.



Gambar 2.4. *Linear Bottleneck* dan *Inverted Residual* (Sandler et al., 2018)

Setiap operasi konvolusi pada lapisan penyusun MobileNet-V2 diikuti proses *batch normalization* dan ReLU6 kecuali pada bagian *pointwise convolution* di lapisan *bottleneck* tidak menggunakan ReLU6. Alasan tidak diikuti ReLU6 dikarenakan pada operasi *pointwise convolution* menghasilkan jumlah *channel* yang lebih kecil, fungsi aktivasi ReLU dapat menghilangkan informasi penting yang terdapat pada *channel* tersebut (Sandler et al., 2018). Gambaran sederhana mengenai arsitektur model MobileNet-V2 dapat dilihat pada Tabel 2.2.

Tabel 2.2. Model MobileNet-V2

Operator	t (expansion factor)	c (output channel)	n (repeat)	s (stride)
Konvolusi 3x3	-	32	1	2
<i>Bottleneck</i>	1	16	1	1
<i>Bottleneck</i>	6	24	2	2
<i>Bottleneck</i>	6	32	3	2
<i>Bottleneck</i>	6	64	4	2
<i>Bottleneck</i>	6	96	3	1
<i>Bottleneck</i>	6	160	3	2
<i>Bottleneck</i>	6	320	1	1
Konvolusi 1x1	-	1280	1	1

Model MobileNet-V2 terdiri dari 19 lapisan yang diawali dengan operasi konvolusi sebanyak satu kali kemudian diikuti 17 lapisan *bottleneck* dan diakhiri dengan operasi konvolusi 1x1 sebanyak satu kali. Pada permasalahan deteksi objek model MobileNet-V2

dapat digunakan sebagai ekstraksi fitur dan digabungkan dengan algoritma deteksi (Sandler et al., 2018).

2.6. *Single Shot Detector (SSD)*

Single Shot Detector (SSD) adalah model atau arsitektur CNN yang diperkenalkan oleh Wei Liu pada tahun 2016, arsitektur ini dapat digunakan sebagai detektor objek dengan hasil keluaran *bounding box* dan kelas yang memungkinkan lebih dari satu objek (*multiple object detection*) (Agarwal, 2018) . Kelebihan dari model ini yaitu pada kecepatan waktu komputasi sehingga dapat diterapkan pada perangkat *mobile* atau sistem tertanam (Liu et al., 2016). Pada pelatihan model SSD memiliki *loss function* yang dituliskan pada persamaan 2.11 (Liu et al., 2016).

$$L(x, c, l, g) = \frac{1}{N} (L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \dots\dots\dots (2.11)$$

Keterangan:

- L = Nilai loss
- L_{conf} = *Confidence loss / classification loss*
- L_{loc} = *Localization loss / regression loss*
- N = Banyak data yang terdeteksi
- α = Faktor penskalaan
- x = *Prediction label*
- c = *Ground truth label*
- l = *Bounding box location*
- g = *Ground truth location*

Perhitungan nilai *localization loss / regression loss* yang dituliskan pada persamaan 2.12.

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^N \sum_{m \in (cx, cy, w, h)} x_{ij}^k smooth_{L1}(l_i^m - \hat{g}_j^m) \dots\dots\dots (2.12)$$

Keterangan:

- L_{loc} = *Localization loss / regression loss*
- N = Banyak data yang terdeteksi
- i = Indeks
- cx = Titik tengah x
- cy = Titik tengah y
- w = Lebar
- h = Tinggi

x_{ij}^k = Nilai konstanta {0,1}; nilai 1 menunjukkan lokasi sesuai dengan *ground truth*

x = *Prediction label*

l = *Bounding box location*

g = *Ground truth location*

Perhitungan nilai *smooth L1* antara *bounding box* dan *ground truth* yang dituliskan pada persamaan 2.13.

$$smooth\ L1 = \begin{cases} 0.5\ x^2 & ; |x| < 1 \\ |x| - 0.5 & ; otherwise \end{cases} \dots\dots\dots (2.13)$$

Perhitungan nilai *confidence loss / classification loss* yang dituliskan pada persamaan 2.14.

$$L_{conf}(x, c) = -\sum_{i \in Pos}^N x_{ij}^p \log(\hat{c}_i^p) - \sum_{i \in Neg} \log(\hat{c}_i^0) \dots\dots\dots (2.14)$$

Keterangan:

L_{conf} = *Confidence loss / classification loss*

N = Banyak data kelas positif

i = Indeks

x_{ij}^p = Nilai konstanta {0,1}; nilai 1 menunjukkan kelas sesuai dengan *ground truth*

x = *Prediction label*

\hat{c} = Nilai *confidence* sebuah kelas terhadap semua kelas

Perhitungan nilai \hat{c} dilakukan dengan fungsi *softmax* yang dituliskan pada persamaan 2.15.

$$\hat{c}_i^p = \frac{e^{(c_i^p)}}{\sum_p e^{(c_i^p)}} \dots\dots\dots (2.15)$$

Keterangan:

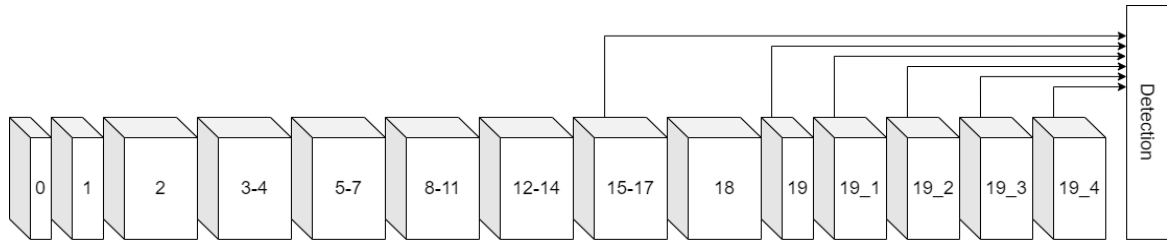
\hat{c}_i^p = Nilai *confidence* sebuah kelas terhadap semua kelas

p = Indeks

exp = Nilai konstanta {0,1}; nilai 1 menunjukkan kelas sesuai dengan *ground truth*

e = Nilai *euler* ($e=2,71828$)

SSD MobileNet-V2 merupakan kombinasi dari dua buah model yang terdiri dari MobileNet-V2 sebagai ekstraksi fitur dan SSD sebagai detektor. Pada model ini menggunakan ukuran input yang menyesuaikan dengan model SSD yaitu 300x300x3. Gambaran mengenai arsitektur SSD MobileNet-V2 diambil dari penelitian (Fan et al., 2018) dan disusun ulang oleh penulis dapat dilihat pada Gambar 2.5.



Gambar 2.5. Arsitektur SSD MobileNet-V2

SSD memiliki 6 buah *feature map* yang diambil dari 2 buah *layer* pada jaringan MobileNet-V2 dan 4 buah blok tambahan setelah *layer* terakhir pada jaringan MobileNet-V2 (Tramontano et al., 2018). Setiap blok tambahan terdiri dari 2 buah konvolusi yaitu konvolusi 1x1 dan konvolusi 3x3. Pertama, *feature map* didapatkan dari konvolusi pertama (*expansion layer*) pada *bottleneck* ke 14 (blok 15). Kedua, *feature map* didapatkan dari *layer* konvolusi 1x1 pada blok ke 19 (lapisan terakhir dari MobileNet-V2). *Feature map* ketiga sampai keenam diambil dari 4 buah blok tambahan pada blok 19_1, 19_2, 19_3, dan 19_4. Setiap *feature map* yang diambil akan dikonvolusikan dengan konvolusi 1x1 pada bagian akhir (Tramontano et al., 2018). Hasil akhir dari proses ini akan mendapatkan prediksi *bounding box* dan kelas. Gambaran detail mengenai arsitektur SSD MobileNet-V2 dapat dilihat pada Tabel 2.3.

Tabel 2.3. Arsitektur SSD MobileNet-V2

No.	Blok	Operasi	Input	Filter	Output	s	Keterangan
0	Input	-	1x300x300x3	-	1x300x300x3	-	Lapisan Input
1	Konvolusi	CONV 3×3	1x300x300x3	32x3x3x3	1x150x150x32	2	Konvolusi 3x3
2	Bottleneck 1	DC	1x150x150x32	1x3x3x32	1x150x150x32	1	First Bottleneck / DSC
		BN					
		ReLU6					
		PC	1x150x150x32	16x1x1x32	1x150x150x16	1	
		BN					
3-4	Bottleneck 2	EC	1x150x150x16	96x1x1x16	1x150x150x96	1	Linear Bottleneck
		BN					
		ReLU6					
		DC	1x150x150x96	1x3x3x96	1x75x75x96	2	
		BN					
		ReLU6					
		PC	1x75x75x96	24x1x1x96	1x75x75x24	1	
		BN					

No.	Blok	Operasi	Input	Filter	Output	s	Keterangan
	Bottleneck 3	EC	1x75x75x24	144x1x1x24	1x75x75x144	1	Inverted Residuals Bottleneck
		BN					
		ReLU6					
		DC	1x75x75x144	1x3x3x144	1x75x75x144	1	
		BN					
		ReLU6					
		PC	1x75x75x144	24x1x1x144	1x75x75x24	1	
		BN					
5-7	Bottleneck 4	EC	1x75x75x24	144x1x1x24	1x75x75x144	1	Linear Bottleneck
		BN					
		ReLU6					
		DC	1x75x75x144	1x3x3x144	1x38x38x144	2	
		BN					
		ReLU6					
		PC	1x38x38x144	32x1x1x144	1x38x38x32	1	
		BN					
	Bottleneck 5	EC	1x38x38x32	192x1x1x32	1x38x38x192	1	Inverted Residuals Bottleneck
		BN					
		ReLU6					
		DC	1x38x38x192	1x3x3x192	1x38x38x192	1	
		BN					
		ReLU6					
		PC	1x38x38x192	32x1x1x192	1x38x38x32	1	
		BN					
	Bottleneck 6	EC	1x38x38x32	192x1x1x32	1x38x38x192	1	Inverted Residuals Bottleneck
		BN					
		ReLU6					
		DC	1x38x38x192	1x3x3x192	1x38x38x192	1	
		BN					
		ReLU6					
		PC	1x38x38x192	32x1x1x192	1x38x38x32	1	
		BN					
8-11	Bottleneck 7	EC	1x38x38x32	192x1x1x32	1x38x38x192	1	Linear Bottleneck
		BN					
		ReLU6					

No.	Blok	Operasi	Input	Filter	Output	s	Keterangan
		DC	1x38x38x192	1x3x3x192	1x19x19x192	2	
		BN					
		ReLU6					
		PC	1x19x19x192	64x1x1x192	1x19x19x64	1	
		BN					
	Bottleneck 8	EC	1x19x19x64	384x1x1x64	1x19x19x384	1	Inverted Residuals Bottleneck
		BN					
		ReLU6					
		DC	1x19x19x384	1x3x3x384	1x19x19x384	1	
		BN					
		ReLU6					
		PC	1x19x19x384	64x1x1x384	1x19x19x64	1	
		BN					
	Bottleneck 9	EC	1x19x19x64	384x1x1x64	1x19x19x384	1	Inverted Residuals Bottleneck
		BN					
		ReLU6					
		DC	1x19x19x384	1x3x3x384	1x19x19x384	1	
		BN					
		ReLU6					
		PC	1x19x19x384	64x1x1x384	1x19x19x64	1	
		BN					
	Bottleneck 10	EC	1x19x19x64	384x1x1x64	1x19x19x384	1	Inverted Residuals Bottleneck
		BN					
		ReLU6					
		DC	1x19x19x384	1x3x3x384	1x19x19x384	1	
		BN					
		ReLU6					
		PC	1x19x19x384	64x1x1x384	1x19x19x64	1	
		BN					
12- 14	Bottleneck 11	EC	1x19x19x64	384x1x1x64	1x19x19x384	1	Linear Bottleneck
		BN					
		ReLU6					
		DC	1x19x19x384	1x3x3x384	1x19x19x384	1	
		BN					
		ReLU6					
		PC	1x19x19x384	96x1x1x384	1x19x19x96	1	
		BN					

No.	Blok	Operasi	Input	Filter	Output	s	Keterangan
	Bottleneck 12	EC	1x19x19x96	576x1x1x96	1x19x19x576	1	Inverted Residuals Bottleneck
		BN					
		ReLU6					
		DC	1x19x19x576	1x3x3x576	1x19x19x576	1	
		BN					
		ReLU6					
		PC	1x19x19x576	96x1x1x576	1x19x19x96	1	
		BN					
	Bottleneck 13	EC	1x19x19x96	576x1x1x96	1x19x19x576	1	Inverted Residuals Bottleneck
		BN					
		ReLU6					
		DC	1x19x19x576	1x3x3x576	1x19x19x576	1	
		BN					
		ReLU6					
		PC	1x19x19x576	96x1x1x576	1x19x19x96	1	
		BN					
15- 17	Bottleneck 14	EC	1x19x19x96	576x1x1x96	1x19x19x576	1	Linear Bottleneck
		BN					
		ReLU6					
		DC	1x19x19x576	1x3x3x576	1x10x10x576	2	
		BN					
		ReLU6					
		PC	1x10x10x576	160x1x1x576	1x10x10x160	1	
		BN					
	Bottleneck 15	EC	1x10x10x160	960x1x1x160	1x10x10x960	1	Inverted Residuals Bottleneck
		BN					
		ReLU6					
		DC	1x10x10x960	1x3x3x960	1x10x10x960	1	
		BN					
		ReLU6					
		PC	1x10x10x960	160x1x1x960	1x10x10x160	1	
		BN					
Bottleneck 16	EC	1x10x10x160	960x1x1x160	1x10x10x960	1	Inverted Residuals Bottleneck	
	BN						
	ReLU6						

No.	Blok	Operasi	Input	Filter	Output	s	Keterangan	
		DC	1x10x10x960	1x3x3x960	1x10x10x960	1		
		BN						
		ReLU6						
		PC	1x10x10x960	160x1x1x960	1x10x10x160	1		
		BN						
18	Bottleneck 17	EC	1x10x10x160	960x1x1x160	1x10x10x960	1	Linear Bottleneck	
		BN						
		ReLU6						
		DC	1x10x10x960	1x3x3x960	1x10x10x960	1		
		BN						
		ReLU6						
		PC	1x10x10x960	320x1x1x960	1x10x10x320	1		
		BN						
19	Konvolusi	CONV 1×1	1x10x10x320	1280x1x1x320	1x10x10x1280	1	Konvolusi 1x1	
19_1	Extra Block 1	CONV 1×1	1x10x10x1280	256x1x1x1280	1x10x10x256	1	Extra Block	
		CONV 3×3	1x10x10x256	512x3x3x256	1x5x5x512	2		
19_2	Extra Block 2	CONV 1×1	1x5x5x512	128x1x1x512	1x5x5x128	1	Extra Block	
		CONV 3×3	1x5x5x128	256x3x3x128	1x3x3x256	2		
19_3	Extra Block 3	CONV 1×1	1x3x3x256	128x1x1x256	1x3x3x128	1	Extra Block	
		CONV 3×3	1x3x3x128	256x3x3x128	1x2x2x256	2		
19_4	Extra Block 4	CONV 1×1	1x2x2x256	64x1x1x256	1x2x2x64	1	Extra Block	
		CONV 3×3	1x2x2x64	128x3x3x64	1x1x1x128	2		
Keterangan:								
CONV = Operasi Konvolusi								
DSC = Depthwise Separable Convolution								
EC = Expansion Convolution								
BN = Batch Normalization								
ReLU6 = Fungsi aktivasi ReLU6								
DC = Depthwise Convolution								
PC = Pointwise Convolution								
Blok Kuning = Nilai stride yang sesuai dengan arsitektur MobileNet-V2								
Blok Orange = Feature map yang diambil dari hasil konvolusi untuk deteksi objek								

Proses pengambilan *feature maps* akan dikonvolusikan dengan sebuah kernel berukuran 1x1 sebanyak 1 kali dan proses ini menghasilkan prediksi *bounding box* dan probabilitas masing-masing kelas. Gambaran mengenai proses pengambilan *feature maps* dapat dilihat pada Tabel 2.4 (Chiu et al., 2020; Tramontano et al., 2018).

Tabel 2.4. *Feature Maps* SSD MobileNet-V2

<i>Feature maps</i>	Blok Input	Input	Filter	Output	s	<i>Reshape</i>	Prediksi
1	<i>Bottleneck 14</i>	1x19x19x576	12x1x1x576	1x19x19x12	1	1x1083x1x4	<i>Bounding Box</i>
		1x19x19x576	15x1x1x576	1x19x19x15	1	1x1083x1x4	<i>Class Probability</i>
2	<i>Layer 19</i>	1x10x10x1280	24x1x1x1280	1x10x10x24	1	1x600x1x4	<i>Bounding Box</i>
		1x10x10x1280	30x1x1x1280	1x10x10x30	1	1x600x1x5	<i>Class Probability</i>
3	<i>Extra Blok 1</i>	1x5x5x512	24x1x1x512	1x5x5x24	1	1x150x1x4	<i>Bounding Box</i>
		1x5x5x512	30x1x1x512	1x5x5x30	1	1x150x1x5	<i>Class Probability</i>
4	<i>Extra Blok 2</i>	1x3x3x256	24x1x1x256	1x3x3x24	1	1x54x1x4	<i>Bounding Box</i>
		1x3x3x256	30x1x1x256	1x3x3x30	1	1x54x1x5	<i>Class Probability</i>
5	<i>Extra Blok 3</i>	1x2x2x256	24x1x1x256	1x2x2x24	1	1x24x1x4	<i>Bounding Box</i>
		1x2x2x256	30x1x1x256	1x2x3x30	1	1x24x1x5	<i>Class Probability</i>
6	<i>Extra Blok 4</i>	1x1x1x128	24x1x1x128	1x1x1x24	1	1x6x1x4	<i>Bounding Box</i>
		1x1x1x128	30x1x1x128	1x1x1x30	1	1x6x1x5	<i>Class Probability</i>

2.7. Augmentasi Data

Augmentasi data adalah salah satu teknik yang digunakan untuk meningkatkan ukuran dan kualitas *dataset* sehingga model *deep learning* dapat dibangun lebih baik, teknik ini dapat dilakukan ketika memiliki data yang berjumlah sedikit atau data yang tidak berimbang (Shorten & Khoshgoftaar, 2019). Hal ini dilakukan untuk mengurangi atau menghindari salah satu permasalahan dalam pengembangan model *deep learning* yaitu *overfitting* (Shorten & Khoshgoftaar, 2019). Pada visi komputer, teknik augmentasi data dilakukan dengan cara manipulasi gambar. Berikut ini adalah contoh dari manipulasi gambar pada augmentasi data.

2.7.1. Translation

Translation atau translasi adalah suatu operasi yang menyebabkan perpindahan objek dari suatu tempat ke tempat yang lain. Pada manipulasi gambar, perpindahan berlaku dalam arah yang sejajar dengan sumbu X dan Y. Objek akan berpindah dari suatu titik koordinat ke titik koordinat baru dan ruang tersisa dapat diisi dengan nilai konstanta (Hadi, 2014). Operasi translasi dalam koordinat 2-D menunjukkan perhitungan satu buah *channel* pada ruang warna RGB, dapat dituliskan pada persamaan 2.16 sampai 2.18.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} tx \\ ty \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix} \dots\dots\dots (2.16)$$

$$x' = tx + x \dots\dots\dots (2.17)$$

$$y' = ty + y \dots\dots\dots (2.18)$$

Keterangan:

- x = Nilai awal koordinat x
- y = Nilai awal koordinat y
- x' = Nilai baru pada koordinat x
- y' = Nilai baru pada koordinat y
- tx = Nilai perpindahan untuk koordinat x
- ty = Nilai perpindahan untuk koordinat y

2.7.2. Rotation

Rotation atau rotasi adalah suatu operasi yang menyebabkan objek bergerak berputar pada sumbu putar berdasarkan sudut putaran tertentu. Pada perputaran mengacu pada besaran sudut 0-360 derajat dengan arah perputaran searah jarum jam maupun berlawanan arah jarum jam (Hadi, 2014). Operasi rotasi dalam koordinat 2-D menunjukkan perhitungan satu buah *channel* pada ruang warna RGB, dapat dituliskan pada persamaan 2.19 sampai 2.21.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x - x_p \\ y - y_p \end{bmatrix} + \begin{bmatrix} x_p \\ y_p \end{bmatrix} \dots\dots\dots (2.19)$$

$$x' = (x - x_p) \cos\theta - (y - y_p) \sin\theta + x_p \dots\dots\dots (2.20)$$

$$y' = (x - x_p) \sin\theta + (y - y_p) \cos\theta + y_p \dots\dots\dots (2.21)$$

Keterangan:

- x = Nilai awal koordinat x
- y = Nilai awal koordinat y

- x_p = Nilai pusat koordinat x
- y_p = Nilai pusat koordinat y
- x' = Nilai baru pada koordinat x
- y' = Nilai baru pada koordinat y
- θ = Nilai besaran sudut perputaran

2.7.3. *Scaling*

Scaling atau penskalaan adalah suatu operasi yang menyebabkan perubahan ukuran objek menjadi lebih kecil atau besar bergantung pada faktor penskalaan. Penskalaan dapat dilakukan berdasarkan sumbu yang dituju yaitu X dan Y. Nilai penskalaan lebih dari 1 mengakibatkan objek pada gambar akan berubah menjadi besar dan sebaliknya (Hadi, 2014). Operasi penskalaan dalam koordinat 2-D menunjukkan perhitungan satu buah *channel* pada ruang warna RGB, dapat dituliskan pada persamaan 2.22 sampai 2.24.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix} \begin{bmatrix} x - x_p \\ y - y_p \end{bmatrix} + \begin{bmatrix} x_p \\ y_p \end{bmatrix} \dots\dots\dots (2.22)$$

$$x' = Sx (x - x_p) + x_p \dots\dots\dots (2.23)$$

$$y' = Sy (y - y_p) + y_p \dots\dots\dots (2.24)$$

Keterangan:

- x = Nilai awal koordinat x
- y = Nilai awal koordinat y
- x_p = Nilai pusat koordinat x
- y_p = Nilai pusat koordinat y
- x' = Nilai baru pada koordinat x
- y' = Nilai baru pada koordinat y
- Sx = Faktor penskalaan terhadap sumbu x
- Sy = Faktor penskalaan terhadap sumbu y

2.7.4. *Reflection / Flipping*

Reflection (refleksi) atau *Flipping* (pembalikan) adalah suatu operasi yang menciptakan pencerminan dari suatu objek. Proses ini dapat dilakukan berdasarkan sumbu yang dituju yaitu X dan Y. Objek pada gambar akan mengalami perubahan arah dari letak aslinya sesuai dengan pemilihan sumbu baik horizontal maupun vertikal (Hadi, 2014).

Operasi refleksi/flipping dalam koordinat 2-D menunjukkan perhitungan satu buah *channel* pada ruang warna RGB, dapat dituliskan pada persamaan 2.25 sampai 2.30.

a. Terhadap sumbu X

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x - x_p \\ y - y_p \end{bmatrix} + \begin{bmatrix} x_p \\ y_p \end{bmatrix} \dots\dots\dots (2.25)$$

$$x' = (x - x_p) + x_p \dots\dots\dots (2.26)$$

$$y' = -(y - y_p) + y_p \dots\dots\dots (2.27)$$

Keterangan:

- x = Nilai awal koordinat x
- y = Nilai awal koordinat y
- x_p = Nilai pusat koordinat x
- y_p = Nilai pusat koordinat y
- x' = Nilai baru pada koordinat x
- y' = Nilai baru pada koordinat y

b. Terhadap sumbu Y

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x - x_p \\ y - y_p \end{bmatrix} + \begin{bmatrix} x_p \\ y_p \end{bmatrix} \dots\dots\dots (2.28)$$

$$x' = -(x - x_p) + x_p \dots\dots\dots (2.29)$$

$$y' = (y - y_p) + y_p \dots\dots\dots (2.30)$$

Keterangan:

- x = Nilai awal koordinat x
- y = Nilai awal koordinat y
- x_p = Nilai pusat koordinat x
- y_p = Nilai pusat koordinat y
- x' = Nilai baru pada koordinat x
- y' = Nilai baru pada koordinat y

2.7.5. *Shearing*

Shearing adalah bentuk transformasi yang membuat distorsi dari bentuk suatu objek, seperti menggeser salah satu sisi tertentu. Operasi ini terdiri dari dua jenis yaitu *shearing* terhadap sumbu X dan Y (Hadi, 2014). Operasi *shearing* dalam koordinat 2-D menunjukkan perhitungan satu buah *channel* pada ruang warna RGB, dapat dituliskan pada 2.31 sampai 2.36.

a. Terhadap sumbu X

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & Shx \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \dots\dots\dots (2.31)$$

$$x' = x + Shx \ y \dots\dots\dots (2.32)$$

$$y' = y \dots\dots\dots (2.33)$$

Keterangan:

x = Nilai awal koordinat x

y = Nilai awal koordinat y

x' = Nilai baru pada koordinat x

y' = Nilai baru pada koordinat y

Shx = Faktor *shearing* terhadap sumbu x

b. Terhadap sumbu Y

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ Shy & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \dots\dots\dots (2.34)$$

$$x' = x_p \dots\dots\dots (2.35)$$

$$y' = Shy \ x + y \dots\dots\dots (2.36)$$

Keterangan:

x = Nilai awal kooordinat x

y = Nilai awal koordinat y

x' = Nilai baru pada koordinat x

y' = Nilai baru pada koordinat y

Shy = Faktor *shearing* terhadap sumbu y

2.7.6. Arithmetic

Arithmetic atau aritmetika adalah operasi yang sederhana dalam manipulasi gambar dengan memanfaatkan operator aritmetika. Operasi aritmetika melakukan perhitungan pada setiap piksel dalam gambar dengan nilai konstanta (Mcandrew, 2004). Persamaan yang digunakan dalam operasi aritmetika dapat dituliskan pada persamaan 2.37 sampai 2.41.

a. Penjumlahan/Pengurangan

$$y = x \pm C \dots\dots\dots (2.37)$$

Keterangan:

y = Nilai baru

x = Nilai awal

C = Nilai konstanta

b. Perkalian

$$y = C x \dots\dots\dots (2.38)$$

Keterangan:

y = Nilai baru

x = Nilai awal

C = Nilai konstanta

Pada perhitungan aritmetika dilakukan proses *thresholding* terhadap hasil dengan persamaan di bawah ini.

$$y = \begin{cases} 255, x > 255 \\ 0, x < 0 \end{cases} \dots\dots\dots (2.39)$$

Keterangan:

y = Nilai baru

x = Nilai awal

2.7.7. Filtering

Filtering atau penapisan adalah operasi pemrosesan gambar yang digunakan untuk mempertajam dan mengaburkan gambar. Filter bekerja dengan menggeser matriks $n \times n$ melintasi gambar dengan karakteristik kernel yang digunakan (Shorten & Khoshgoftaar, 2019). *Filter blur* digunakan untuk memperbaiki kualitas gambar yaitu dengan mengurangi *noise* / derau pada gambar (Hadi, 2014). Di bawah ini beberapa contoh persamaan dalam operasi penapisan untuk *blurring* yang dapat dituliskan pada persamaan 2.40 sampai 2.41.

a. *Average Filter*

$$f(x, y)' = \frac{1}{9} \sum_{p=-1}^1 \sum_{q=-1}^1 f(x + p, y + q) \dots\dots\dots (2.40)$$

Keterangan:

$f(x, y)'$ = Nilai akhir

$f(x, y)$ = Nilai awal pada indeks (x, y)

x, y = Indeks pada input

p, q = Indeks pergeseran

b. *Median Filter*

$$f(x, y)' = \text{median}(f(x - 1, y - 1), f(x - 1, y), f(x - 1, y + 1), f(x, y - 1), f(x, y), f(x, y + 1), f(x + 1, y - 1), f(x + 1, y), f(x + 1, y + 1)) \dots\dots\dots (2.41)$$

Pada penggunaan ukuran filter memungkinkan untuk ukuran penapis lebih besar sehingga menyebabkan gambar lebih *blur* atau kabur.

Keterangan:

$f(x, y)'$ = Nilai akhir suatu piksel

$f(x, y)$ = Nilai suatu piksel pada indeks (x,y)

x, y = Indeks pada input

2.8. Evaluation Metrics

Evaluation metrics atau metrik evaluasi digunakan untuk mengetahui atau mengukur kinerja model deteksi terhadap objek yang harus dideteksi (Padilla et al., 2020). Pada kasus deteksi objek, terdapat beberapa kriteria evaluasi yang digunakan yaitu *True Positive* (TP), *False Positive* (FP), dan *False Negative* (FN), TP berarti objek terdeteksi benar sesuai kotak pembatas *ground truth*, FP berarti objek terdeteksi salah atau tidak sesuai dengan kotak pembatas *ground truth*, FN berarti objek tidak terdeteksi (Magalhães et al., 2021). Dalam konteks deteksi objek hasil *True Negative* (TN) tidak berlaku karena adanya kemungkinan jumlah kotak pembatas yang tidak terbatas terhadap suatu gambar (Padilla et al., 2020). Berikut ini adalah beberapa contoh matriks evaluasi yang digunakan dalam deteksi objek.

2.8.1. Intersection Over Union (IOU)

Intersection Over Union (IOU) atau *Jaccard Index* adalah metode yang digunakan untuk mengetahui kesamaan antar dua set data, metode ini dapat dilakukan dengan menghitung luas perpotongan antara prediksi kotak pembatas dengan kotak pembatas *ground truth* dibagi dengan gabungan area keduanya (Alsing, 2018; Melinte et al., 2020). IOU dapat dituliskan pada persamaan 2.42 (Alsing, 2018).

$$IOU = \frac{\text{Area of Overlap}}{\text{Area of Union}} = \frac{\text{Area}(B \cap B_{gt})}{\text{Area}(B \cup B_{gt})} \dots\dots\dots (2.42)$$

Keterangan:

IOU = *Intersection over Union*

B = Kotak pembatas prediksi

B_{gt} = Kotak pembatas *ground truth*

Kriteria dalam penentuan evaluasi model deteksi memiliki beberapa persyaratan yaitu nilai ambang batas IOU dan kelas prediksi. Model deteksi dinyatakan sebagai TP apabila IoU lebih besar atau sama dengan 0.5 dan kelas prediksi sesuai dengan *ground truth*. FP

apabila model deteksi tidak memenuhi salah satu dari kriteria yang diberikan. FN apabila model deteksi tidak dapat mendeteksi objek (Melinte et al., 2020).

2.8.2. *Accuracy*

Accuracy atau akurasi adalah hasil perbandingan antara jumlah terdeteksi benar dengan seluruh hasil pengamatan (Adhikari, 2018; Pichaikutty, 2020). Nilai *accuracy* dapat dituliskan pada persamaan 2.43 (Adhikari, 2018).

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \dots\dots\dots (2.43)$$

Keterangan:

TP = *True Positive*

FP = *False Positive*

FN = *False Negative*

TN = *True Negative*

2.8.3. *Precision*

Precision atau presisi adalah hasil perbandingan antara jumlah terdeteksi benar dengan total pengamatan yang terdeteksi (positif) (Adhikari, 2018; Pichaikutty, 2020). Nilai *precision* dapat dituliskan pada persamaan 2.44 (Adhikari, 2018).

$$Precision = \frac{TP}{TP+FP} \dots\dots\dots (2.44)$$

Keterangan:

TP = *True Positive*

FP = *False Positive*

2.8.4. *Recall*

Recall atau penarikan adalah hasil perbandingan antara jumlah terdeteksi benar dengan total data sesuai *ground truth* (Adhikari, 2018; Pichaikutty, 2020). Nilai *recall* dapat dituliskan pada persamaan 2.45 (Adhikari, 2018).

$$Recall = \frac{TP}{TP+FN} \dots\dots\dots (2.45)$$

Keterangan:

TP = *True Positive*

FN = *False Negative*

2.9. *Transfer Learning*

Transfer Learning (TL) adalah salah satu paradigma yang digunakan untuk mencegah *overfitting* (Shorten & Khoshgoftaar, 2019). Paradigma ini digunakan untuk mengatasi masalah waktu dan sumber daya dalam pelatihan model CNN supaya menghasilkan kinerja terbaik (Alsing, 2018). Adanya paradigma ini, pengembangan model tidak memerlukan proses pembelajaran dari awal, melainkan menggunakan model yang sudah diimplementasikan pada *dataset* besar seperti ImageNet, Pascal VOC, Kitti, dan MS COCO (Athanasiadis et al., 2018). TL menggunakan pengetahuan dari pelatihan yang sudah dilakukan dalam *dataset* besar kemudian pengetahuan model (berupa parameter atau fitur) ditransferkan untuk pelatihan *dataset* baru dalam menyelesaikan masalah yang berbeda (Lamprousi, 2021). Pada penelitian ini menggunakan model *pre-trained* yang sudah diimplementasikan pada *dataset* MS COCO dengan percobaan beberapa parameter seperti nilai *learning rate*, *batch*, dan jumlah *epoch* untuk menentukan hasil model terbaik.

2.10. Python

Python adalah bahasa pemrograman yang dikembangkan oleh Guido van Rossum di *National Research Institute*, Belanda pada akhir tahun 80-an dan akhirnya rilis pada tahun 1991, bahasa pemrograman ini dapat digunakan dalam berbagai pekerjaan seperti pembuatan game, *website*, dan GUI (*Graphical User Interface*) (Thompson, 2016). Salah satu aspek yang membuat Python menjadi bahasa pemrograman yang populer yaitu dukungan dari berbagai macam *library* yang dapat digunakan oleh pengguna dengan memberikan pernyataan impor sederhana. (Thompson, 2016). Python juga mendukung berbagai sistem operasi yang biasa digunakan seperti Windows, Linux dan MacOS sehingga mudah digunakan untuk berbagai perangkat komputer. Saat ini Python memiliki 2 buah versi dasar yaitu 2.x dan 3.x dengan peningkatan versi melalui pembaruan dalam jangka waktu tertentu.

2.11. Tensorflow

Tensorflow adalah *open-source library* yang dikembangkan oleh Google pada November 2015, *library* ini mendukung berbagai macam algoritma *machine learning* dan *deep learning* yang dapat digunakan oleh pengguna untuk mengimplementasikan model pada aplikasi bisnis maupun keperluan riset (Géron, n.d.; Singh & Manure, 2020). Tensorflow dibangun dengan dua bahasa pemrograman yaitu C++ dan Python yang dapat digunakan dalam berbagai sistem operasi seperti Windows, Linux dan MacOS (Hope et al., 2017). Secara umum Tensorflow dirancang dengan mempertimbangkan aspek portabilitas

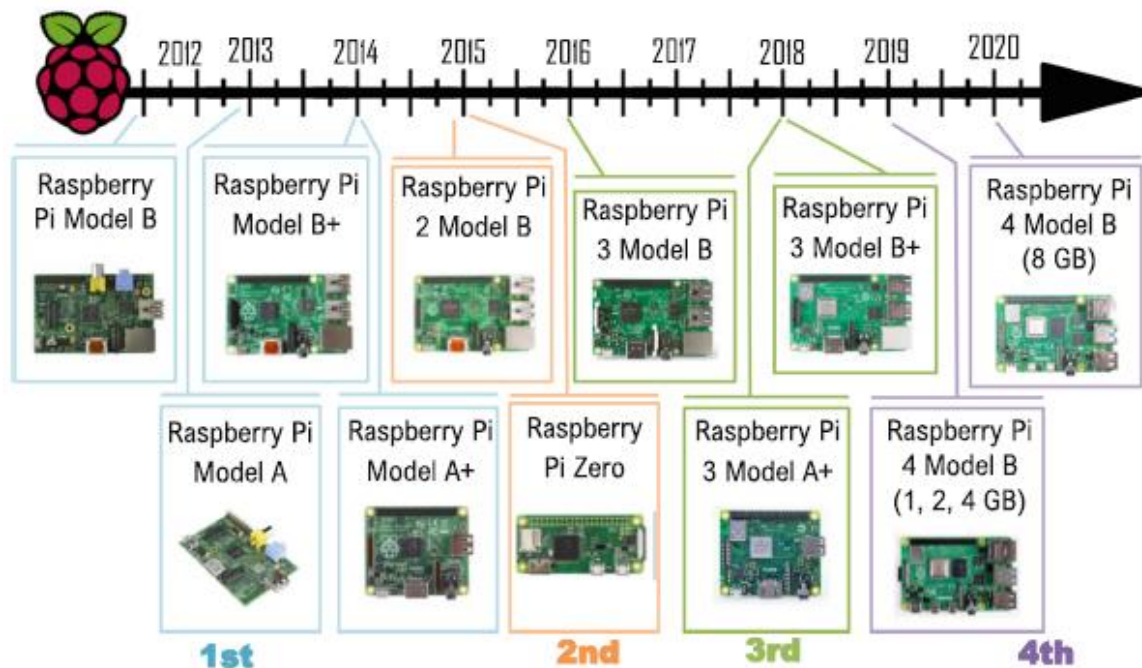
sehingga dapat dieksekusi di berbagai lingkungan dan platform perangkat keras seperti android, IOS, dan *Single Board Computer* Raspberry Pi (Hope et al., 2017). Adapun aspek lain yaitu fleksibilitas, *library* ini mendukung berbagai macam hardware dalam sebuah perangkat mesin yaitu CPU (*Central Processing Unit*), GPU (*Graphic Processing Unit*), dan TPU (*Tensor Processing Unit*) (Abadi et al., 2016). Tensorflow memiliki modul lain yang bernama Tensorflow Lite (TF Lite), modul ini dikembangkan untuk mengoptimalkan model yang telah dibangun pada Tensorflow sehingga model tersebut dapat berjalan pada perangkat *mobile* secara efisien (Shin & Kim, 2022).

2.12. OpenCV

Awal mula OpenCV dikembangkan oleh Gary Bradsky (Intel) pada tahun 1999 dan rilis pertama pada tahun 2000. Saat ini, OpenCV digunakan sebagai *library* yang mendukung banyak algoritma terkait visi komputer dan *machine learning* (Mordvintsev & Abid, 2017). Perkembangan ini membuat OpenCV mendukung berbagai macam bahasa pemrograman seperti Python, Java, C++ sehingga *library* ini mendukung berbagai jenis sistem operasi pengembangan seperti Windows, Linux, Mac OS, Android, IOS dan sebagainya (Mordvintsev & Abid, 2017). Pada penelitian ini terfokus pada penggunaan OpenCV dengan bahasa pemrograman Python yang berjalan pada sistem operasi Windows dan Linux.

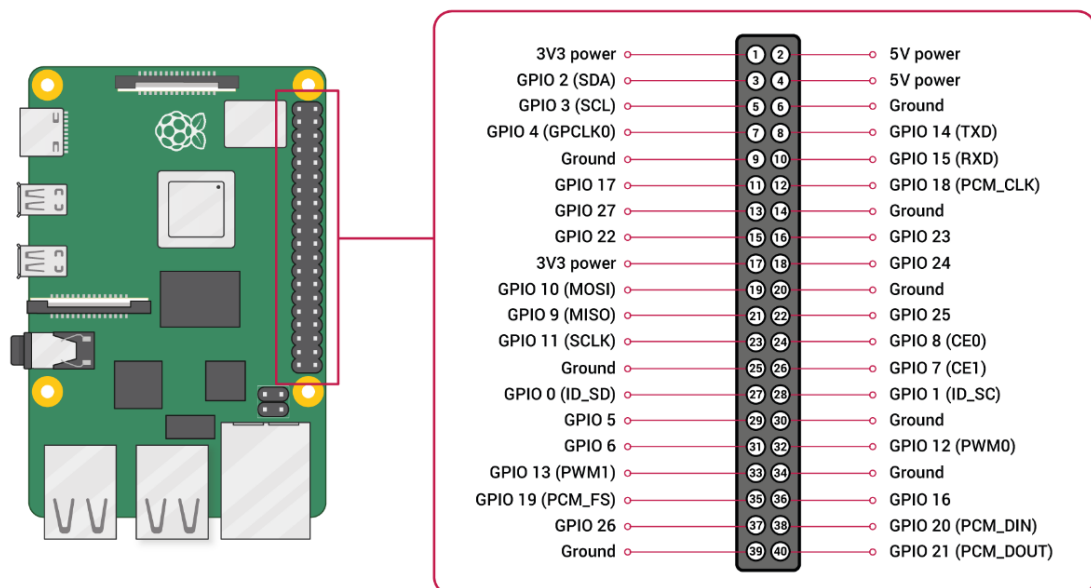
2.13. Raspberry Pi

Raspberry Pi adalah *Single Board Computer* (SBC) yang dikembangkan oleh Raspberry Pi Foundation di Inggris yang dimulai dari tahun 2012 hingga sekarang (Ben abdel ouahab et al., 2021). Secara fisik SBC ini berukuran sebesar kartu kredit yang bisa dihubungkan dengan perangkat periferal sebagai sarana pendukung seperti monitor, keyboard, mouse dan sebagainya. Pada awal kemunculan dimulai dengan versi sederhana hingga berkembang seperti saat ini dengan penambahan berbagai fitur supaya mencakup berbagai macam proyek. Perkembangan versi Raspberry Pi dapat dilihat pada Gambar 2.6.



Gambar 2.6. *Timeline* Perkembangan Raspberry Pi (Google, 2022)

Salah satu kelebihan SBC ini dibandingkan dengan komputer yang berukuran mini lainnya adalah tersedianya pin GPIO (*General Purpose Input Output*) yang dapat digunakan untuk mengontrol berbagai macam perangkat elektronik seperti sensor. Hal ini memungkinkan untuk pengembangan proyek penelitian, robotika, dan otomatisasi mesin. Susunan GPIO pada Raspberry Pi 4 dapat dilihat pada Gambar 2.7.



Gambar 2.7. Susunan GPIO Raspberry Pi 4 (Google, 2022)

Raspberry Pi mendukung sistem operasi berbasis Linux sehingga perangkat ini mendukung berbagai macam bahasa pemrograman, *library*, dan aplikasi. Selain itu, SBC ini dapat dimanfaatkan sebagai media lain seperti hiburan, server penyimpanan, dan berbagai kegiatan lainnya.

2.14. Google Coral USB Accelerator

Google Coral adalah komponen *hardware* yang berperan sebagai *Edge* TPU untuk meningkatkan akselerasi pembelajaran mesin. *Hardware* ini dikembangkan oleh perusahaan Google yang mendukung *library* Tensorflow Lite, modul dari Tensorflow yang lebih ringan, dirancang untuk *mobile* and *embedded devices* (Pinto de Aguiar et al., 2020). Google Coral terhubung dengan Raspberry Pi melalui *interface* USB 3.0/2.0, perangkat ini membantu peran CPU dalam meningkatkan kecepatan pemrosesan model (*coprocessor*). Google Coral mendukung berbagai sistem operasi seperti Windows, Linux, dan MacOS dengan memasang *library* pendukung kemudian dijalankan melalui bahasa pemrograman Python maupun C++. Perangkat Google Coral USB dapat dilihat pada Gambar 2.8.



Gambar 2.8. Google Coral USB Accelerator (Pinto de Aguiar et al., 2020)

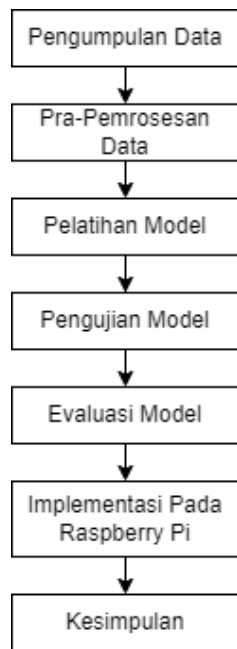
BAB III

METODOLOGI PENELITIAN

Bab ini membahas mengenai metodologi penelitian yang dilakukan dalam penyelesaian masalah deteksi rambu lalu lintas.

3.1. Gambaran Umum Penyelesaian Masalah

Gambaran umum yang dilakukan dalam penyelesaian masalah deteksi rambu lalu lintas menggunakan model SSD MobileNet-V2 pada Raspberry Pi dapat dilihat pada diagram alir yang ditunjukkan oleh Gambar 3.1.



Gambar 3.1. Gambaran Umum Penyelesaian Umum

Langkah-langkah penyelesaian masalah dalam penelitian ini terbagi menjadi beberapa tahapan. Pertama, pengumpulan data yang berupa gambar dengan anotasi rambu lalu lintas. Kedua, pra-pemrosesan data yang dilakukan dengan augmentasi data supaya data lebih seimbang. Ketiga, pembagian data yang dilakukan dengan membagi data menjadi beberapa bagian yaitu data latih, data uji, dan data validasi. Keempat, pelatihan model dengan salah satu model *deep learning* yaitu SSD MobileNet-V2. Kelima, pengujian model menggunakan data validasi sesuai dengan pembagian data. Keenam, evaluasi yang dilakukan dengan membandingkan hasil dari tiap pengujian. Ketujuh, implementasi model terbaik pada

perangkat Raspberry Pi, Kedelapan, kesimpulan hasil dari penelitian yang berisi hasil akhir dan saran pengembangan berikutnya.

3.2. Pengumpulan Data

Dataset yang digunakan pada penelitian ini yaitu *German Traffic Sign Detection Benchmark* (GTSDb) yang dapat diunduh pada *website* <https://benchmark.ini.rub.de>. *Dataset* ini berisi Citra RGB dengan ukuran 1360x800 piksel dan berjumlah 900 yang terdiri dari 1206 anotasi rambu lalu lintas (terbagi dalam beberapa kelas rambu lalu lintas). Contoh citra dari dataset dapat dilihat pada Gambar 3.2.



Gambar 3.2. Contoh Citra Rambu Lalu Lintas

Setiap citra berisi gabungan antara rambu lalu lintas dengan ukuran yang bervariasi dan dilengkapi latar belakang yang diambil di daerah Bochum (Jerman). Pengambilan citra dilakukan di berbagai tempat seperti perkotaan, pedesaan, dan jalan tol serta berbagai kondisi seperti siang, senja, dan berbagai kondisi cuaca. Secara umum rambu lalu lintas terbagi menjadi empat buah kelas yaitu yaitu *prohibitory* (larangan), *danger* (peringatan/bahaya), *mandatory* (perintah), *other* (lain-lain).

3.3. Pra-Pemrosesan Data

Pra-pemrosesan data terbagi ke dalam beberapa langkah yaitu konversi *dataset*, augmentasi data, dan pembagian data.

3.3.1. Konversi Data

Konversi data dilakukan dengan mengubah format citra pada *dataset* yang berupa PPM (*Portable Pixel Map*) menjadi PNG (*Portable Network Graphics*) dan menyimpan anotasi kelas pada sebuah berkas CSV (*Comma Separated Values*) yang disusun berdasarkan hasil konversi citra. Hasil konversi citra dan berkas anotasi disimpan dalam sebuah folder yang sama supaya mempermudah pengaksesan data. Gambaran mengenai struktur berkas CSV yang disimpan dapat dilihat pada Gambar 3.3.

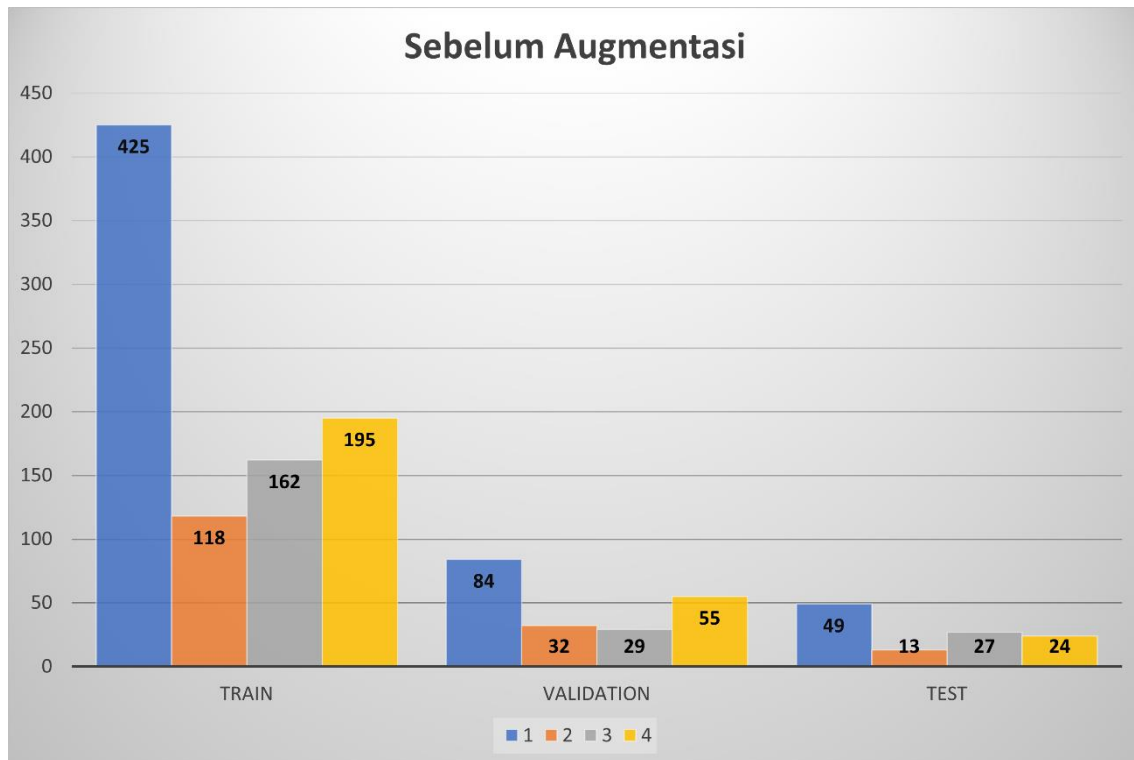
filename,width,height,xmin,ymin,xmax,ymax,classId		
00000.png,1360,800,774,411,815,446,3		

Gambar 3.3. Struktur Berkas CSV Awal

Struktur terdiri dari nama file, dimensi citra (lebar dan tinggi), lokasi objek, dan kelas objek. Setiap citra memiliki variasi dalam jumlah anotasi sehingga tiap citra memungkinkan untuk memiliki anotasi lebih dari satu. *Dataset* memiliki 4 buah kelas dasar antara lain: kelas 1 mengacu pada rambu yang bersifat *prohibitory*, kelas 2 mengacu pada rambu yang bersifat *mandatory*, kelas 3 mengacu pada rambu yang bersifat *danger*, dan kelas 4 mengacu pada rambu yang bersifat *other*.

3.3.2. Augmentasi Data

Augmentasi citra dilakukan untuk meningkatkan jumlah *dataset* dikarenakan *dataset* memiliki persebaran jumlah kelas yang tidak merata. Hal ini bertujuan untuk menyeimbangkan jumlah anotasi kelas pada *dataset* supaya pelatihan model lebih baik. Persebaran jumlah kelas pada *dataset* sebelum proses augmentasi data dapat dilihat pada Gambar 3.4.



Gambar 3.4. Jumlah Kelas Awal

Persebaran jumlah kelas sebelum proses augmentasi data adalah 900 anotasi data *train*, 200 anotasi data *validation*, dan 113 anotasi data *test*. Data *train* terdiri dari 425 anotasi kelas 1, 118 anotasi kelas 2, 162 anotasi kelas 3, dan 195 anotasi kelas 4. Data *validation* terdiri dari 84 anotasi kelas 1, 32 anotasi kelas 2, 29 anotasi kelas 3, dan 55 anotasi kelas 4. Data *test* terdiri dari 49 anotasi kelas 1, 13 anotasi kelas 2, 27 anotasi kelas 3, dan 24 anotasi kelas 4. Augmentasi data dilakukan dengan operasi pengolahan citra digital dengan beberapa metode. Adapun metode yang digunakan dalam pengolahan citra digital adalah sebagai berikut.

1) *Translation*

Operasi *translation* atau translasi digunakan untuk melakukan pergeseran citra yang dilakukan dengan pergeseran. Posisi piksel pada citra akan bergeser sejauh nilai pergeseran sesuai sumbu yang diinginkan baik horizontal maupun vertikal. Persamaan operasi *translation* menggunakan persamaan 2.1, dapat dituliskan pada persamaan 3.1.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} tx \\ ty \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix} \dots\dots\dots (3.1)$$

Contoh perhitungan operasi *translation* pada sebuah piksel:

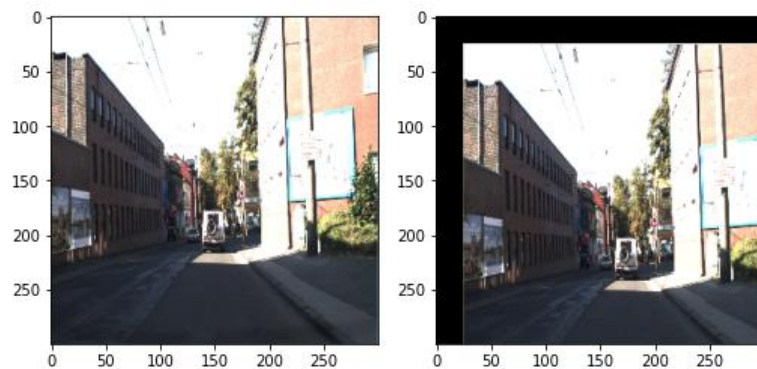
Matriks M menyatakan titik sebuah piksel pada citra, x mengacu pada sumbu horizontal dan y mengacu pada sumbu vertikal. Matriks T menyatakan nilai pergeseran untuk masing-masing sumbu.

$$M = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \text{ dan } T = \begin{bmatrix} tx \\ ty \end{bmatrix} = \begin{bmatrix} 25 \\ 25 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 25 \\ 25 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 25 \\ 25 \end{bmatrix}$$

Proses di atas merupakan perhitungan sederhana dari sebuah piksel pada citra. Penerapan operasi ini pada citra RGB dilakukan dengan perhitungan tiap *channel* dan sebanyak ukuran lebar dan tinggi citra masukan. Gambaran operasi *translasi* dapat dilihat pada Gambar 3.5.



Gambar 3.5. Proses *Translasi* Citra

Bagian kiri menunjukkan citra sebelum proses *translasi* dan bagian kanan menunjukkan citra sesudah proses *translasi*. Posisi yang tidak ditempati oleh hasil *translation* piksel diisi dengan nilai konstanta seperti 0 (nol) dan bagian yang tidak masuk ke dalam jendela utama akan terpotong.

2) *Rotation*

Operasi *rotation* atau rotasi digunakan untuk melakukan perputaran citra yang dilakukan dengan perputaran piksel. Posisi piksel pada citra akan berputar sejauh derajat / sudut perputaran. Persamaan operasi *rotation* menggunakan persamaan 2.4, dapat dilihat pada persamaan 3.2.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x - x_p \\ y - y_p \end{bmatrix} + \begin{bmatrix} x_p \\ y_p \end{bmatrix} \dots\dots\dots (3.2)$$

Contoh perhitungan operasi *rotation* pada sebuah piksel:

Matriks M menyatakan titik sebuah piksel pada citra, x mengacu pada sumbu horizontal dan y mengacu pada sumbu vertikal. C menyatakan matriks titik pusat sebuah piksel dan θ menyatakan besaran sudut perputaran.

$$M = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}; \text{berpusat pada titik } C = \begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} 150 \\ 150 \end{bmatrix}; \theta = 45$$

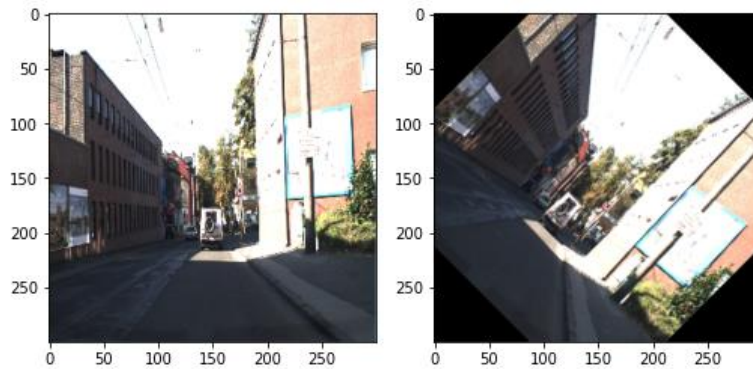
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos 45 & -\sin 45 \\ \sin 45 & \cos 45 \end{bmatrix} \begin{bmatrix} x - x_p \\ y - y_p \end{bmatrix} + \begin{bmatrix} x_p \\ y_p \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\sqrt{2} & -\frac{1}{2}\sqrt{2} \\ \frac{1}{2}\sqrt{2} & \frac{1}{2}\sqrt{2} \end{bmatrix} \begin{bmatrix} 0 - 150 \\ 0 - 150 \end{bmatrix} + \begin{bmatrix} 150 \\ 150 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\sqrt{2}(-150) - \frac{1}{2}\sqrt{2}(-150) + 150 \\ \frac{1}{2}\sqrt{2}(-150) + \frac{1}{2}\sqrt{2}(-150) + 150 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 150 \\ 150 - 150\sqrt{2} \end{bmatrix}$$

Proses di atas merupakan perhitungan sederhana dari sebuah piksel pada citra. Penerapan operasi ini pada citra RGB dilakukan dengan perhitungan tiap *channel* dan sebanyak ukuran lebar dan tinggi citra masukan. Gambaran operasi *rotation* dapat dilihat pada Gambar 3.6.



Gambar 3.6. Proses *Rotation* Citra

Bagian kiri menunjukkan citra sebelum proses *rotation* dan bagian kanan menunjukkan citra sesudah proses *rotation*. Posisi yang tidak ditempati oleh hasil *rotation* piksel diisi dengan nilai konstanta seperti 0 (nol) dan bagian yang tidak masuk ke dalam jendela utama akan terpotong.

3) *Scaling*

Operasi *scaling* atau penskalaan digunakan untuk melakukan perubahan ukuran citra menjadi besar atau kecil yang bergantung dengan faktor penskalaan. Posisi piksel pada citra

akan berputar sejauh derajat / sudut perputaran. Contoh perhitungan operasi *scaling* pada sebuah *channel* citra menggunakan persamaan 2.7, dapat dituliskan pada persamaan 3.3.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x - x_p \\ y - y_p \end{bmatrix} + \begin{bmatrix} x_p \\ y_p \end{bmatrix} \dots\dots\dots(3.3)$$

Contoh perhitungan operasi *scaling* pada sebuah piksel:

Matriks M menyatakan titik sebuah piksel pada citra, x mengacu pada sumbu horizontal dan y mengacu pada sumbu vertikal. C menyatakan matriks titik pusat sebuah piksel dan S menyatakan faktor penskalaan masing-masing sumbu.

$$M = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}; \text{berpusat pada titik } C = \begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} 150 \\ 150 \end{bmatrix}; S = \begin{bmatrix} S_x \\ S_y \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

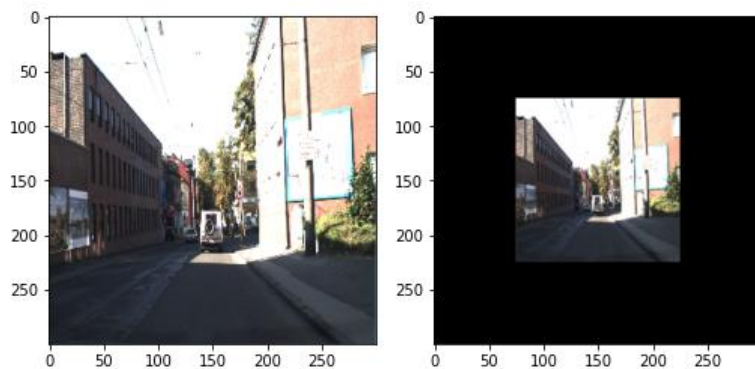
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} 0 - 150 \\ 0 - 150 \end{bmatrix} + \begin{bmatrix} 150 \\ 150 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 0.5 (-150) \\ 0.5 (-150) \end{bmatrix} + \begin{bmatrix} 150 \\ 150 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -75 \\ -75 \end{bmatrix} + \begin{bmatrix} 150 \\ 150 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 75 \\ 75 \end{bmatrix}$$

Proses di atas merupakan perhitungan sederhana dari sebuah piksel pada citra. Penerapan operasi ini pada citra RGB dilakukan dengan perhitungan tiap *channel* dan sebanyak ukuran lebar dan tinggi citra masukan. Gambaran operasi *scaling* dapat dilihat pada Gambar 3.7.



Gambar 3.7. Proses *Scaling* Citra

Bagian kiri menunjukkan citra sebelum proses *scaling* dan bagian kanan menunjukkan citra sesudah proses *scaling*. Posisi yang tidak ditempati oleh hasil *scaling* piksel diisi dengan nilai konstanta seperti 0 (nol) dan bagian yang tidak masuk ke dalam jendela utama akan terpotong.

4) *Reflection / Flipping*

Operasi *reflection* (pencerminan) atau *flipping* (pembalikan) digunakan untuk mengubah posisi citra menjadi terbalik secara horizontal maupun vertikal. Contoh perhitungan operasi *reflection / flipping* pada sebuah *channel* citra menggunakan persamaan 2.13, dapat dituliskan pada persamaan 3.4.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x - x_p \\ y - y_p \end{bmatrix} + \begin{bmatrix} x_p \\ y_p \end{bmatrix} \dots\dots\dots (3.4)$$

Contoh perhitungan operasi *reflection / flipping* pada sebuah piksel terhadap sumbu Y:

Matriks M menyatakan titik sebuah piksel pada citra, x mengacu pada sumbu horizontal dan y mengacu pada sumbu vertikal. C menyatakan matriks titik pusat sebuah piksel.

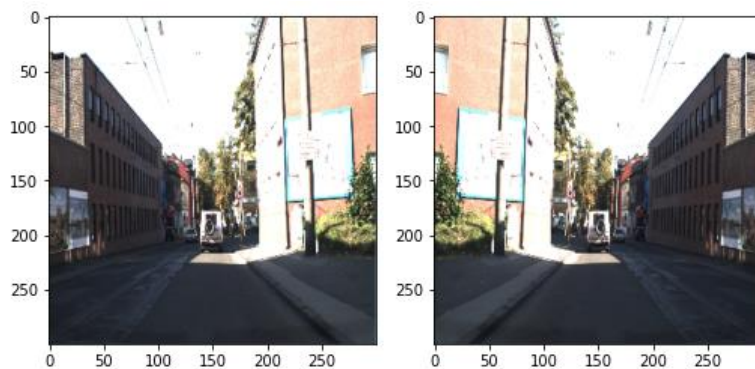
$$M = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}; \text{berpusat pada titik } C = \begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} 150 \\ 150 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 - 150 \\ 0 - 150 \end{bmatrix} + \begin{bmatrix} 150 \\ 150 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 150 \\ -150 \end{bmatrix} + \begin{bmatrix} 150 \\ 150 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 300 \\ 0 \end{bmatrix}$$

Proses di atas merupakan perhitungan sederhana dari sebuah piksel pada citra. Penerapan operasi ini pada citra RGB dilakukan dengan perhitungan tiap *channel* dan sebanyak ukuran lebar dan tinggi citra masukan. Gambaran operasi *reflection / flipping* dapat dilihat pada Gambar 3.8.



Gambar 3.8. Proses *Reflection / Flipping* Citra

Bagian kiri menunjukkan citra sebelum proses *reflection / flipping* dan bagian kanan menunjukkan citra sesudah proses *reflection / flipping*.

5) *Shearing*

Operasi *shearing* digunakan untuk membuat efek distorsi pada posisi citra dengan cara menarik ke salah satu sisi baik secara horizontal maupun vertikal. Contoh perhitungan operasi *shearing* pada sebuah *channel* citra menggunakan persamaan 2.16, dapat dituliskan pada persamaan 3.5.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & Shx \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \dots\dots\dots (3.5)$$

Contoh perhitungan operasi *shearing* pada sebuah piksel terhadap sumbu X:

Matriks M menyatakan titik sebuah piksel pada citra, x mengacu pada sumbu horizontal dan y mengacu pada sumbu vertikal. C menyatakan matriks titik pusat sebuah piksel dan Shx menyatakan faktor *shearing* terhadap sumbu X.

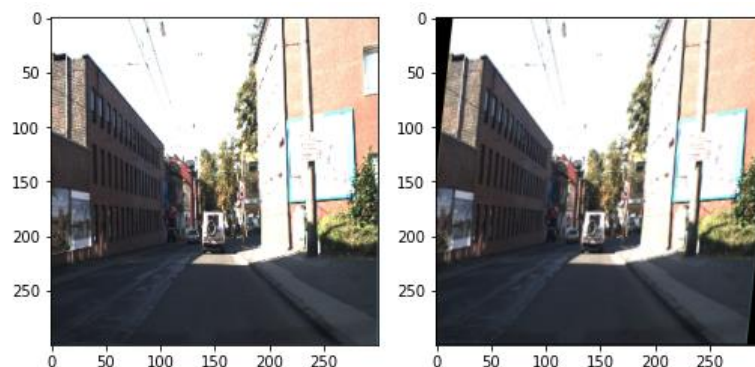
$$M = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 300 \end{bmatrix}; Shx = 0.1$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 300 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 0 + 30 \\ 300 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 30 \\ 300 \end{bmatrix}$$

Proses di atas merupakan perhitungan sederhana dari sebuah piksel pada citra. Penerapan operasi ini pada citra RGB dilakukan dengan perhitungan tiap *channel* dan sebanyak ukuran lebar dan tinggi citra masukan. Gambaran operasi *shearing* dapat dilihat pada Gambar 3.9.



Gambar 3.9. Proses *Shearing* Citra

Bagian kiri menunjukkan citra sebelum proses *shearing* dan bagian kanan menunjukkan citra sesudah proses *shearing*. Posisi yang tidak ditempati oleh hasil *shearing* piksel diisi dengan nilai konstanta seperti 0 (nol) dan bagian yang tidak masuk ke dalam jendela utama akan terpotong.

6) Arithmetic

Operasi *arithmetic* atau aritmetika digunakan untuk mengubah nilai tiap piksel pada citra dengan operasi aritmetika seperti penjumlahan, pengurangan dan perkalian. Perubahan nilai juga ditambahkan nilai ambang batas supaya nilai tetap berada diantara 0 sampai 255. Ketika nilai kurang dari 0 maka dibulatkan menjadi 0 dan nilai lebih dari 255 maka dibulatkan menjadi 255. Contoh perhitungan operasi *arithmetic* pada sebuah *channel* citra menggunakan persamaan 2.22 sampai 2.24, dapat dituliskan pada persamaan 3.6.

a. Penjumlahan

$$y = x + C \dots\dots\dots(3.6)$$

Contoh perhitungan operasi penjumlahan:

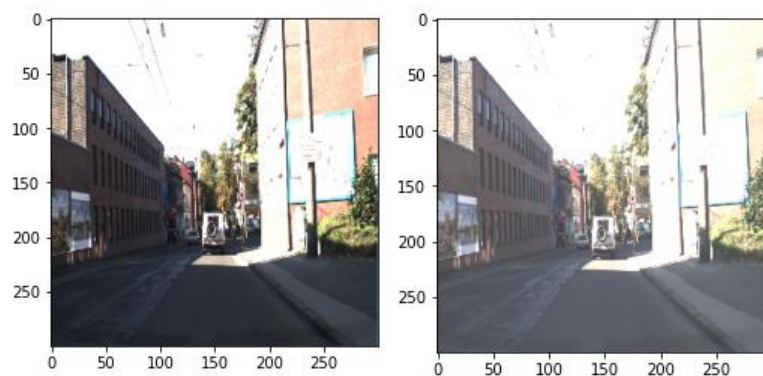
X menyatakan nilai suatu piksel dan C menyatakan nilai konstanta yang digunakan untuk operasi penjumlahan nilai piksel.

$$x = 25; C = 50$$

$$y = 25 + 50$$

$$y = 75$$

Proses di atas merupakan perhitungan sederhana dari sebuah piksel pada citra. Penerapan operasi ini pada citra RGB dilakukan dengan perhitungan tiap *channel* dan sebanyak ukuran lebar dan tinggi citra masukan. Gambaran operasi penjumlahan dapat dilihat pada Gambar 3.10.



Gambar 3.10. Proses Penjumlahan Nilai Piksel

Bagian kiri menunjukkan citra sebelum proses penjumlahan dan bagian kanan menunjukkan citra sesudah proses penjumlahan. Citra yang mengalami proses penjumlahan akan menjadi lebih terang karena adanya penambahan nilai tiap pikselnya.

b. Pengurangan

$$y = x - C \dots\dots\dots(3.7)$$

Contoh perhitungan operasi pengurangan:

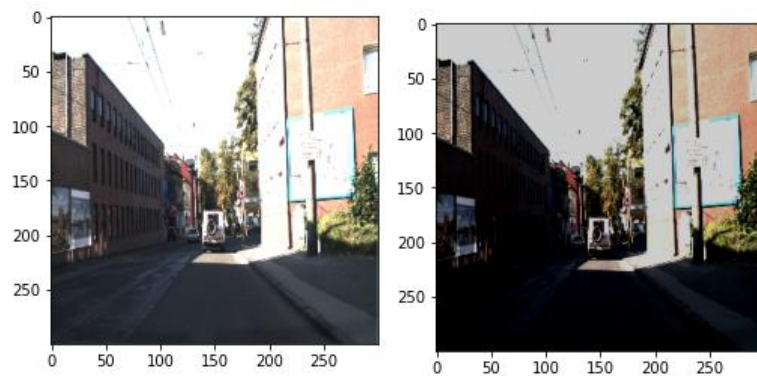
X menyatakan nilai suatu piksel dan C menyatakan nilai konstanta yang digunakan untuk operasi pengurangan nilai piksel.

$$x = 100; C = 50$$

$$y = 100 - 50$$

$$y = 50$$

Proses di atas merupakan perhitungan sederhana dari sebuah piksel pada citra. Penerapan operasi ini pada citra RGB dilakukan dengan perhitungan tiap *channel* dan sebanyak ukuran lebar dan tinggi citra masukan. Gambaran operasi pengurangan dapat dilihat pada Gambar 3.11.



Gambar 3.11. Proses Pengurangan Nilai Piksel

Bagian kiri menunjukkan citra sebelum proses pengurangan dan bagian kanan menunjukkan citra sesudah proses pengurangan. Citra yang mengalami proses pengurangan akan menjadi lebih gelap karena adanya pengurangan nilai tiap pikselnya.

c. Perkalian

$$y = Cx \dots\dots\dots(3.8)$$

Contoh perhitungan operasi perkalian:

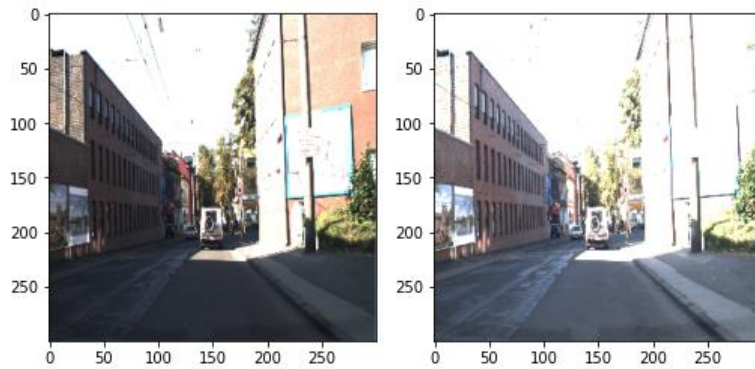
X menyatakan nilai suatu piksel dan C menyatakan nilai konstanta yang digunakan untuk operasi perkalian nilai piksel.

$$x = 100; C = 2$$

$$y = 2 (100)$$

$$y = 200$$

Proses di atas merupakan perhitungan sederhana dari sebuah piksel pada citra. Penerapan operasi ini pada citra RGB dilakukan dengan perhitungan tiap *channel* dan sebanyak ukuran lebar dan tinggi citra masukan. Gambaran operasi perkalian dapat dilihat pada Gambar 3.12.



Gambar 3.12. Proses Perkalian Nilai Pixel

Bagian kiri menunjukkan citra sebelum proses perkalian dan bagian kanan menunjukkan citra sesudah proses perkalian. Citra yang mengalami proses perkalian akan menjadi sangat terang atau sangat gelap sesuai dengan nilai konstanta.

7) Filtering

Operasi *filtering* atau penapisan digunakan untuk mengubah nilai tiap piksel pada citra sebuah *kernel* yang berjalan mengikuti ukuran citra. Operasi penapisan yang digunakan yaitu *filter blur* yang bertujuan untuk memperhalus gambar. Perubahan nilai juga ditambahkan nilai ambang batas supaya nilai tetap berada diantara 0 sampai 255. Ketika nilai kurang dari 0 maka dibulatkan menjadi 0 dan nilai lebih dari 255 maka dibulatkan menjadi 255. Contoh perhitungan operasi *filtering* pada sebuah *channel* citra menggunakan persamaan 2.25 sampai 2.26, dapat dituliskan pada persamaan 3.9 dan 3.10.

a. Average Filter (Rata-rata)

$$f(x, y)' = \frac{1}{9} \sum_{p=-1}^1 \sum_{q=-1}^1 f(x + p, y + q) \dots\dots\dots(3.9)$$

Contoh perhitungan operasi *average filter*:

Matriks M menyatakan potongan matriks sebuah citra dan nilai yang tersusun merupakan nilai piksel. K menyatakan ukuran kernel yaitu 3x3 yang digunakan untuk operasi penapisan. Nilai yang terlibat dalam perhitungan *average filter* adalah nilai yang masuk dalam pergerakan kernel. Hasil dari perhitungan ini menggantikan nilai lama yang berada di bagian tengah kernel.

$$M = \begin{bmatrix} 90 & 180 & 99 \\ 81 & 54 & 18 \\ 9 & 72 & 171 \end{bmatrix}; \quad K = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

$$f(x, y)' = \frac{1}{9} (90) + \frac{1}{9} (180) + \frac{1}{9} (99) + \frac{1}{9} (81) + \frac{1}{9} (54) + \frac{1}{9} (18) + \frac{1}{9} (9)$$

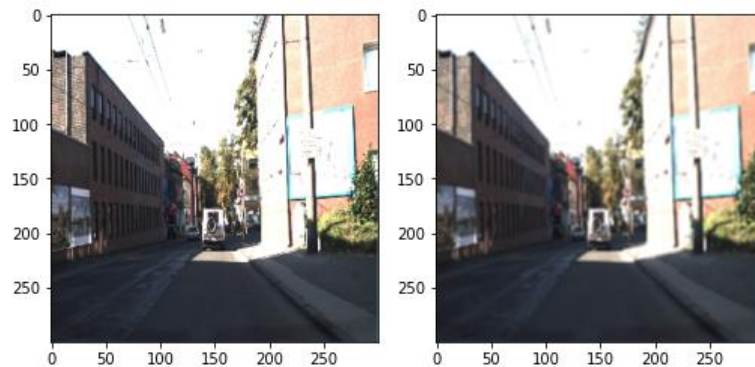
$$+\frac{1}{9} (72) + \frac{1}{9} (171)$$

$$f(x,y)' = 10 + 20 + 11 + 9 + 6 + 2 + 1 + 8 + 19$$

$$f(x,y)' = 86$$

$$M = \begin{bmatrix} 90 & 180 & 99 \\ 81 & 54 & 18 \\ 9 & 72 & 171 \end{bmatrix} \rightarrow \begin{bmatrix} 90 & 180 & 99 \\ 81 & 86 & 18 \\ 9 & 72 & 171 \end{bmatrix}$$

Proses di atas merupakan perhitungan sederhana dari sebuah potongan citra. Penerapan operasi ini pada citra RGB dilakukan dengan perhitungan tiap *channel* dan sebanyak ukuran lebar dan tinggi citra masukan. Gambaran operasi *average filter* dapat dilihat pada Gambar 3.13.



Gambar 3.13. Proses *Average Filter*

Bagian kiri menunjukkan citra sebelum proses *average filter* dan bagian kanan menunjukkan citra sesudah proses *average filter*.

b. *Median Filter* (Nilai Tengah)

$$f(x,y)' = \text{median}(f(x-1,y-1), f(x-1,y), f(x-1,y+1), f(x,y-1), f(x,y), f(x,y+1), f(x+1,y-1), f(x+1,y), f(x+1,y+1)) \dots\dots\dots(3.10)$$

Contoh perhitungan operasi *median filter*:

Matriks M menyatakan potongan matriks sebuah citra dan nilai yang tersusun merupakan nilai piksel. K menyatakan ukuran kernel yaitu 3x3 yang digunakan untuk operasi penapisan. Nilai yang terlibat dalam perhitungan *median filter* adalah nilai yang masuk dalam pergerakan kernel. Operasi ini menjalankan fungsi median yang disusun dengan urutan nilai terkecil hingga terbesar Hasil dari perhitungan menggantikan nilai lama yang berada di bagian tengah kernel.

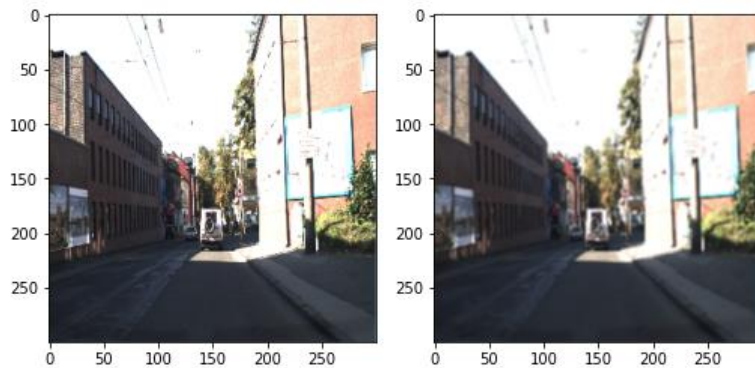
$$M = \begin{bmatrix} 90 & 180 & 99 \\ 81 & 54 & 18 \\ 9 & 72 & 171 \end{bmatrix}; K = \begin{bmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{bmatrix}$$

$$f(x,y)' = \text{median}(9, 18, 54, 72, 81, 90, 99, 171, 180)$$

$$f(x,y)' = 81$$

$$M = \begin{bmatrix} 90 & 180 & 99 \\ 81 & 54 & 18 \\ 9 & 72 & 171 \end{bmatrix} \rightarrow \begin{bmatrix} 90 & 180 & 99 \\ 81 & 81 & 18 \\ 9 & 72 & 171 \end{bmatrix}$$

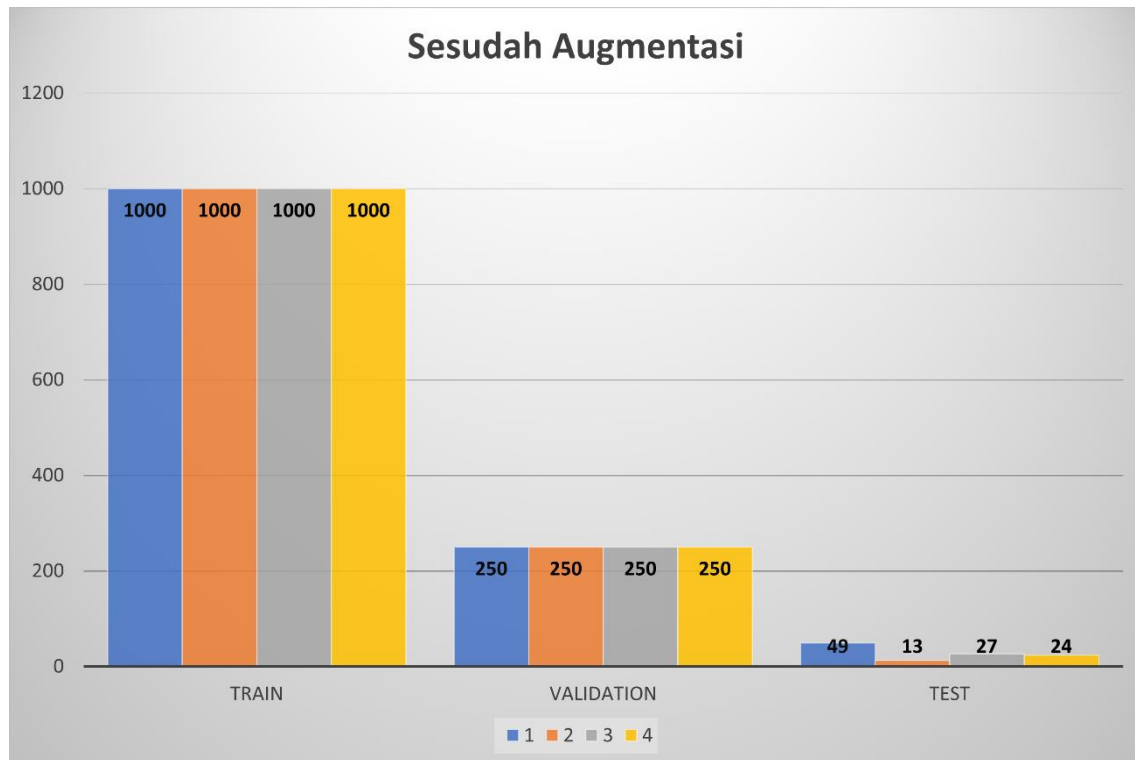
Proses di atas merupakan perhitungan sederhana dari sebuah potongan citra. Penerapan operasi ini pada citra RGB dilakukan dengan perhitungan tiap *channel* dan sebanyak ukuran lebar dan tinggi citra masukan. Gambaran operasi *median filter* dapat dilihat pada Gambar 3.14.



Gambar 3.14. Proses *Median Filter*

Bagian kiri menunjukkan citra sebelum proses *median filter* dan bagian kanan menunjukkan citra sesudah proses *median filter*.

Persebaran jumlah kelas setelah proses augmentasi data adalah 4000 anotasi data *train*, 1000 anotasi data *validation*, dan 113 anotasi data *test*. Data *train* terdiri dari 1000 anotasi kelas 1, 1000 anotasi kelas 2, 1000 anotasi kelas 3, dan 1000 anotasi kelas 4. Data *validation* terdiri dari 250 anotasi kelas 1, 250 anotasi kelas 2, 250 anotasi kelas 3, dan 250 anotasi kelas 4. Data *test* terdiri dari 49 anotasi kelas 1, 13 anotasi kelas 2, 27 anotasi kelas 3, dan 24 anotasi kelas 4. Pada data *test* tidak dilakukan proses augmentasi data dikarenakan data ini digunakan untuk mengukur kinerja model. Persebaran jumlah kelas pada *dataset* sesudah proses augmentasi data dapat dilihat pada Gambar 3.15.



Gambar 3.15. Jumlah Kelas Akhir

3.3.3. Pembagian Data

Pembagian data setelah proses augmentasi data dilakukan dengan membagi data menjadi data latih, data validasi, dan data uji. Data latih dan data validasi digunakan untuk pelatihan model, sedangkan data uji digunakan untuk pengujian model. Data latih berjumlah 4000 anotasi dengan pembagian 1000 anotasi tiap kelas, data validasi berjumlah 1000 anotasi dengan pembagian 250 anotasi tiap kelas, dan data uji berjumlah 113 anotasi yang terbagi menjadi beberapa kelas yang telah dijelaskan pada Gambar 3.15.

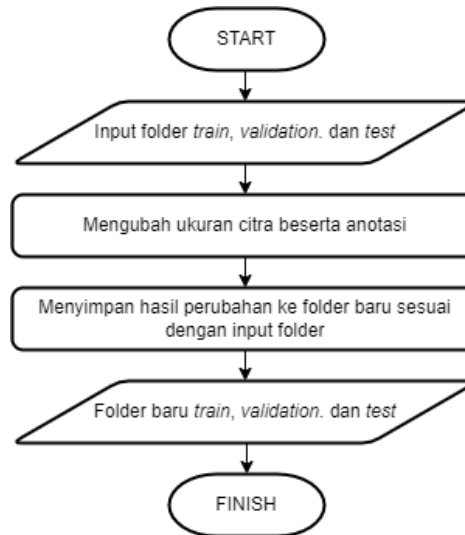
3.4. Pelatihan Model

Pelatihan model untuk deteksi rambu lalu lintas terdiri dari beberapa tahapan utama yaitu proses persiapan data dan pelatihan model menggunakan model SSD (*Single Shot Detector*) MobileNet-V2.

3.4.1. Persiapan Data

Persiapan data diawali dengan mengubah ukuran citra yang sudah dilakukan pada pra-pemrosesan data menjadi ukuran 300x300x3 beserta anotasi. Hal ini bertujuan untuk menyesuaikan dengan ukuran input yang digunakan oleh model. Selain itu, perubahan ukuran diawal dapat mengurangi waktu komputasi dibandingkan dengan ukuran citra yang

lebih besar. Perubahan ukuran citra dilakukan tiap masing-masing folder yaitu *train*, *validation*, dan *test*. Proses ini juga diikuti dengan perubahan anotasi yang ada pada citra menyesuaikan dengan ukuran baru. Hasil perubahan citra beserta anotasi disimpan ke dalam folder baru yang menyesuaikan folder input. Diagram alir dalam persiapan data dapat dilihat pada Gambar 3.16.



Gambar 3.16. Diagram Alir Persiapan Data

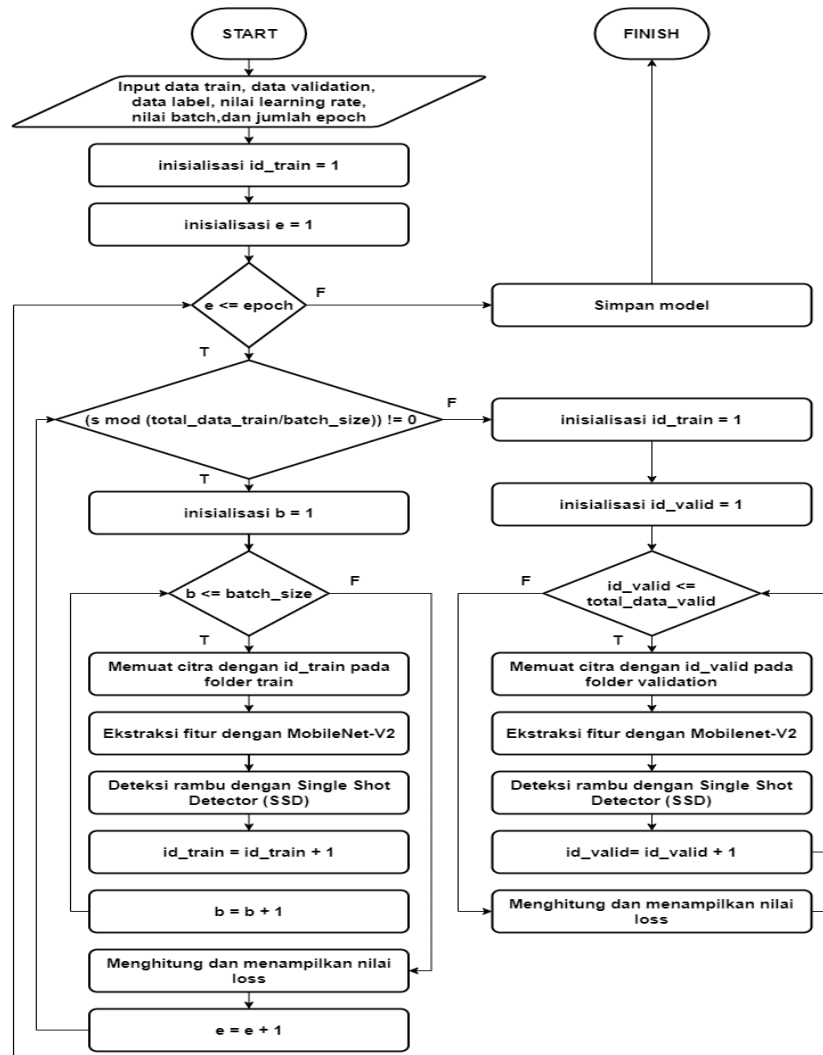
Hasil perubahan ukuran citra beserta anotasi disimpan ke dalam berkas CSV untuk mempermudah penggunaan pada proses berikutnya. Struktur CSV terdiri dari nama file, dimensi citra (lebar dan tinggi), lokasi objek, dan id kelas. Gambaran mengenai struktur berkas CSV yang disimpan dapat dilihat pada Gambar 3.17.

filename	class_text	width	height	xmin	ymin	xmax	ymax	classId
00306_118.png	Danger	300	300	82	176	89	188	3

Gambar 3.17. Struktur Berkas CSV Akhir

3.4.2. Skema Pelatihan Model

Pelatihan model dilakukan dengan bantuan *library* yang dikembangkan oleh Google pada tahun 2015 yaitu Tensorflow. *Library* ini dapat berjalan dengan bahasa pemrograman Python dan dapat bekerja dari berbagai sistem operasi. Tensorflow memiliki kemampuan untuk digunakan secara *cloud service* melalui Google Colaboratory dengan jupyter notebook yang dapat digunakan secara *online* serta mendukung perangkat komputasi yang disediakan seperti *Central Processing Unit* (CPU), *Graphics Processing Unit* (GPU), atau *Tensor Processing Unit* (TPU). Skema pelatihan model dapat dilihat pada diagram alir yang ditunjukkan oleh Gambar Gambar 3.17.



Gambar 3.18. Diagram Alir Pelatihan Model

Pelatihan model diawali dengan memberi masukan data *train*, data *validation*, data label, nilai *learning rate*, nilai *batch* dan jumlah *epoch*. Pelatihan dimulai dengan masukan dari data *train* (citra dan anotasi) kemudian dilakukan proses ekstraksi fitur dengan MobileNet-V2 dan deteksi menggunakan SSD. Proses ini berjalan sesuai nilai *batch* untuk satu *epoch* yang dilakukan. *Batch* memiliki arti jumlah data yang diproses sekaligus dalam satu *epoch* ketika nilai *batch* diasumsikan dengan 10 maka pemrosesan 10 gambar sekaligus dinyatakan dengan 1 *epoch*. Semakin besar nilai *batch* akan berdampak pada waktu komputasi dalam 1 *epoch* dan setiap *epoch* yang berjalan dilakukan perhitungan nilai *loss*. Setelah tercapai jumlah *epoch* sesuai banyak data *train* (jumlah *epoch* dibagi dengan nilai *batch*) dilakukan proses evaluasi menggunakan data *validation* dengan menghitung nilai *loss* sebanyak data yang diberikan. Setelah jumlah *epoch* yang diinginkan terpenuhi maka proses selesai dan model akan disimpan untuk tahapan berikutnya.

3.4.3. Pelatihan Model SSD MobileNet-V2

SSD MobileNet-V2 tersusun atas MobileNet-V2 yang digunakan untuk ekstraksi fitur dan SSD yang digunakan sebagai detektor objek. Model ini memiliki ukuran citra masukan sebesar 300x300x3. MobileNet-V2 memiliki 19 lapisan yang terdiri dari 1 buah lapisan konvolusi 3x3, 17 buah lapisan *bottleneck*, dan 1 buah lapisan konvolusi 1x1. SSD memiliki 6 buah *feature maps* yang diambil dari 2 lapisan MobileNet-V2 pada lapisan *bottleneck* k-14 dan konvolusi 1x1 bagian akhir MobileNet-V2. Sisa dari *feature maps* diambil dari 4 buah *layer* tambahan yang ditambahkan pada bagian akhir MobileNet-V2. Gambaran dan informasi mengenai model SSD MobileNet-V2 dapat dilihat pada Gambar 2.5 dan Tabel 2.3. Hasil akhir pada model berupa prediksi *bounding box* yang menunjukkan lokasi objek dan kelas suatu objek. Contoh perhitungan pada model SSD MobileNet-V2 adalah sebagai berikut.

- 1) Masukkan citra RGB yang memiliki ukuran 300x300x3. Berikut ini adalah gambaran dalam bentuk matriks masing-masing *channel*:

$$R = \begin{bmatrix} 165 & 160 & 163 & 162 & 165 & 161 & \dots \\ 169 & 168 & 167 & 169 & 161 & 164 & \dots \\ 165 & 170 & 169 & 164 & 162 & 163 & \dots \\ 166 & 161 & 163 & 164 & 160 & 167 & \dots \\ 161 & 169 & 162 & 165 & 168 & 169 & \dots \\ 160 & 161 & 164 & 165 & 166 & 160 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$G = \begin{bmatrix} 150 & 151 & 155 & 160 & 153 & 152 & \dots \\ 160 & 152 & 150 & 151 & 155 & 150 & \dots \\ 153 & 154 & 155 & 159 & 157 & 158 & \dots \\ 157 & 155 & 159 & 154 & 151 & 152 & \dots \\ 151 & 158 & 157 & 153 & 155 & 153 & \dots \\ 150 & 152 & 155 & 159 & 150 & 160 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$B = \begin{bmatrix} 100 & 110 & 101 & 102 & 100 & 107 & \dots \\ 106 & 107 & 104 & 100 & 110 & 106 & \dots \\ 107 & 104 & 103 & 104 & 102 & 108 & \dots \\ 104 & 107 & 106 & 105 & 101 & 100 & \dots \\ 110 & 101 & 104 & 107 & 100 & 104 & \dots \\ 100 & 104 & 105 & 109 & 110 & 100 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

- 2) Operasi konvolusi menggunakan kernel 3x3x3 sebanyak N kali dengan nilai *stride* 2 sehingga masukan citra yang awalnya berukuran 1xWxHx3 berubah menjadi 1xW/2xH/2xN. Proses konvolusi dilakukan tiap *channel* masukan terhadap kedalaman masing-masing kernel kemudian hasil tiap *channel* akan dijumlahkan, proses ini

berulang sebanyak N kali. Operasi ini melibatkan operasi *zero-padding* sebelum proses konvolusi dengan menambahkan nilai 0 pada bagian tepi citra. Setelah operasi konvolusi selesai dilanjutkan dengan operasi tambahan yaitu *batch normalization* dan ReLU6. Berikut ini adalah tahapan pada lapisan pertama MobileNet-V2.

a) *Zero-padding*

Contoh *zero-padding* terhadap citra masukan:

$$R = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 165 & 160 & 163 & 162 & 165 & 161 & \dots \\ 0 & 169 & 168 & 167 & 169 & 161 & 164 & \dots \\ 0 & 165 & 170 & 169 & 164 & 162 & 163 & \dots \\ 0 & 166 & 161 & 163 & 164 & 160 & 167 & \dots \\ 0 & 161 & 169 & 162 & 165 & 168 & 169 & \dots \\ 0 & 160 & 161 & 164 & 165 & 166 & 160 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$G = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 150 & 151 & 155 & 160 & 153 & 152 & \dots \\ 0 & 160 & 152 & 150 & 151 & 155 & 150 & \dots \\ 0 & 153 & 154 & 155 & 159 & 157 & 158 & \dots \\ 0 & 157 & 155 & 159 & 154 & 151 & 152 & \dots \\ 0 & 151 & 158 & 157 & 153 & 155 & 153 & \dots \\ 0 & 150 & 152 & 155 & 159 & 150 & 160 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 100 & 110 & 101 & 102 & 100 & 107 & \dots \\ 0 & 106 & 107 & 104 & 100 & 110 & 106 & \dots \\ 0 & 107 & 104 & 103 & 104 & 102 & 108 & \dots \\ 0 & 104 & 107 & 106 & 105 & 101 & 100 & \dots \\ 0 & 110 & 101 & 104 & 107 & 100 & 104 & \dots \\ 0 & 100 & 104 & 105 & 109 & 110 & 100 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

b) Inisialisasi kernel

Contoh inisialisasi kernel 3x3x3:

$$K_1 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}; K_2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}; K_3 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

c) Operasi konvolusi

Operasi konvolusi dilakukan dengan perhitungan tiap *channel* terhadap satu buah kernel kemudian masing-masing elemen hasil konvolusi dijumlahkan dan dilakukan berulang sebanyak N kali dengan nilai *stride* 2. Keluaran dari operasi konvolusi ini

akan mengurangi ukuran dimensi menjadi setengah ukuran masukan dan menambah jumlah *channel* sebanyak N.

Operasi konvolusi matriks R dengan kernel pertama dengan *zero-padding* dan nilai *stride* 2:

$$R' = R * K_1$$

$$R' = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 165 & 160 & 163 & 162 & 165 & 161 & \dots \\ 0 & 169 & 168 & 167 & 169 & 161 & 164 & \dots \\ 0 & 165 & 170 & 169 & 164 & 162 & 163 & \dots \\ 0 & 166 & 161 & 163 & 164 & 160 & 167 & \dots \\ 0 & 161 & 169 & 162 & 165 & 168 & 169 & \dots \\ 0 & 160 & 161 & 164 & 165 & 166 & 160 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} * \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Contoh perhitungan operasi konvolusi untuk satu elemen matriks menggunakan persamaan 2.1, dapat dituliskan pada persamaan 3.11.

$$R'(0,0) = (R * K_1) = \sum_{i=-1}^1 \sum_{j=-1}^1 R(m,n)K_1(i-m,j-n) \dots \dots \dots (3.11)$$

$$R'(0,0) = 0 \times 1 + 0 \times 0 + 0 \times 1 + 0 \times 0 + 165 \times 1 + 160 \times 0 + 0 \times 1 \\ + 169 \times 0 + 168 \times 1$$

$$R'(0,0) = 333$$

Operasi konvolusi matriks G dengan kernel kedua dengan *zero-padding* dan nilai *stride* 2:

$$G' = G * K_2$$

$$G' = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 150 & 151 & 155 & 160 & 153 & 152 & \dots \\ 0 & 160 & 152 & 150 & 151 & 155 & 150 & \dots \\ 0 & 153 & 154 & 155 & 159 & 157 & 158 & \dots \\ 0 & 157 & 155 & 159 & 154 & 151 & 152 & \dots \\ 0 & 151 & 158 & 157 & 153 & 155 & 153 & \dots \\ 0 & 150 & 152 & 155 & 159 & 150 & 160 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Contoh perhitungan operasi konvolusi untuk satu elemen matriks menggunakan persamaan 2.1, dapat dituliskan pada persamaan di bawah ini.

$$G'(0,0) = (G * K_2) = \sum_{i=-1}^1 \sum_{j=-1}^1 G(m,n)K_2(i-m,j-n)$$

$$G'(0,0) = 0 \times 0 + 0 \times 1 + 0 \times 0 + 0 \times 1 + 150 \times 0 + 151 \times 1 + 0 \times 0 \\ + 160 \times 1 + 152 \times 0$$

$$G'(0,0) = 311$$

Operasi konvolusi matriks B dengan kernel ketiga dengan *zero-padding* dan nilai *stride* 2:

$$B' = B * K_3$$

$$B' = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 100 & 110 & 101 & 102 & 100 & 107 & \dots \\ 0 & 106 & 107 & 104 & 100 & 110 & 106 & \dots \\ 0 & 107 & 104 & 103 & 104 & 102 & 108 & \dots \\ 0 & 104 & 107 & 106 & 105 & 101 & 100 & \dots \\ 0 & 110 & 101 & 104 & 107 & 100 & 104 & \dots \\ 0 & 100 & 104 & 105 & 109 & 110 & 100 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Contoh perhitungan operasi konvolusi untuk satu elemen matriks menggunakan persamaan 2.1, dapat dituliskan pada persamaan di bawah ini.

$$B'(0,0) = (B * K_3) = \sum_{i=-1}^1 \sum_{j=-1}^1 B(m,n) K_3(i-m, j-n)$$

$$B'(0,0) = 0 \times 1 + 0 \times 1 + 0 \times 1 + 0 \times 0 + 100 \times 0 + 110 \times 0 + 0 \times 1 + 106 \times 1 + 107 \times 1$$

$$B'(0,0) = 213$$

Hasil konvolusi tiap *channel* akan dijumlahkan sesuai dengan letak elemen yang bersesuaian sehingga setiap proses konvolusi ini menghasilkan satu buah *channel*. Contoh penjumlahan untuk salah satu elemen dapat dituliskan pada persamaan di bawah ini:

$$C(0,0) = R'(0,0) + G'(0,0) + B'(0,0)$$

$$C(0,0) = 333 + 311 + 213$$

$$C(0,0) = 857$$

Hasil konvolusi disusun berdasarkan penjumlahan elemen yang bersesuaian dengan ukuran 150x150 untuk tiap operasi konvolusinya sebanyak N *channel*. Contoh matriks hasil penjumlahan dari operasi konvolusi menunjukkan nilai pada satu buah *channel*:

$$C_1 = \begin{bmatrix} 857 & 1275 & 1281 & \dots \\ 1359 & 2082 & 2071 & \dots \\ 1363 & 2074 & 2056 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

d) Batch normalization

Operasi *batch normalization* dilakukan tiap *channel* hasil konvolusi. Berikut ini tahapan yang dilakukan dalam *batch normalization* untuk tiap *channel*.

- i) Menghitung nilai rata-rata menggunakan persamaan 2.2, dapat dituliskan pada persamaan 3.12.

$$\mu = \frac{1}{m} \sum_{i=1}^m C_i \text{total} \dots \dots \dots (3.12)$$

$$\mu = \frac{1}{22500} \sum_{i=1}^{22500} (857 + 1275 + 1281 + \dots)$$

$$\mu = 1602$$

- ii) Menghitung nilai varian menggunakan persamaan 2.3, dapat dituliskan pada persamaan 3.13.

$$\sigma^2 = \frac{1}{m} \sum_{i=1}^m (C_i - \mu)^2 \dots \dots \dots (3.13)$$

$$\sigma^2 = \frac{(857 - 1602)^2 + (1275 - 1602)^2 + (1281 - 1602)^2 + \dots}{22500}$$

$$\sigma^2 = 442.27$$

- iii) Normalisasi nilai tiap elemen menggunakan persamaan 2.4 dengan inisialisasi nilai epsilon (ϵ) = 0.001, dapat dituliskan pada persamaan 3.14.

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} \dots \dots \dots (3.14)$$

$$\hat{x} = \begin{bmatrix} \frac{857 - 1602}{\sqrt{442.27 + 0.001}} & \frac{1257 - 1602}{\sqrt{442.27 + 0.001}} & \frac{1281 - 1602}{\sqrt{442.27 + 0.001}} & \dots \\ \frac{1359 - 1602}{\sqrt{442.27 + 0.001}} & \frac{2082 - 1602}{\sqrt{442.27 + 0.001}} & \frac{2071 - 1602}{\sqrt{442.27 + 0.001}} & \dots \\ \frac{1363 - 1602}{\sqrt{442.27 + 0.001}} & \frac{2074 - 1602}{\sqrt{442.27 + 0.001}} & \frac{2056 - 1602}{\sqrt{442.27 + 0.001}} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$\hat{\hat{x}} = \begin{bmatrix} -35.42 & -15.54 & -15.26 & \dots \\ -11.55 & 22.82 & 22.30 & \dots \\ -11.36 & 22.44 & 21.58 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

- iv) Penskalaan dan pergeseran nilai tiap elemen dengan persamaan 2.5 dengan inisialisasi nilai gamma (γ) = 1 dan beta (β) = 0, dapat dituliskan pada persamaan 3.15.

$$y_i = \gamma \hat{x} + \beta \dots \dots \dots (3.15)$$

$$\hat{y} = 1 \begin{bmatrix} -35.42 & -15.54 & -15.26 & \dots \\ -11.55 & 22.82 & 22.30 & \dots \\ -11.36 & 22.44 & 21.58 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} + 0$$

$$\hat{y} = \begin{bmatrix} -35.42 & -15.54 & -15.26 & \dots \\ -11.55 & 22.82 & 22.30 & \dots \\ -11.36 & 22.44 & 21.58 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

e) ReLU6

Operasi ReLU6 dilakukan tiap *channel* hasil proses *batch normalization* dengan melakukan *thresholding* sebuah nilai yang memiliki rentang antara 0 sampai 6. Operasi ReLU6 dilakukan tiap elemen pada matriks dengan menggunakan persamaan 2.7, dapat dituliskan pada persamaan 3.16.

$$f(x) = \min(\max(0, x), 6) \dots \dots \dots (3.16)$$

$$f(x) = \begin{bmatrix} \min(\max(0, -35.42), 6) & \min(\max(0, -15.54), 6) & \min(\max(0, -15.26), 6) & \dots \\ \min(\max(0, -11.55), 6) & \min(\max(0, 22.82), 6) & \min(\max(0, 22.30), 6) & \dots \\ \min(\max(0, -11.36), 6) & \min(\max(0, 22.44), 6) & \min(\max(0, 21.58), 6) & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$f(x) = \begin{bmatrix} 0 & 0 & 0 & \dots \\ 0 & 6 & 6 & \dots \\ 0 & 6 & 6 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

3) Lapisan *bottleneck* terdiri dari 3 buah operasi yaitu konvolusi 1x1 (*expansion convolution*), *depthwise convolution* 3x3, *pointwise convolution* 1x1 (*projection convolution*). Lapisan *bottleneck* memiliki fitur tambahan *inverted residuals connection* ketika ukuran data masukan pada lapisan *bottleneck* sama dengan ukuran data keluaran pada lapisan *bottleneck*. Bagian ini akan menjumlahkan data masukan dengan hasil keluaran dari 3 operasi lapisan *bottleneck*. Pada lapisan *bottleneck* pertama memiliki sedikit perbedaan dengan lapisan *bottleneck* yang lain. Pada lapisan *bottleneck* yang pertama hanya terdiri dari 2 operasi konvolusi yaitu *depthwise convolution* dan *pointwise convolution*. Lapisan ini tidak ditambahkan operasi *expansion convolution* karena ukuran *channel* sudah diperbesar pada konvolusi sebelum *bottleneck* yang pertama. Berikut ini adalah tahapan pada lapisan *bottleneck*:

a) *Expansion convolution*

Operasi konvolusi ekspansi dilakukan operasi konvolusi terhadap kernel 1x1xM dengan nilai M menunjukkan jumlah *channel* masukan. Hasil konvolusi dari jumlah M *channel* akan berubah menjadi satu buah *channel*. Operasi ini memiliki masukan dengan ukuran 1xWxHxM dan dikonvolusikan dengan nilai *stride* 1 dan faktor ekspansi sebesar 6 sehingga ukuran *channel* berubah menjadi 6 kali lipat. Berikut ini adalah tahapan yang dilakukan dalam konvolusi *pointwise* untuk tiap *channel*.

- i) Data masukan dari proses sebelumnya berjumlah M *channel*. Contoh perhitungan diambil beberapa *channel* untuk menunjukkan proses konvolusi ekspansi. Berikut ini adalah 3 buah *channel* masukan.

$$C_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 6 & 6 & 6 & 6 & 6 & \dots \\ 0 & 6 & 6 & 6 & 6 & 6 & \dots \\ 0 & 6 & 6 & 6 & 6 & 6 & \dots \\ 0 & 6 & 6 & 6 & 6 & 6 & \dots \\ 0 & 6 & 6 & 6 & 6 & 6 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$C_2 = \begin{bmatrix} 6 & 6 & 6 & 6 & 6 & 6 & \dots \\ 6 & 6 & 6 & 6 & 6 & 6 & \dots \\ 6 & 6 & 6 & 6 & 6 & 6 & \dots \\ 6 & 6 & 6 & 6 & 6 & 6 & \dots \\ 6 & 6 & 6 & 6 & 6 & 6 & \dots \\ 6 & 6 & 6 & 6 & 6 & 6 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$C_3 = \begin{bmatrix} 0 & 6 & 0 & 6 & 0 & 6 & \dots \\ 6 & 6 & 6 & 6 & 6 & 6 & \dots \\ 0 & 6 & 6 & 6 & 6 & 6 & \dots \\ 6 & 6 & 6 & 6 & 6 & 6 & \dots \\ 0 & 6 & 6 & 6 & 6 & 6 & \dots \\ 6 & 6 & 6 & 6 & 6 & 6 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

- ii) Inisialisasi kernel sebanyak jumlah *channel* pada data masukan. Contoh perhitungan menggunakan beberapa kernel untuk mempermudah perhitungan konvolusi ekspansi.

$$K_1 = K_2 = K_3 = [1]$$

- iii) Operasi konvolusi ekspansi dilakukan dengan konvolusi tiap *channel* terhadap masing-masing kernel dengan nilai *stride* 1.

$$C_1' = C_1 * K_1$$

$$C_1' = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 6 & 6 & 6 & 6 & 6 & \dots \\ 0 & 6 & 6 & 6 & 6 & 6 & \dots \\ 0 & 6 & 6 & 6 & 6 & 6 & \dots \\ 0 & 6 & 6 & 6 & 6 & 6 & \dots \\ 0 & 6 & 6 & 6 & 6 & 6 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} * [1]$$

Contoh perhitungan operasi konvolusi untuk satu elemen matriks menggunakan persamaan 2.1, dapat dituliskan pada persamaan di bawah ini.

$$C_1'(0,0) = (C_1 * K_1) = \sum_{i=0}^0 C_1(m,n)K_1(i)$$

$$C_1'(0,0) = 0 \times 1$$

$$C_1'(0,0) = 0$$

$$C_2' = C_2 * K_2$$

$$C_2' = \begin{bmatrix} 6 & 6 & 6 & 6 & 6 & 6 & \dots \\ 6 & 6 & 6 & 6 & 6 & 6 & \dots \\ 6 & 6 & 6 & 6 & 6 & 6 & \dots \\ 6 & 6 & 6 & 6 & 6 & 6 & \dots \\ 6 & 6 & 6 & 6 & 6 & 6 & \dots \\ 6 & 6 & 6 & 6 & 6 & 6 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} * [1]$$

Contoh perhitungan operasi konvolusi untuk satu elemen matriks menggunakan persamaan 2.1, dapat dituliskan pada persamaan di bawah ini.

$$C_2'(0,0) = (C_2 * K_2) = \sum_{i=0}^0 C_2(m,n)K_2(i)$$

$$C_2'(0,0) = 6 \times 1$$

$$C_2'(0,0) = 6$$

$$C_3' = C_3 * K_3$$

$$C_3' = \begin{bmatrix} 0 & 6 & 0 & 6 & 0 & 6 & \dots \\ 6 & 6 & 6 & 6 & 6 & 6 & \dots \\ 0 & 6 & 6 & 6 & 6 & 6 & \dots \\ 6 & 6 & 6 & 6 & 6 & 6 & \dots \\ 0 & 6 & 6 & 6 & 6 & 6 & \dots \\ 6 & 6 & 6 & 6 & 6 & 6 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} * [1]$$

Contoh perhitungan operasi konvolusi untuk satu elemen matriks menggunakan persamaan 2.1, dapat dituliskan pada persamaan di bawah ini.

$$C_3'(0,0) = (C_3 * K_3) = \sum_{i=0}^0 C_3(m,n)K_3(i)$$

$$C_3'(0,0) = 0 \times 1$$

$$C_3'(0,0) = 0$$

Hasil konvolusi ketiga *channel* dapat dilihat pada matriks di bawah ini.

$$C_1' = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 6 & 6 & 6 & 6 & 6 & \dots \\ 0 & 6 & 6 & 6 & 6 & 6 & \dots \\ 0 & 6 & 6 & 6 & 6 & 6 & \dots \\ 0 & 6 & 6 & 6 & 6 & 6 & \dots \\ 0 & 6 & 6 & 6 & 6 & 6 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$C_2' = \begin{bmatrix} 6 & 6 & 6 & 6 & 6 & 6 & \dots \\ 6 & 6 & 6 & 6 & 6 & 6 & \dots \\ 6 & 6 & 6 & 6 & 6 & 6 & \dots \\ 6 & 6 & 6 & 6 & 6 & 6 & \dots \\ 6 & 6 & 6 & 6 & 6 & 6 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$C_3' = \begin{bmatrix} 0 & 6 & 0 & 6 & 0 & 6 & \dots \\ 6 & 6 & 6 & 6 & 6 & 6 & \dots \\ 0 & 6 & 6 & 6 & 6 & 6 & \dots \\ 6 & 6 & 6 & 6 & 6 & 6 & \dots \\ 0 & 6 & 6 & 6 & 6 & 6 & \dots \\ 6 & 6 & 6 & 6 & 6 & 6 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Operasi konvolusi ekspansi dilakukan dengan menjumlahkan elemen yang bersesuaian pada semua *channel* yang telah dilakukan konvolusi sehingga dapat dituliskan pada persamaan di bawah ini.

$$C_{new}(0,0) = C_1'(0,0) + C_2'(0,0) + C_3'(0,0)$$

$$C_{new}(0,0) = 0 + 6 + 0$$

$$C_{new}(0,0) = 6$$

Hasil operasi konvolusi *pointwise* dapat dilihat pada matriks di bawah ini.

$$C_{new} = \begin{bmatrix} 6 & 12 & 6 & 12 & 6 & 12 & \dots \\ 12 & 18 & 18 & 18 & 18 & 18 & \dots \\ 6 & 18 & 18 & 18 & 18 & 18 & \dots \\ 12 & 18 & 18 & 18 & 18 & 18 & \dots \\ 6 & 18 & 18 & 18 & 18 & 18 & \dots \\ 12 & 18 & 18 & 18 & 18 & 18 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

iv) *Batch normalization*

Operasi *batch normalization* dilakukan tiap *channel* hasil konvolusi. Berikut ini tahapan yang dilakukan dalam *batch normalization* untuk tiap *channel*.

$$C_{new} = \begin{bmatrix} 6 & 12 & 6 & 12 & 6 & 12 & \dots \\ 12 & 18 & 18 & 18 & 18 & 18 & \dots \\ 6 & 18 & 18 & 18 & 18 & 18 & \dots \\ 12 & 18 & 18 & 18 & 18 & 18 & \dots \\ 6 & 18 & 18 & 18 & 18 & 18 & \dots \\ 12 & 18 & 18 & 18 & 18 & 18 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

- (1) Menghitung nilai rata-rata pada salah satu *channel* yaitu C_{new} menggunakan persamaan 2.2, dapat dituliskan pada persamaan di bawah ini.

$$\mu = \frac{1}{m} \sum_{i=1}^m C_{i total}$$

$$\mu = \frac{1}{5625} \sum_{i=1}^{22500} (6 + 12 + 6 + 12 + 6 + 12 + \dots)$$

$$\mu = 15.33$$

- (2) Menghitung nilai varian menggunakan persamaan 2.3, dapat dituliskan pada persamaan di bawah ini.

$$\sigma^2 = \frac{1}{m} \sum_{i=1}^m (C_i - \mu)^2$$

$$\sigma^2 = \frac{(6 - 15.33)^2 + (12 - 15.33)^2 + (6 - 15.33)^2 + (12 - 15.33)^2 + \dots}{5625}$$

$$\sigma^2 = 18.8$$

- (3) Normalisasi nilai tiap elemen menggunakan persamaan 2.4 dengan inisialisasi nilai epsilon (ϵ) = 0.001, dapat dituliskan pada persamaan di bawah ini.

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

$$\hat{x} = \begin{bmatrix} \frac{6 - 15.33}{\sqrt{18.8 + 0.001}} & \frac{12 - 15.33}{\sqrt{18.8 + 0.001}} & \frac{6 - 15.33}{\sqrt{18.8 + 0.001}} & \frac{12 - 15.33}{\sqrt{18.8 + 0.001}} & \dots \\ \frac{12 - 15.33}{\sqrt{18.8 + 0.001}} & \frac{18 - 15.33}{\sqrt{18.8 + 0.001}} & \frac{18 - 15.33}{\sqrt{18.8 + 0.001}} & \frac{18 - 15.33}{\sqrt{18.8 + 0.001}} & \dots \\ \frac{6 - 15.33}{\sqrt{18.8 + 0.001}} & \frac{18 - 15.33}{\sqrt{18.8 + 0.001}} & \frac{18 - 15.33}{\sqrt{18.8 + 0.001}} & \frac{18 - 15.33}{\sqrt{18.8 + 0.001}} & \dots \\ \frac{12 - 15.33}{\sqrt{18.8 + 0.001}} & \frac{18 - 15.33}{\sqrt{18.8 + 0.001}} & \frac{18 - 15.33}{\sqrt{18.8 + 0.001}} & \frac{18 - 15.33}{\sqrt{18.8 + 0.001}} & \dots \\ \frac{6 - 15.33}{\sqrt{18.8 + 0.001}} & \frac{18 - 15.33}{\sqrt{18.8 + 0.001}} & \frac{18 - 15.33}{\sqrt{18.8 + 0.001}} & \frac{18 - 15.33}{\sqrt{18.8 + 0.001}} & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$\hat{\hat{x}} = \begin{bmatrix} -2.15 & -0.76 & -2.15 & -0.76 & \dots \\ -0.76 & 0.61 & 0.61 & 0.61 & \dots \\ -2.15 & 0.61 & 0.61 & 0.61 & \dots \\ -0.76 & 0.61 & 0.61 & 0.61 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

- (4) Penskalaan dan pergeseran nilai tiap elemen dengan persamaan 2.5 dengan inisialisasi nilai gamma (γ) = 1 dan beta (β) = 0, dapat dituliskan pada persamaan di bawah ini.

$$y_i = \gamma \hat{x} + \beta$$

$$\hat{y} = 1 \begin{bmatrix} -2.15 & -0.76 & -2.15 & -0.76 & \dots \\ -0.76 & 0.61 & 0.61 & 0.61 & \dots \\ -2.15 & 0.61 & 0.61 & 0.61 & \dots \\ -0.76 & 0.61 & 0.61 & 0.61 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} 0$$

$$\hat{y} = \begin{bmatrix} -2.15 & -0.76 & -2.15 & -0.76 & \dots \\ -0.76 & 0.61 & 0.61 & 0.61 & \dots \\ -2.15 & 0.61 & 0.61 & 0.61 & \dots \\ -0.76 & 0.61 & 0.61 & 0.61 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

v) ReLU6

Operasi ReLU6 dilakukan tiap *channel* hasil proses *batch normalization* dengan melakukan *thresholding* sebuah nilai yang memiliki rentang antara 0 sampai 6. Operasi ReLU6 dilakukan tiap elemen pada matriks dengan menggunakan persamaan 2.7, dapat dituliskan pada persamaan di bawah ini.

$$f(x) = \min(\max(0, x), 6)$$

$$f(x) =$$

$$\begin{bmatrix} \min(\max(0, -2.15), 6) & \min(\max(0, -0.76), 6) & \min(\max(0, -2.15), 6) & \min(\max(0, -0.76), 6) & \dots \\ \min(\max(0, -0.76), 6) & \min(\max(0, 0.61), 6) & \min(\max(0, 0.61), 6) & \min(\max(0, 0.61), 6) & \dots \\ \min(\max(0, -2.15), 6) & \min(\max(0, 0.61), 6) & \min(\max(0, 0.61), 6) & \min(\max(0, 0.61), 6) & \dots \\ \min(\max(0, -0.76), 6) & \min(\max(0, 0.61), 6) & \min(\max(0, 0.61), 6) & \min(\max(0, 0.61), 6) & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$f(x) = \begin{bmatrix} 0 & 0 & 0 & 0 & \dots \\ 0 & 0.61 & 0.61 & 0.61 & \dots \\ 0 & 0.61 & 0.61 & 0.61 & \dots \\ 0 & 0.61 & 0.61 & 0.61 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

b) *Depthwise convolution*

Operasi konvolusi *depthwise* dilakukan dengan memisahkan tiap *channel* masukan dan dilakukan operasi konvolusi terhadap kernel 3x3. Hasil konvolusi tiap *channel* masukan disusun ulang sesuai urutan konvolusi. Operasi ini memiliki masukan dengan ukuran 1xWxHxM dan dikonvolusikan dengan nilai *stride* 1 sehingga dimensi keluaran akan tetap. Berikut ini adalah tahapan yang dilakukan dalam konvolusi *depthwise* untuk tiap *channel*.

- i) Data masukan dari proses sebelumnya berjumlah M *channel* akan tetapi untuk mempermudah perhitungan diambil beberapa *channel* untuk menunjukkan proses konvolusi *depthwise*. Berikut ini adalah 3 buah *channel* masukan.

$$C_1 = C_2 = C_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & \dots \\ 0 & 0.61 & 0.61 & 0.61 & \dots \\ 0 & 0.61 & 0.61 & 0.61 & \dots \\ 0 & 0.61 & 0.61 & 0.61 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

ii) *Zero-padding* terhadap *channel* masukan.

$$C_1 = C_2 = C_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0.61 & 0.61 & 0.61 & \dots \\ 0 & 0 & 0.61 & 0.61 & 0.61 & \dots \\ 0 & 0 & 0.61 & 0.61 & 0.61 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

iii) Inisialisasi kernel sebanyak jumlah *channel* pada data masukan. Contoh perhitungan menggunakan beberapa kernel untuk mempermudah perhitungan konvolusi *depthwise*.

$$K_1 = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}; K_2 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}; K_3 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

iv) Operasi konvolusi *depthwise* dilakukan dengan konvolusi tiap *channel* terhadap masing-masing kernel dengan nilai *stride* 1.

$$C_1' = C_1 * K_1$$

$$C_1' = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0.61 & 0.61 & 0.61 & \dots \\ 0 & 0 & 0.61 & 0.61 & 0.61 & \dots \\ 0 & 0 & 0.61 & 0.61 & 0.61 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} * \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

Contoh perhitungan operasi konvolusi untuk satu elemen matriks menggunakan persamaan 2.1, dapat dituliskan pada persamaan di bawah ini.

$$C_1'(0,0) = (C_1 * K_1) = \sum_{i=-1}^1 \sum_{j=-1}^1 C_1(m,n) K_1(i-m, j-n)$$

$$C_1'(0,0) = 0 \times 1 + 0 \times 0 + 0 \times 1 + 0 \times 1 + 0 \times 0 + 0 \times 1 + 0 \times 1 \\ + 0 \times 0 + 0.61 \times 1$$

$$C_1'(0,0) = 0.61$$

$$C_2' = C_2 * K_2$$

$$C_2' = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0.61 & 0.61 & 0.61 & \dots \\ 0 & 0 & 0.61 & 0.61 & 0.61 & \dots \\ 0 & 0 & 0.61 & 0.61 & 0.61 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Contoh perhitungan operasi konvolusi untuk satu elemen matriks menggunakan persamaan 2.1, dapat dituliskan pada persamaan di bawah ini.

$$C_2'(0,0) = (C_2 * K_2) = \sum_{i=-1}^1 \sum_{j=-1}^1 C_2(m,n) K_2(i-m, j-n)$$

$$C_2'(0,0) = 0 \times 1 + 0 \times 1 + 0 \times 1 + 0 \times 0 + 0 \times 0 + 0 \times 0 + 0 \times 1 + 0 \times 1 + 0.61 \times 1$$

$$C_2'(0,0) = 0.61$$

$$C_3' = C_3 * K_3$$

$$C_3' = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0.61 & 0.61 & 0.61 & \dots \\ 0 & 0 & 0.61 & 0.61 & 0.61 & \dots \\ 0 & 0 & 0.61 & 0.61 & 0.61 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Contoh perhitungan operasi konvolusi untuk satu elemen matriks menggunakan persamaan 2.1, dapat dituliskan pada persamaan di bawah ini.

$$C_3'(0,0) = (C_3 * K_3) = \sum_{i=-1}^1 \sum_{j=-1}^1 C_3(m,n) K_3(i-m, j-n)$$

$$C_3'(0,0) = 0 \times 0 + 0 \times 1 + 0 \times 0 + 0 \times 0 + 0 \times 1 + 0 \times 0 + 0 \times 0 + 0 \times 1 + 0.61 \times 0$$

$$C_3'(0,0) = 0$$

v) *Batch normalization*

Operasi *batch normalization* dilakukan tiap *channel* hasil konvolusi. Berikut ini tahapan yang dilakukan dalam *batch normalization* untuk tiap *channel*.

$$C_1' = \begin{bmatrix} 0.61 & 1.22 & \dots \\ 1.22 & 3.66 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

$$C_2' = \begin{bmatrix} 0.61 & 1.83 & \dots \\ 1.22 & 3.66 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

$$C_3' = \begin{bmatrix} 0 & 0.61 & \dots \\ 0 & 1.83 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

(1) Menghitung nilai rata-rata pada salah satu *channel* yaitu C_1 menggunakan persamaan 2.2, dapat dituliskan pada persamaan di bawah ini.

$$\mu = \frac{1}{m} \sum_{i=1}^m C_{i total}$$

$$\mu = \frac{1}{5625} \sum_{i=1}^{22500} (0.61 + 1.22 + \dots)$$

$$\mu = 1.67$$

- (2) Menghitung nilai varian menggunakan persamaan 2.3, dapat dituliskan pada persamaan di bawah ini.

$$\sigma^2 = \frac{1}{m} \sum_{i=1}^m (C_i - \mu)^2$$

$$\sigma^2 = \frac{(0.61 - 1.67)^2 + (1.22 - 1.67)^2 + \dots}{5625}$$

$$\sigma^2 = 1.37$$

- (3) Normalisasi nilai tiap elemen menggunakan persamaan 2.4 dengan inisialisasi nilai epsilon (ϵ) = 0.001, dapat dituliskan pada persamaan di bawah ini.

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

$$\hat{x} = \begin{bmatrix} \frac{0.61 - 1.67}{\sqrt{1.37 + 0.001}} & \frac{1.22 - 1.67}{\sqrt{1.37 + 0.001}} & \dots \\ \frac{1.22 - 22}{\sqrt{1.37 + 0.001}} & \frac{3.66 - 22}{\sqrt{1.37 + 0.001}} & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

$$\hat{\hat{x}} = \begin{bmatrix} -0.90 & -0.38 & \dots \\ -0.38 & 1.69 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

- (4) Penskalaan dan pergeseran nilai tiap elemen dengan persamaan 2.5 dengan inisialisasi nilai gamma (γ) = 1 dan beta (β) = 0, dapat dituliskan pada persamaan di bawah ini.

$$y_i = \gamma \hat{x} + \beta$$

$$\hat{y} = 1 \begin{bmatrix} -0.90 & -0.38 & \dots \\ -0.38 & 1.69 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix} + 0$$

$$\hat{y} = \begin{bmatrix} -0.90 & -0.38 & \dots \\ -0.38 & 1.69 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

Hasil akhir *batch normalization* dari ketiga contoh *channel* input, dapat dilihat pada matriks di bawah ini.

$$C_1 = \begin{bmatrix} -0.90 & -0.38 & \dots \\ -0.38 & 1.69 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

$$C_2 = \begin{bmatrix} -1.06 & 0 & \dots \\ -0.53 & 1.60 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

$$C_3 = \begin{bmatrix} -0.82 & 0 & \dots \\ -0.82 & 1.64 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

vi) ReLU6

Operasi ReLU6 dilakukan tiap *channel* hasil proses *batch normalization* dengan melakukan *thresholding* sebuah nilai yang memiliki rentang antara 0 sampai 6. Operasi ReLU6 dilakukan tiap elemen pada matriks dengan menggunakan persamaan 2.7, dapat dituliskan pada persamaan di bawah ini.

$$f(x) = \min(\max(0, x), 6)$$

$$f(x) = \begin{bmatrix} \min(\max(0, -0.90), 6) & \min(\max(0, -0.38), 6) & \dots \\ \min(\max(0, -0.38), 6) & \min(\max(0, 1.69), 6) & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

$$f(x) = \begin{bmatrix} 0 & 0 & \dots \\ 0 & 1.69 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

Hasil akhir ReLU6 dari ketiga contoh *channel* input, dapat dilihat pada matriks di bawah ini.

$$C_1 = \begin{bmatrix} 0 & 0 & \dots \\ 0 & 1.69 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

$$C_2 = \begin{bmatrix} 0 & 0 & \dots \\ 0 & 1.60 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

$$C_3 = \begin{bmatrix} 0 & 0 & \dots \\ 0 & 1.64 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

c) *Pointwise convolution*

Operasi konvolusi *pointwise* dilakukan operasi konvolusi terhadap kernel $1 \times 1 \times C$ dengan nilai C menunjukkan jumlah *channel* masukan. Hasil konvolusi dari C *channel* akan berubah menjadi satu buah *channel*. Operasi ini memiliki masukan dengan ukuran $1 \times W \times H \times M$ dan dikonvolusikan dengan nilai *stride* 1 sebanyak N kali sehingga ukuran *channel* yang berubah. Berikut ini adalah tahapan yang dilakukan dalam konvolusi *pointwise* untuk tiap *channel*.

- i) Data masukan dari proses sebelumnya berjumlah 32 *channel* akan tetapi untuk mempermudah perhitungan diambil beberapa *channel* untuk menunjukkan proses konvolusi *pointwise*. Berikut ini adalah 3 buah *channel* masukan.

$$C_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & \dots \\ 0 & 1.69 & 1.69 & 1.69 & \dots \\ 0 & 1.69 & 1.69 & 1.69 & \dots \\ 0 & 1.69 & 1.69 & 1.69 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$C_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & \dots \\ 0 & 1.60 & 1.60 & 1.60 & \dots \\ 0 & 1.60 & 1.60 & 1.60 & \dots \\ 0 & 1.60 & 1.60 & 1.60 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$C_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & \dots \\ 0 & 1.64 & 1.64 & 1.64 & \dots \\ 0 & 1.64 & 1.64 & 1.64 & \dots \\ 0 & 1.64 & 1.64 & 1.64 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

- ii) Inisialisasi kernel sebanyak jumlah *channel* pada data masukan yaitu 32 akan tetapi untuk mempermudah perhitungan diambil beberapa kernel untuk proses konvolusi.

$$K_1 = K_2 = K_3 = [1]$$

- iii) Operasi konvolusi *pointwise* dilakukan dengan konvolusi tiap *channel* terhadap masing-masing kernel dengan nilai *stride* 1.

$$C_1' = C_1 * K_1$$

$$C_1' = \begin{bmatrix} 0 & 0 & 0 & 0 & \dots \\ 0 & 1.69 & 1.69 & 1.69 & \dots \\ 0 & 1.69 & 1.69 & 1.69 & \dots \\ 0 & 1.69 & 1.69 & 1.69 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} * [1]$$

Contoh perhitungan operasi konvolusi untuk satu elemen matriks menggunakan persamaan 2.1, dapat dituliskan pada persamaan di bawah ini.

$$C_1'(0,0) = (C_1 * K_1) = \sum_{i=0}^0 C_1(m,n)K_1(i)$$

$$C_1'(0,0) = 0 \times 1$$

$$C_1'(0,0) = 0$$

$$C_2' = C_2 * K_2$$

$$C_2' = \begin{bmatrix} 0 & 0 & 0 & 0 & \dots \\ 0 & 1.60 & 1.60 & 1.60 & \dots \\ 0 & 1.60 & 1.60 & 1.60 & \dots \\ 0 & 1.60 & 1.60 & 1.60 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} * [1]$$

Contoh perhitungan operasi konvolusi untuk satu elemen matriks menggunakan persamaan 2.1, dapat dituliskan pada persamaan di bawah ini.

$$C_2'(0,0) = (C_2 * K_2) = \sum_{i=0}^0 C_2(m,n)K_2(i)$$

$$C_2'(0,0) = 0 \times 1$$

$$C_2'(0,0) = 0$$

$$C_3' = C_3 * K_3$$

$$C_3' = \begin{bmatrix} 0 & 0 & 0 & 0 & \dots \\ 0 & 1.64 & 1.64 & 1.64 & \dots \\ 0 & 1.64 & 1.64 & 1.64 & \dots \\ 0 & 1.64 & 1.64 & 1.64 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} * [1]$$

Contoh perhitungan operasi konvolusi untuk satu elemen matriks menggunakan persamaan 2.1, dapat dituliskan pada persamaan di bawah ini.

$$C_3'(0,0) = (C_3 * K_3) = \sum_{i=0}^0 C_3(m,n)K_3(i)$$

$$C_3'(0,0) = 0 \times 1$$

$$C_3'(0,0) = 0$$

Hasil konvolusi ketiga *channel* dapat dilihat pada matriks di bawah ini.

$$C_1' = \begin{bmatrix} 0 & 0 & 0 & 0 & \dots \\ 0 & 1.69 & 1.69 & 1.69 & \dots \\ 0 & 1.69 & 1.69 & 1.69 & \dots \\ 0 & 1.69 & 1.69 & 1.69 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$C_2' = \begin{bmatrix} 0 & 0 & 0 & 0 & \dots \\ 0 & 1.60 & 1.60 & 1.60 & \dots \\ 0 & 1.60 & 1.60 & 1.60 & \dots \\ 0 & 1.60 & 1.60 & 1.60 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$C_3' = \begin{bmatrix} 0 & 0 & 0 & 0 & \dots \\ 0 & 1.64 & 1.64 & 1.64 & \dots \\ 0 & 1.64 & 1.64 & 1.64 & \dots \\ 0 & 1.64 & 1.64 & 1.64 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Operasi konvolusi *pointwise* dilakukan dengan menjumlahkan elemen yang bersesuaian pada semua *channel* yang telah dilakukan konvolusi. Contoh

penjumlahan untuk salah satu elemen dapat dituliskan pada persamaan di bawah ini:

$$C_{new}(0,0) = C_1'(0,0) + C_2'(0,0) + C_3'(0,0)$$

$$C_{new}(0,0) = 0 + 0 + 0$$

$$C_{new}(0,0) = 0$$

Hasil operasi konvolusi *pointwise* dapat dilihat pada matriks di bawah ini.

$$C_{new} = \begin{bmatrix} 0 & 0 & 0 & 0 & \dots \\ 0 & 4.93 & 4.93 & 4.93 & \dots \\ 0 & 4.93 & 4.93 & 4.93 & \dots \\ 0 & 4.93 & 4.93 & 4.93 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

iv) *Batch normalization*

Operasi *batch normalization* dilakukan tiap *channel* hasil konvolusi. Berikut ini tahapan yang dilakukan dalam *batch normalization* untuk tiap *channel*.

$$C_{new} = \begin{bmatrix} 0 & 0 & 0 & 0 & \dots \\ 0 & 4.93 & 4.93 & 4.93 & \dots \\ 0 & 4.93 & 4.93 & 4.93 & \dots \\ 0 & 4.93 & 4.93 & 4.93 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

- (1) Menghitung nilai rata-rata pada salah satu *channel* yaitu C_{new} menggunakan persamaan 2.2, dapat dituliskan pada persamaan di bawah ini.

$$\mu = \frac{1}{m} \sum_{i=1}^m C_{i total}$$

$$\mu = \frac{1}{5625} \sum_{i=1}^{22500} (0 + 0 + 0 + \dots)$$

$$\mu = 2.77$$

- (2) Menghitung nilai varian menggunakan persamaan 2.3, dapat dituliskan pada persamaan di bawah ini.

$$\sigma^2 = \frac{1}{m} \sum_{i=1}^m (C_i - \mu)^2$$

$$\sigma^2 = \frac{(0 - 2.77)^2 + (0 - 2.77)^2 + (0 - 2.77)^2 + (0 - 2.77)^2 + \dots}{5625}$$

$$\sigma^2 = 5.98$$

- (3) Normalisasi nilai tiap elemen menggunakan persamaan 2.4 dengan inisialisasi nilai epsilon (ϵ) = 0.001, dapat dituliskan pada persamaan di bawah ini.

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

$$\hat{x} = \begin{bmatrix} \frac{0 - 2.77}{\sqrt{5.98 + 0.001}} & \frac{0 - 2.77}{\sqrt{5.98 + 0.001}} & \frac{0 - 2.77}{\sqrt{5.98 + 0.001}} & \frac{0 - 2.77}{\sqrt{5.98 + 0.001}} & \dots \\ \frac{0 - 2.77}{\sqrt{5.98 + 0.001}} & \frac{4.39 - 2.77}{\sqrt{5.98 + 0.001}} & \frac{4.39 - 2.77}{\sqrt{5.98 + 0.001}} & \frac{4.39 - 2.77}{\sqrt{5.98 + 0.001}} & \dots \\ \frac{0 - 2.77}{\sqrt{5.98 + 0.001}} & \frac{4.39 - 2.77}{\sqrt{5.98 + 0.001}} & \frac{4.39 - 2.77}{\sqrt{5.98 + 0.001}} & \frac{4.39 - 2.77}{\sqrt{5.98 + 0.001}} & \dots \\ \frac{0 - 2.77}{\sqrt{5.98 + 0.001}} & \frac{4.39 - 2.77}{\sqrt{5.98 + 0.001}} & \frac{4.39 - 2.77}{\sqrt{5.98 + 0.001}} & \frac{4.39 - 2.77}{\sqrt{5.98 + 0.001}} & \dots \\ \frac{0 - 2.77}{\sqrt{5.98 + 0.001}} & \frac{4.39 - 2.77}{\sqrt{5.98 + 0.001}} & \frac{4.39 - 2.77}{\sqrt{5.98 + 0.001}} & \frac{4.39 - 2.77}{\sqrt{5.98 + 0.001}} & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$\hat{\hat{x}} = \begin{bmatrix} -1.13 & -1.13 & -1.13 & -1.13 & \dots \\ -1.13 & 0.66 & 0.66 & 0.66 & \dots \\ -1.13 & 0.66 & 0.66 & 0.66 & \dots \\ -1.13 & 0.66 & 0.66 & 0.66 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

- (4) Penskalaan dan pergeseran nilai tiap elemen dengan persamaan 2.5 dengan inisialisasi nilai gamma (γ) = 1 dan beta (β) = 0, dapat dituliskan pada persamaan di bawah ini.

$$y_i = \gamma \hat{x} + \beta$$

$$\hat{y} = 1 \begin{bmatrix} -1.13 & -1.13 & -1.13 & -1.13 & \dots \\ -1.13 & 0.66 & 0.66 & 0.66 & \dots \\ -1.13 & 0.66 & 0.66 & 0.66 & \dots \\ -1.13 & 0.66 & 0.66 & 0.66 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} + 0$$

$$\hat{y} = \begin{bmatrix} -1.13 & -1.13 & -1.13 & -1.13 & \dots \\ -1.13 & 0.66 & 0.66 & 0.66 & \dots \\ -1.13 & 0.66 & 0.66 & 0.66 & \dots \\ -1.13 & 0.66 & 0.66 & 0.66 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

d) *Inverted residuals connection*

Inverted residuals connection dilakukan ketika ukuran data yang sama antara data masukan sebelum lapisan *bottleneck* dan ukuran data keluaran setelah lapisan *bottleneck*. Operasi ini dilakukan dengan penjumlahan data masukan dengan data keluaran. Sebagai contoh sederhana operasi ini dapat dilihat pada matriks di bawah ini.

$$C_{in} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 6 & 6 & 6 & 6 & 6 & \dots \\ 0 & 6 & 6 & 6 & 6 & 6 & \dots \\ 0 & 6 & 6 & 6 & 6 & 6 & \dots \\ 0 & 6 & 6 & 6 & 6 & 6 & \dots \\ 0 & 6 & 6 & 6 & 6 & 6 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$C_{out} = \begin{bmatrix} -1.13 & -1.13 & -1.13 & -1.13 & -1.13 & -1.13 & \dots \\ -1.13 & 0.66 & 0.66 & 0.66 & 0.66 & 0.66 & \dots \\ -1.13 & 0.66 & 0.66 & 0.66 & 0.66 & 0.66 & \dots \\ -1.13 & 0.66 & 0.66 & 0.66 & 0.66 & 0.66 & \dots \\ -1.13 & 0.66 & 0.66 & 0.66 & 0.66 & 0.66 & \dots \\ -1.13 & 0.66 & 0.66 & 0.66 & 0.66 & 0.66 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

C_{in} menunjukkan matriks dari *channel* masukan dan C_{out} menunjukkan matriks dari *channel* hasil operasi *bottleneck*. Kedua *channel* memiliki ukuran data yang sama sehingga dilakukan proses penjumlahan pada letak elemen yang bersesuaian. Contoh penjumlahan untuk salah satu elemen dapat dituliskan pada persamaan di bawah ini:

$$C_{new}(0,0) = C_{in}(0,0) + C_{out}(0,0)$$

$$C_{new}(0,0) = 0 + (-1.33)$$

$$C_{new}(0,0) = -1.33$$

Hasil operasi *inverted residuals connection* dapat dilihat pada matriks di bawah ini.

$$C_{new} = \begin{bmatrix} -1.13 & -1.13 & -1.13 & -1.13 & -1.13 & -1.13 & \dots \\ -1.13 & 6.66 & 6.66 & 6.66 & 6.66 & 6.66 & \dots \\ -1.13 & 6.66 & 6.66 & 6.66 & 6.66 & 6.66 & \dots \\ -1.13 & 6.66 & 6.66 & 6.66 & 6.66 & 6.66 & \dots \\ -1.13 & 6.66 & 6.66 & 6.66 & 6.66 & 6.66 & \dots \\ -1.13 & 6.66 & 6.66 & 6.66 & 6.66 & 6.66 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Operasi *bottleneck* dilakukan berulang sebanyak 17 kali dengan informasi yang dapat dilihat pada Tabel 2.3. Setelah itu dilanjutkan dengan konvolusi 1x1 pada bagian akhir model MobileNet-V2 sehingga citra masukan telah melalui 19 buah lapisan untuk ekstraksi fitur.

- 4) SSD memiliki 6 buah *feature maps* yang diambil dari 2 lapisan MobileNet-V2 dan 4 lapisan tambahan. Dua lapisan MobileNet-V2 diambil dari hasil *expansion convolution* pada lapisan *bottleneck* 14 (lapisan 15) dan hasil konvolusi 1x1 pada lapisan 19. Empat buah lapisan tambahan masing-masing terdiri dari 2 buah operasi konvolusi yaitu konvolusi 1x1 dan konvolusi 3x3. *Feature maps* diambil dari hasil konvolusi 3x3 tiap

lapisan tambahan. Informasi mengenai pengambilan 6 buah *feature maps* dapat dilihat pada Tabel 2.3. Hasil dari proses ini menghasilkan prediksi lokasi dan kelas objek.

- 5) Hasil pengambilan *feature maps* berupa lokasi dan kelas objek terdiri dari beberapa informasi yang dapat dituliskan pada matriks di bawah ini.

$$\begin{pmatrix} cx \\ cy \\ w \\ h \end{pmatrix} = \begin{pmatrix} 87 \\ 183 \\ 8 \\ 12 \end{pmatrix} \text{ dan } \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix} = \begin{pmatrix} -1.2 \\ 0.6 \\ -0.6 \\ 3.6 \\ -2.4 \end{pmatrix}$$

Matriks sebelah kiri menunjukkan lokasi objek yang terdiri dari lokasi titik awal (x,y) serta ukuran *bounding box* yang ditunjukkan oleh panjang dan lebar. Matriks sebelah kanan menunjukkan nilai masing-masing kelas, kelas 0 menunjukkan sebagai *background* dan kelas 1-4 menunjukkan objek sesuai *dataset*. Untuk perhitungan evaluasi model perlu dilakukan konversi terhadap kedua hasil tersebut. Berikut adalah proses yang dilakukan.

- a) Mencari lokasi objek dilakukan dengan mencari nilai xmin, ymin, xmax, dan ymax yang dituliskan dengan perhitungan di bawah ini:

$$\begin{pmatrix} xmin \\ ymin \\ xmax \\ ymax \end{pmatrix} = \begin{pmatrix} x - w/2 \\ y - w/2 \\ x + w/2 \\ y + h/2 \end{pmatrix}$$

$$\begin{pmatrix} xmin \\ ymin \\ xmax \\ ymax \end{pmatrix} = \begin{pmatrix} 87 - 4 \\ 183 - 6 \\ 87 + 4 \\ 183 + 6 \end{pmatrix}$$

$$\begin{pmatrix} xmin \\ ymin \\ xmax \\ ymax \end{pmatrix} = \begin{pmatrix} 83 \\ 177 \\ 91 \\ 189 \end{pmatrix}$$

- b) Mencari probabilitas kelas objek dilakukan dengan perhitungan melalui fungsi *sigmoid* terhadap hasil *feature maps* menggunakan persamaan 2.8. Contoh perhitungan menggunakan fungsi *sigmoid* dapat dituliskan pada persamaan 3.17.

$$f(x) = \frac{1}{(1+e^{-x})} \dots\dots\dots(3.17)$$

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix} = \begin{pmatrix} \frac{1}{(1 + e^{1.2})} \\ \frac{1}{(1 + e^{-0.6})} \\ \frac{1}{(1 + e^{0.6})} \\ \frac{1}{(1 + e^{-3.6})} \\ \frac{1}{(1 + e^{2.4})} \end{pmatrix}$$

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix} = \begin{pmatrix} 0.23 \\ 0.64 \\ 0.35 \\ 0.97 \\ 0.08 \end{pmatrix}$$

- 6) Hasil pengambilan *feature maps* dilakukan proses *Non-Maximum Suppression* (NMS) untuk mengurangi jumlah irisan *bounding box* dan digantikan dengan probabilitas kelas yang paling tinggi. Proses NMS dilakukan dengan perhitungan nilai *Intersection over Union* (IoU) menggunakan persamaan 2.42, dapat dituliskan pada persamaan 3.18.

$$Ground\ truth = \begin{pmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} 82 \\ 176 \\ 90 \\ 188 \end{pmatrix}$$

$$Bounding\ box = \begin{pmatrix} x_3 \\ y_3 \\ x_4 \\ y_4 \end{pmatrix} = \begin{pmatrix} 83 \\ 177 \\ 91 \\ 189 \end{pmatrix}$$

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} = \frac{Area(B \cap B_{gt})}{Area(B \cup B_{gt})} \dots \dots \dots (3.18)$$

$$Area\ of\ Overlap = (\min(x_2, x_4) - \max(x_1, x_3)) * (\min(y_2, y_4) - \max(y_1, y_3))$$

$$Area\ of\ Overlap = (\min(90, 91) - \max(82, 83)) * (\min(188, 189) - \max(176, 177))$$

$$Area\ of\ Overlap = (90 - 83) * (188 - 177)$$

$$Area\ of\ Overlap = 77$$

$$Area\ of\ Union = ((x_2 - x_1) * (y_2 - y_1) + (x_4 - x_3) * (y_4 - y_3)) - Area\ of\ Overlap$$

$$Area\ of\ Union = (90 - 82) * (188 - 176) + (91 - 83) * (189 - 177) - 77$$

$$Area\ of\ Union = (8) * (12) + (8) * (11) - 77$$

$$Area\ of\ Union = 107$$

$$IoU = \frac{77}{107}$$

$$\text{IoU} = 0.71$$

IoU memiliki nilai ambang batas yang sudah ditentukan dalam pelatihan ketika nilai IoU ≥ 0.5 maka *bounding box* termasuk ke dalam objek yang terdeteksi dan sebaliknya. Hasil irisan *bounding box* akan direduksi sehingga *bounding box* yang terpilih memiliki nilai IoU dan probabilitas kelas yang paling besar.

- 7) Menghitung nilai *loss* dilakukan untuk mengetahui hasil dari proses pembelajaran model SSD MobileNet-V2 yang dilakukan dengan persamaan 2.11.

$$\text{Bounding box } (l) = \begin{pmatrix} cx \\ cy \\ w \\ h \end{pmatrix} = \begin{pmatrix} 87 \\ 183 \\ 8 \\ 12 \end{pmatrix}$$

$$\text{Ground truth box } (g) = \begin{pmatrix} cx \\ cy \\ w \\ h \end{pmatrix} = \begin{pmatrix} 86 \\ 182 \\ 8 \\ 12 \end{pmatrix}$$

$$\text{Class probability } (x) = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} = \begin{pmatrix} 0.23 \\ 0.64 \\ 0.35 \\ 0.97 \\ 0.08 \end{pmatrix}$$

$$\text{Ground class } = (c) = (3)$$

Perhitungan nilai *loss* dapat dituliskan pada persamaan 3.19

$$L(x, c, l, g) = \frac{1}{N} (L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \dots\dots\dots (3.19)$$

Proses diawali dengan mencari nilai *confidence loss* / *classification loss* menggunakan persamaan 2.14 sampai 2.15, dapat dituliskan pada persamaan 3.20.

$$L_{conf}(x, c) = -\sum_{i \in Pos}^N x_{ij}^p \log(\hat{c}_i^p) - \sum_{i \in Neg} \log(\hat{c}_i^0) \dots\dots\dots (3.20)$$

Proses ini melakukan perhitungan terhadap kelas positif yang ditandai dengan x_{ij}^p , nilai ini akan bernilai 1 ketika sesuai dengan kelas *ground truth* dan bernilai 0 ketika tidak sesuai dengan kelas *ground truth* sehingga perhitungan hanya kelas yang bersesuaian dengan *ground truth*. Contoh perhitungan dapat dituliskan pada persamaan 3.21.

$$L_{conf}(x, c) = -\sum_{i \in Pos}^N x_{ij}^p \log\left(\frac{e^{(c_i^p)}}{\sum_p e^{(c_i^p)}}\right) - \sum_{i \in Neg} \log\left(\frac{e^{(c_i^0)}}{\sum_p e^{(c_i^p)}}\right) \dots\dots\dots (3.21)$$

$$L_{conf}(x, c) = -\left(\log\left(\frac{e^{0.97}}{e^{(0.23+0.64+0.35+0.97+0.08)}}\right)\right) - \left(\log\left(\frac{e^{0.23}}{e^{(0.23+0.64+0.35+0.97+0.08)}}\right)\right)$$

$$L_{conf}(x, c) = -(-0.564) - (-0.885)$$

$$L_{conf}(x, c) = 1.450$$

Proses pertama, menghitung nilai *localization loss / regression loss* yang ditandai dengan x_{ij}^k , nilai ini akan bernilai 1 ketika hasil deteksi memenuhi nilai ambang batas $\text{IoU} \geq 0.5$ (terdeteksi). Perhitungan nilai *loss* dilakukan antara lokasi prediksi dengan *ground truth* dengan melibatkan proses *smooth L1* pada persamaan 2.11 sampai 2.12. Contoh perhitungan dapat dituliskan pada persamaan 3.22.

$$L_{loc}(x, l, g) = \sum_{i \in Pos} \sum_{m \in (cx, cy, w, h)} x_{ij}^k \text{smooth}_{L1}(l_i^m - \hat{g}_j^m) \dots\dots\dots(3.22)$$

$$L_{loc}(x, l, g) = \text{smooth}_{L1}(l^{cx} - \hat{g}^{cx}) + \text{smooth}_{L1}(l^{cy} - \hat{g}^{cy}) + \text{smooth}_{L1}(l^w - \hat{g}^w) \\ + \text{smooth}_{L1}(l^h - \hat{g}^h)$$

Proses kedua, menghitung nilai *smooth L1* menggunakan persamaan 2.12 sehingga contoh perhitungan dapat dituliskan pada persamaan di bawah ini.

$$L_{loc}(x, l, g) = \text{smooth}_{L1}(87 - 86) + \text{smooth}_{L1}(183 - 182) + \text{smooth}_{L1}(8 - 8) \\ + \text{smooth}_{L1}(12 - 12)$$

$$L_{loc}(x, l, g) = \text{smooth}_{L1}(1) + \text{smooth}_{L1}(1) + \text{smooth}_{L1}(0) + \text{smooth}_{L1}(0)$$

$$L_{loc}(x, l, g) = (1 - 0.5) + (1 - 0.5) + (0.5 \times 0^2) + 0.5 \times 0^2)$$

$$L_{loc}(x, l, g) = 1$$

Proses ketiga, menghitung nilai *loss* secara keseluruhan dengan menjumlahkan antara L_{conf} dan L_{loc} . Parameter lain yang digunakan adalah nilai N (jumlah deteksi positif) sebagai contoh digunakan nilai 1 dan $\alpha=1$, contoh perhitungan dapat dituliskan pada persamaan di bawah ini.

$$L(x, c, l, g) = \frac{1}{1} (1.450 + 1 \times 1)$$

$$L(x, c, l, g) = 2.450$$

- 8) Perubahan bobot dilakukan dengan metode RMSProp untuk mengoptimalkan perubahan bobot menggunakan persamaan 2.9 sampai 2.10. Beberapa parameter yang digunakan dalam proses perhitungan ini yaitu $\gamma=0.9$, δ menunjukkan nilai gradien yang diambil dari nilai *loss* pada proses sebelumnya, dan nilai rata-rata perubahan eksponensial awal yang diinisialisasi dengan 0. Contoh perhitungan dapat dituliskan pada persamaan 3.32 sampai 3.24.

$$E[\delta^2]_t = \gamma E[\delta^2]_{t-1} + (1 - \gamma) \delta_t^2 \dots\dots\dots(3.23)$$

$$E[\delta^2]_t = (0.9)0 + (1 - 0.9)(2.450)^2$$

$$E[\delta^2]_t = 0.600$$

Setelah itu, melakukan pembaruan bobot yang melibatkan beberapa parameter yaitu *learning rate* ($\alpha = 0.001$), nilai epsilon ($\epsilon = 1.0$), nilai *loss*, dan inisialisasi bobot awal

dengan nilai 1. Contoh perhitungan pembaruan bobot dapat dituliskan pada persamaan di bawah ini.

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{E[\delta^2]_t + \varepsilon}} \delta_t \dots\dots\dots(3.24)$$

$$\theta_t = 1 - \frac{0.001}{\sqrt{0.600 + 1.0}} (2.450)$$

$$\theta_t = 0.998$$

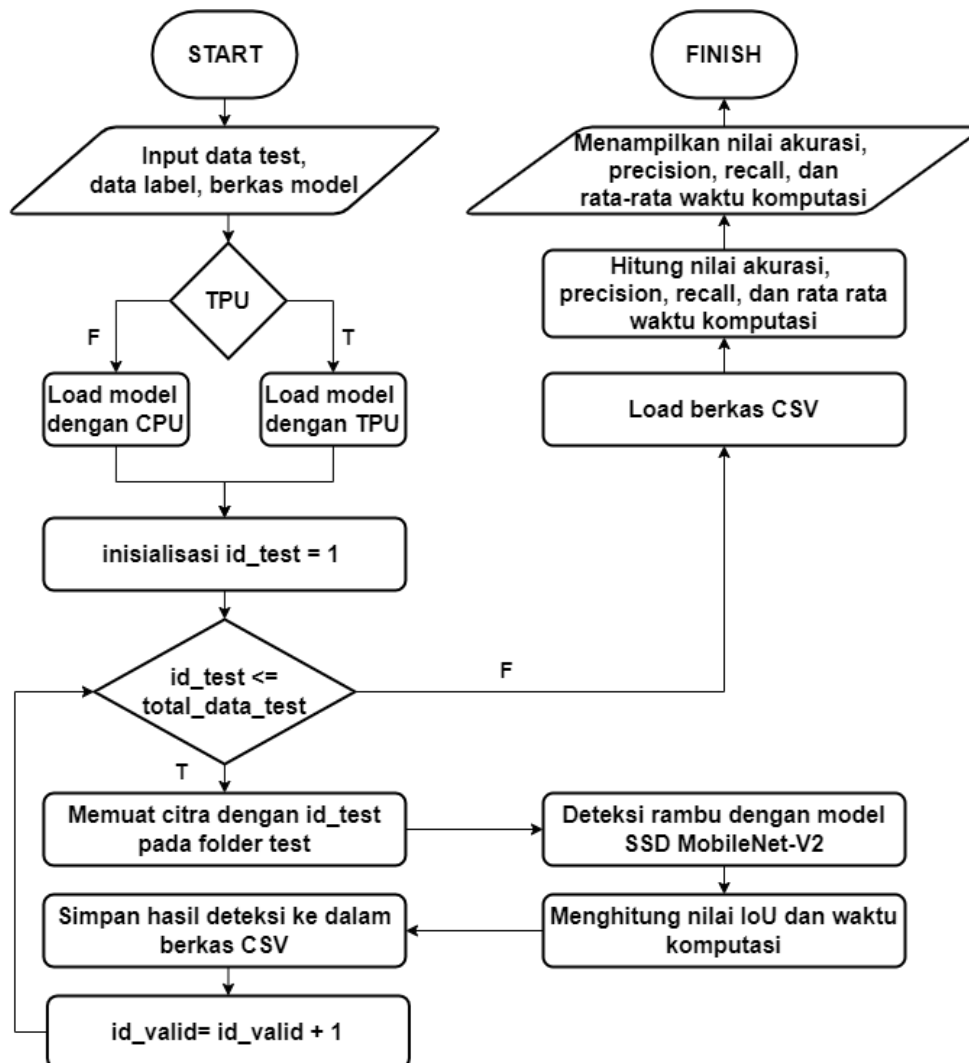
- 9) Proses perhitungan nilai *loss* dan pembaruan bobot dilakukan setiap *batch* pemrosesan data atau ukuran 1 *epoch*. Pelatihan model SSD MobileNet-V2 dilakukan hingga jumlah *epoch* yang ditentukan kemudian model disimpan untuk digunakan pada proses berikutnya.

3.5. Pengujian Model

Pengujian model dilakukan untuk mengetahui kinerja dari masing-masing model terhadap beberapa parameter masukan seperti *learning rate*, nilai *batch*, dan jumlah *epoch*. Tahapan ini menggunakan model yang sudah disimpan dari hasil pelatihan kemudian dimuat ke dalam program untuk proses deteksi. Pengujian model menggunakan data *test* yang berjumlah 113 anotasi) untuk pengujian melalui Jupyter Notebook dan data tambahan berjumlah 43 untuk pengujian melalui Raspberry Pi. Alasan dari penggunaan data tambahan dikarenakan ukuran objek yang terlalu kecil pada *dataset* sehingga diperlukan data tambahan yang bersesuaian untuk pengujian melalui Raspberry Pi. Kinerja model dinilai dari nilai akurasi, *precision*, *recall*, dan rata-rata waktu komputasi. Pengujian model dibagi menjadi dua buah skema yaitu pengujian melalui Jupyter Notebook dan pengujian melalui Raspberry Pi.

3.5.1. Skema Pengujian Jupyter Notebook

Skema pengujian melalui Jupyter Notebook memiliki beberapa tahapan yang perlu dilakukan, dapat dilihat pada Gambar 3.19.

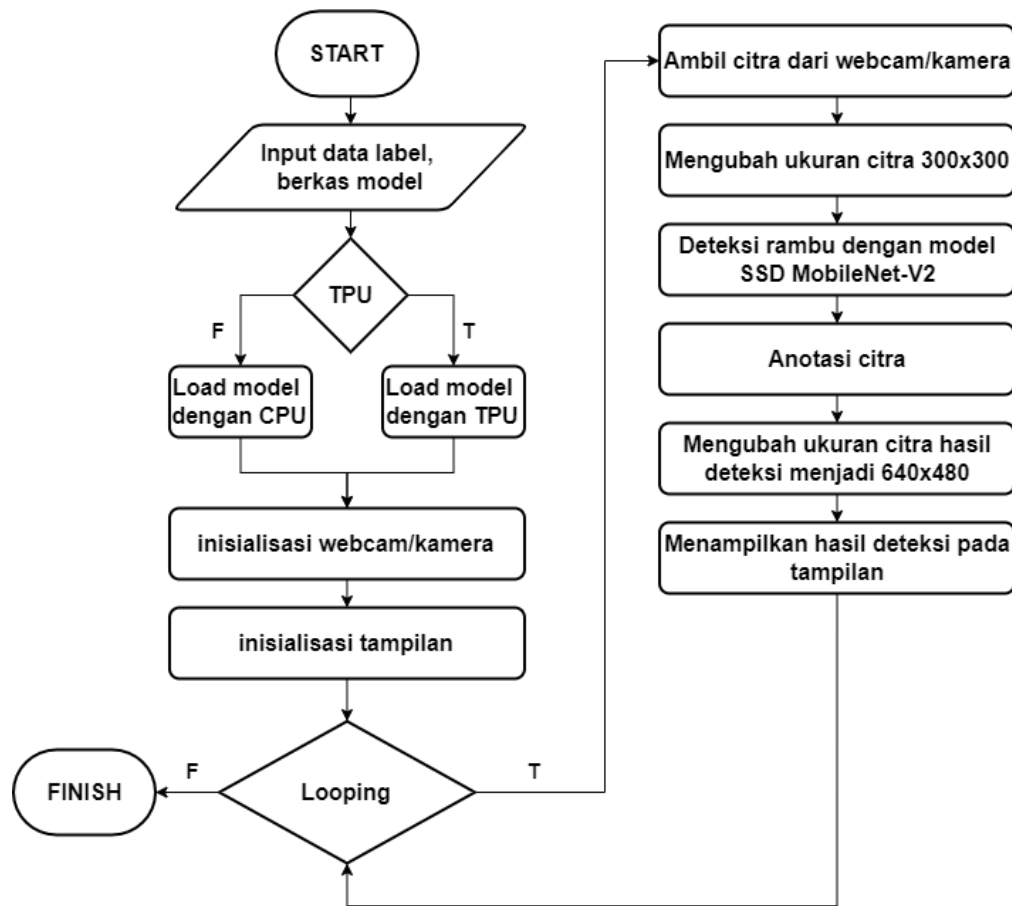


Gambar 3.19. Skema Pengujian Jupyter Notebook

Langkah pertama, masukkan data *test*, data label, berkas model yang berupa lokasi penyimpanan untuk mempermudah pemanggilan berkas. Langkah kedua, memberi masukan untuk menggunakan TPU atau CPU sebagai perangkat keras pengolah data. Langkah ketiga, memuat data *test* berupa citra dan anotasi dalam bentuk berkas excel. Langkah keempat, proses deteksi citra menggunakan model SSD MobileNet-v2. Langkah kelima, menghitung nilai IoU dan waktu komputasi. Langkah keenam, hasil deteksi disimpan kedalam berkas excel untuk proses perhitungan kinerja model. Langkah ketujuh, memuat berkas excel hasil deteksi untuk dilakukan proses perhitungan nilai akurasi, *precision*, dan *recall* kemudian menampilkan hasil perhitungan tersebut.

3.5.2. Skema Pengujian Raspberry Pi

Skema pengujian melalui Raspberry Pi menggunakan bantuan *webcam* memiliki beberapa tahapan yang perlu dilakukan, dapat dilihat pada Gambar 3.20.



Gambar 3.20. Skema Pengujian Jupyter Notebook

Langkah pertama, masukkan data label, berkas model yang berupa lokasi penyimpanan untuk mempermudah pemanggilan berkas. Langkah kedua, memberi masukan untuk menggunakan TPU atau CPU sebagai perangkat keras pengolah data. Langkah ketiga, inisialisasi *webcam* dan tampilan. Langkah keempat, mengambil citra dari *webcam* dan citra diubah menjadi ukuran 300x300. Langkah kelima, proses deteksi citra menggunakan model SSD MobileNet-V2. Langkah keenam, menganotasi citra dari masukan *webcam* dengan hasil deteksi. Langkah ketujuh, mengubah ukuran citra hasil deteksi menjadi ukuran 640x480 menyesuaikan dengan ukuran tampilan. Langkah kedelapan, menampilkan informasi deteksi seperti lokasi objek, kelas objek, nilai *confidence*, dan waktu komputasi..

3.6. Evaluasi Model

Evaluasi model dilakukan dengan matriks evaluasi terhadap hasil deteksi objek sesuai dengan data yang akan diuji. Hasil evaluasi model menjadi tolak ukur untuk mengetahui kinerja model terhadap kasus deteksi rambu lalu lintas. Evaluasi model dilakukan untuk membandingkan hasil model terhadap beberapa kasus uji *hyperparameter* yaitu *learning*

rate, nilai *batch*, dan jumlah *epoch* serta pengaruh augmentasi data. Jumlah *epoch* ditentukan melalui proses training yang menerapkan *early stopping* (penghentian awal) untuk menghindari kondisi *overfitting* yang ditinjau dari nilai *loss* saat pelatihan model. Pelatihan model dilakukan dengan batasan *epoch* sebesar 30000 ketika penerapan *early stopping* terjadi memungkinkan jumlah *epoch* berada di bawah 30000 *epoch*. Informasi mengenai kasus uji dapat dilihat pada Tabel 3.1.

Tabel 3.1. Kasus Uji *HyperParameter*

Kasus Uji	<i>Learning rate</i>	<i>Batch</i>	Tipe Data
I	0.0001	2	{Tanpa Augmentasi, Dengan Augmentasi}
II	0.0005	2	{Tanpa Augmentasi, Dengan Augmentasi}
III	0.005	2	{Tanpa Augmentasi, Dengan Augmentasi}
IV	0.0001	4	{Tanpa Augmentasi, Dengan Augmentasi}
V	0.0005	4	{Tanpa Augmentasi, Dengan Augmentasi}
VI	0.005	4	{Tanpa Augmentasi, Dengan Augmentasi}
VII	0.0001	8	{Tanpa Augmentasi, Dengan Augmentasi}
VIII	0.0005	8	{Tanpa Augmentasi, Dengan Augmentasi}
IX	0.005	8	{Tanpa Augmentasi, Dengan Augmentasi}

Pengujian model deteksi rambu lalu lintas menggunakan model SSD MobileNet-V2 memiliki beberapa kriteria evaluasi yang terdiri dari *True Positive* (TP), *False Positive* (FP), dan *False Negative* (FN). TP berarti objek terdeteksi benar sesuai dengan *ground truth*, FP berarti objek terdeteksi salah atau tidak sesuai dengan *ground truth*, FN berarti objek tidak terdeteksi. Objek yang terdeteksi ditandai dengan nilai $\text{IoU} \geq 0.5$ antara prediksi lokasi *bounding box* dan lokasi *ground truth* yang dihitung melalui persamaan 2.42. Objek yang terdeteksi benar (TP) ketika nilai IoU terpenuhi dan prediksi kelas sesuai dengan *ground truth* sedangkan prediksi kelas yang tidak sesuai dengan *ground truth* maka dianggap sebagai terdeteksi salah (FP). Kriteria lain apabila model tidak dapat mendeteksi objek maka dianggap sebagai tidak terdeteksi (FN). Setelah itu, dilakukan evaluasi dengan menghitung nilai akurasi, *precision*, *recall*, dan rata-rata waktu komputasi.

BAB IV

HASIL DAN PEMBAHASAN

Bab ini membahas mengenai implementasi dan hasil yang dilakukan dalam penyelesaian masalah deteksi rambu lalu lintas.

4.1. Implementasi

Implementasi dalam penyelesaian masalah deteksi rambu lalu lintas menggunakan model SSD MobileNet-V2 terbagi menjadi beberapa bagian yaitu spesifikasi perangkat dan implementasi model.

4.1.1. Spesifikasi Perangkat

Pra-pemrosesan data (konversi data, augmentasi data, dan pembagian data) menggunakan laptop dengan spesifikasi sebagai berikut.

1. CPU : Intel(R) i5-7200U @ 2.50GHz (2 *cores*)
2. GPU : NVIDIA GeForce 930MX 2 GB
3. RAM : 12 GB
4. OS : Windows 10
5. Storage : SSD 250 GB

Pelatihan model SSD MobileNet-V2 menggunakan *cloud services* yang bernama Google Colaboratory dengan spesifikasi sebagai berikut.

1. CPU : Intel(R) Xeon(R) @ 2.30GHz (2 *cores*)
2. GPU : NVIDIA TESLA P100 16 GB
3. RAM : 24 GB
4. OS : Ubuntu 18.04 64-bit
5. Storage : Google Drive

Pengujian model SSD MobileNet-V2 melalui Raspberry Pi versi 4 dengan spesifikasi sebagai berikut.

1. CPU : Broadcom BCM2711 @1.5GHZ (4 *cores*)
2. GPU : Videocore VI @500Mhz
3. RAM : 4 GB
4. OS : Raspbian OS Bullseye (Debian 11 / Linux)
5. Storage : Micro SD 64 GB
6. Webcam : Logitech C525 (via USB 3.0)
7. TPU : Google Coral USB Accelerator (via USB 3.0)
8. Power : 5V 3A

4.1.2. Implementasi Penelitian

Pra-pemrosesan data diimplementasikan menggunakan laptop dengan bahasa pemrograman Python yang terbagi ke beberapa proses yaitu konversi data, augmentasi data, pembagian data. Konversi data dilakukan dengan mengubah citra yang berformat *Portable Pixel Map* (PPM) menjadi *Portable Network Graphics* (PNG) menggunakan bantuan *library* OpenCV serta menyimpan anotasi *dataset* ke dalam berkas excel. Augmentasi data dilakukan dengan beberapa operasi pemrosesan citra digital menggunakan bantuan *library* *imgaug*. *Library* ini mampu melakukan pemrosesan citra digital beserta anotasi sehingga hasil perubahan citra diikuti dengan perubahan anotasi apabila terjadi perubahan posisi objek. Setelah itu, proses dilanjutkan dengan mengubah ukuran citra hasil augmentasi menjadi 300x300 beserta perubahan ukuran anotasi. Pembagian data terbagi menjadi 3 jenis yaitu *train*, *validation*, dan *test*. Data *train* dan *validation* digunakan untuk proses pelatihan data yang diunggah ke *cloud storage* yaitu Google Drive. Data *test* digunakan untuk proses pengujian model yang disimpan pada perangkat pengujian.

Pengembangan Model dilakukan dengan bantuan *library* Tensorflow yang dapat berjalan pada berbagai macam sistem operasi dan perangkat komputasi. Proses pelatihan model dilakukan bantuan Google Colaboratory yang sudah terpasang Tensorflow. Pelatihan model menggunakan SSD MobileNet-V2 diimplementasikan pada *library* Tensorflow dengan pengaturan beberapa *hyperparameter* seperti nilai *learning rate*, nilai *batch*, dan jumlah *epoch* serta pengaruh augmentasi data. Setelah pelatihan model sesuai dengan *hyperparameter* yang akan diuji, model disimpan dan dikonversi ke dalam format Tensorflow Lite (TFLite) supaya dapat berjalan di perangkat komputasi rendah yaitu Raspberry Pi.

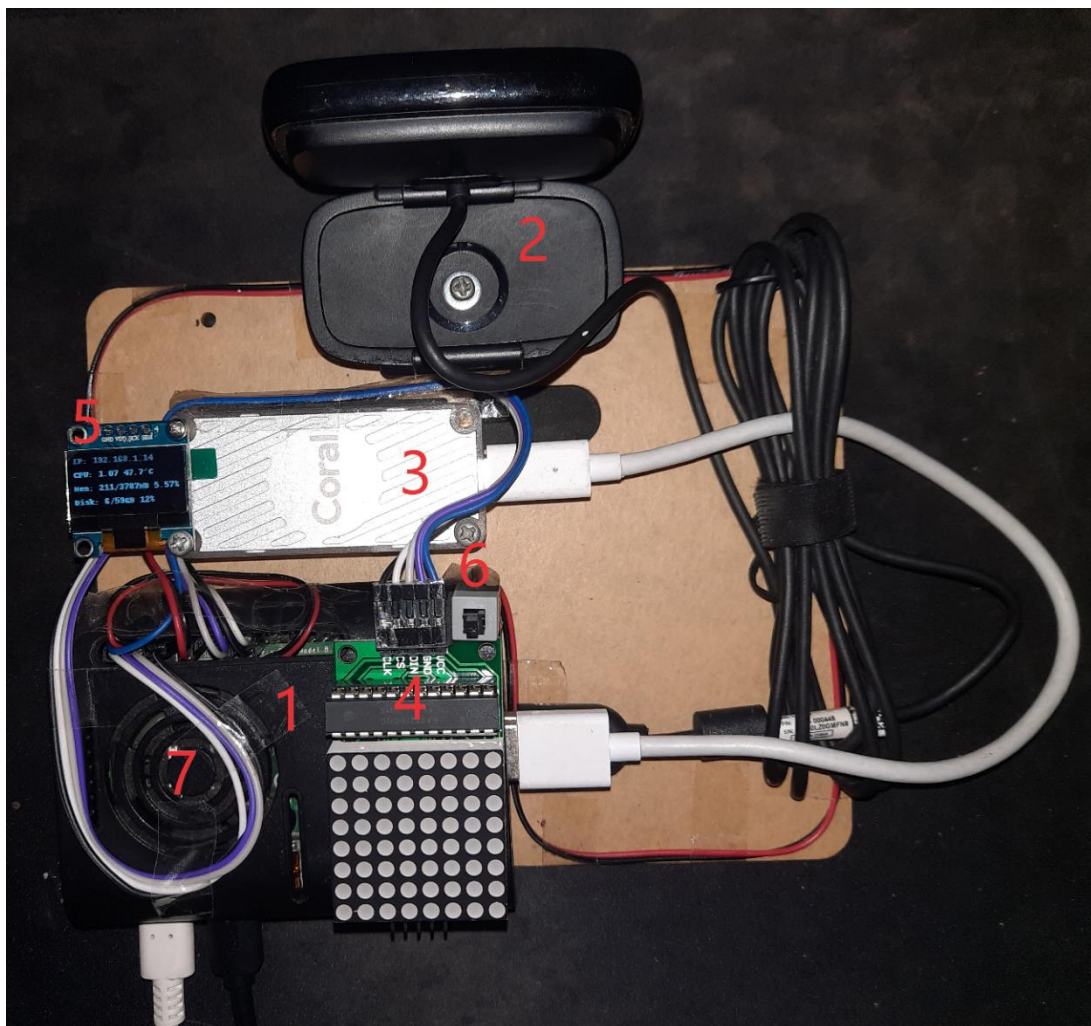
Setiap hasil konversi model digunakan untuk evaluasi model melalui Jupyter Notebook pada Raspberry Pi. Evaluasi model dilakukan untuk mengetahui kinerja model yang ditunjukkan dengan hasil perhitungan nilai akurasi, *precision*, dan waktu komputasi. Evaluasi model digunakan untuk menentukan *hyperparameter* terbaik dalam pengembangan model untuk sistem deteksi rambu lalu lintas.

4.1.3. Implementasi Perangkat Raspberry Pi

Implementasi perangkat diawali dengan persiapan komponen yang dibutuhkan yaitu Raspberry Pi 4, Micro SD, *webcam*, Google Coral USB (TPU), modul LED DOT Matrix berukuran 8x8, modul LCD OLED 0.96 inch, tombol *switch*, kipas, dan akrilik berukuran

17x14 cm. instalasi sistem operasi Raspbian OS Bullseye yang berbasis linux pada media penyimpanan Micro SD. Setelah itu, persiapkan lingkungan sistem dengan instalasi *library* tambahan seperti OpenCV, PyQt5 (pembuatan tampilan sederhana), dan Tensorflow Lite Runtime (versi ringan dari Tensorflow).

Perangkat Raspberry Pi 4 disusun di atas lempengan akrilik dengan penambahan beberapa komponen untuk melengkapi alat tersebut. Alat ini memerlukan sumber daya listrik dengan tegangan listrik sebesar 5 Volt dan arus listrik sebesar 3 Ampere melalui lubang *port* bertipe C. Susunan perangkat Raspberry Pi 4 dengan tambahan komponen dapat dilihat pada Gambar 4.1.



Gambar 4.1. Susunan Alat

Berikut ini adalah penjelasan dari masing-masing komponen penyusun alat deteksi rambu lalu lintas yang ditunjukkan pada angka:

1. Raspberry Pi 4

Perangkat Raspberry Pi 4 digunakan sebagai komputer induk dari sistem yang terdiri dari lingkungan sistem dan komponen pendukung. Program akan dieksekusi dalam terminal linux dengan memuat model melalui *library* Tensorflow Lite Runtime dan komponen pendukung dijalankan pada program tersebut.

2. Webcam Logitech C525

Webcam digunakan sebagai masukan dalam sistem dengan pengambilan video secara langsung dan diambil tiap *frame* untuk dilakukan pemrosesan deteksi rambu lalu lintas. *Webcam* ini memiliki resolusi sebesar 8 MP (1280x720p) dengan *frame rate* sebesar 30 FPS serta memiliki fitur *autofocus* supaya penangkapan gambar lebih baik. Perangkat ini dihubungkan ke Raspberry Pi melalui *port* USB 2.0 untuk transfer data dari *webcam*.

3. Google Coral USB Accelerator

Google Coral USB Accelerator atau *Tensor Processing Unit* (TPU) digunakan sebagai perangkat komputasi yang mendukung *library* Tensorflow Lite Runtime untuk membantu pekerjaan CPU. Perangkat ini melakukan komputasi dari data masukan yang diberikan oleh Raspberry Pi sesuai dengan model yang digunakan kemudian hasil pemrosesan data akan dikembalikan ke Raspberry Pi. Google Coral membantu peran CPU dalam komputasi model sehingga tidak membebani CPU serta TPU memiliki kemampuan komputasi yang lebih baik. Hal ini akan meningkatkan kinerja deteksi rambu lalu lintas dalam segi waktu komputasi. TPU dihubungkan ke Raspberry Pi melalui *port* USB 3.0 untuk transfer data dengan kecepatan yang lebih tinggi dan diiringi dengan pemrosesan data yang relatif lebih cepat.

4. Modul LED DOT Matrix 8x8

Modul LED DOT Matrix 8x8 digunakan sebagai panel untuk menampilkan hasil berupa kelas dari deteksi rambu lalu lintas. Modul ini akan berkedip menampilkan angka sesuai dengan nomor kelas dari proses deteksi objek yang diatur dalam program Python. Modul LED terhubung dengan Raspberry Pi menggunakan pin GPIO (*General Purpose Input Output*) yang dapat dilihat pada Tabel 4.1.

Tabel 4.1. Susunan GPIO Modul LED DOT Matrix

Nomor Pin	Nama Pin	Warna Kabel	Kegunaan
2	5V Power	Biru	Memberikan sumber listrik dengan tegangan sebesar 5V untuk menjalankan modul

Nomor Pin	Nama Pin	Warna Kabel	Kegunaan
19	GPIO 10 (SPI0 MOSI)	Abu-abu	Memberikan data serial kepada modul LED DOT Matrix dari Raspberry Pi
20	Ground	Ungu	Menyeimbangkan tegangan masukan dari modul LED DOT Matrix
23	GPIO 11 (SPI0 SCLK)	Hitam	Clock yang digunakan untuk pengiriman data
24	GPIO 8 (SPI0 CE)	Putih	Menghubungkan Raspberry Pi dengan <i>chip</i> dari modul LED DOT Matrix untuk proses penampilan data

5. Modul LCD OLED 0.96 Inch

Modul LCD OLED berukuran 0.96 Inch (128x64 piksel) digunakan sebagai layar kecil yang digunakan untuk menampilkan informasi perangkat Raspberry Pi seperti IP *address*, penggunaan memori, penggunaan RAM, beban CPU, dan suhu perangkat. Informasi ini diatur dalam program Python. Modul OLED terhubung dengan Raspberry Pi menggunakan pin GPIO (*General Purpose Input Output*) yang dapat dilihat pada Tabel 4.2.

Tabel 4.2. Susunan GPIO Modul OLED 0.96 Inch

Nomor Pin	Nama Pin	Warna Kabel	Kegunaan
1	3.3V Power	Biru	Memberikan sumber listrik dengan tegangan sebesar 3.3V untuk menjalankan modul
9	Ground	Hitam	Menyeimbangkan tegangan masukan dari modul LCD OLED
3	GPIO 2 (I2C SDA)	Ungu	Memberikan data serial kepada modul LCD OLED dari Raspberry Pi
5	GPIO 3 (I2C SCL / SCK)	Hitam	Clock yang digunakan untuk pengiriman data

6. Tombol *Switch*

Tombol *switch* digunakan sebagai tombol perintah untuk menyalakan ulang atau mematikan Raspberry Pi. Perintah menyalakan ulang dilakukan dengan menekan tombol

switch selama 1-3 detik dan perintah mematikan dilakukan dengan menekan tombol *switch* selama lebih dari 3 detik, kedua perintah tersebut diatur dalam program Python. Tombol *switch* terhubung dengan Raspberry Pi menggunakan pin GPIO (*General Purpose Input Output*) yang dapat dilihat pada Tabel 4.3.

Tabel 4.3. Susunan GPIO Tombol Switch

Nomor Pin	Nama Pin	Warna Kabel	Kegunaan
13	GPIO 27	Merah	Memberikan sumber listrik dengan tegangan sebesar 3.3V untuk menjalankan modul
14	Ground	Hitam	Menyeimbangkan tegangan masukan dari tombol <i>switch</i>

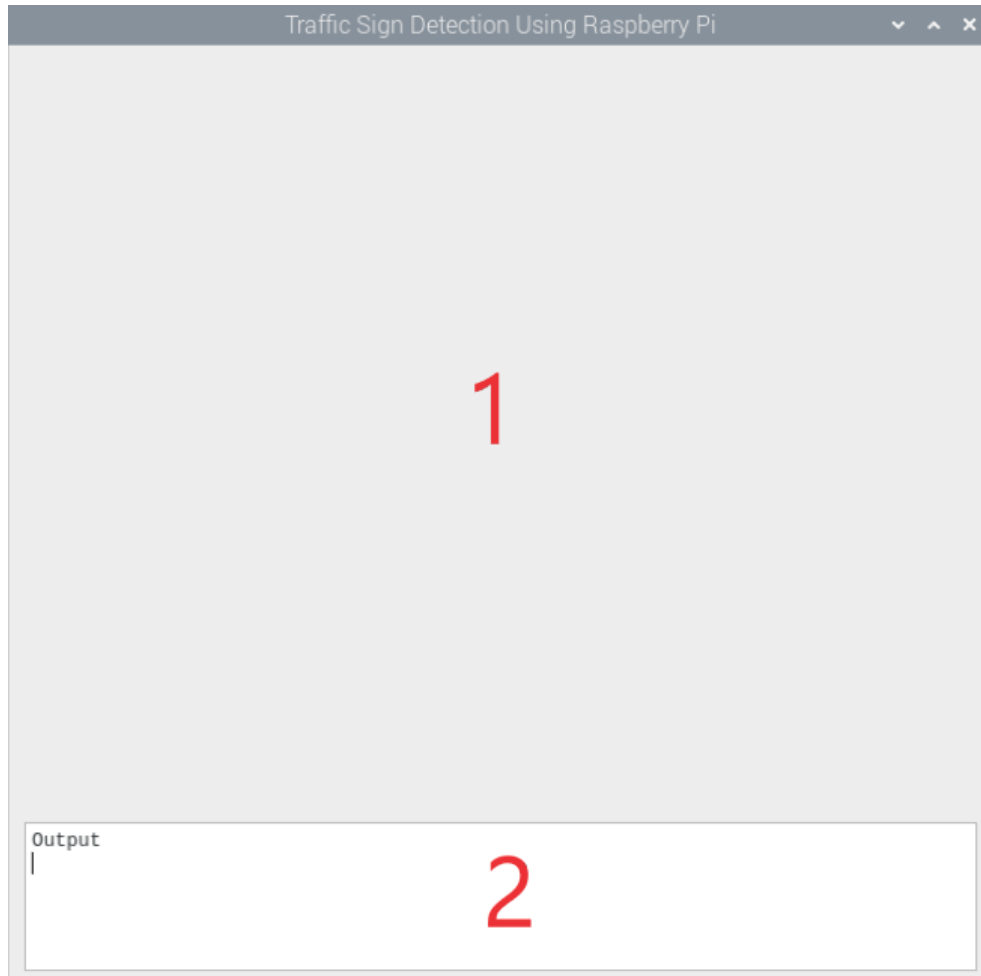
7. Kipas

Kipas digunakan untuk menjaga suhu perangkat supaya tidak terjadi *overheat* karena panas yang berlebihan dapat mempengaruhi kinerja dari Raspberry Pi. Kipas terhubung dengan Raspberry Pi menggunakan pin GPIO (*General Purpose Input Output*) yang dapat dilihat pada Tabel 4.4.

Tabel 4.4. Susunan GPIO Kipas

Nomor Pin	Nama Pin	Warna Kabel	Kegunaan
4	5V Power	Merah	Memberikan sumber listrik dengan tegangan sebesar 5V untuk menjalankan modul
6	Ground	Hitam	Menyeimbangkan tegangan masukan dari kipas

Pengujian deteksi rambu lalu lintas melalui Raspberry Pi menggunakan tampilan sederhana yang dibuat melalui *library* PyQt5. Tampilan ini hanya memiliki satu buah antarmuka yang tersusun atas *frame* untuk menampilkan hasil deteksi secara langsung melalui masukan dari *webcam* dan kotak dialog untuk menampilkan informasi hasil deteksi. Angka 1 menunjukkan area hasil deteksi melalui masukan *webcam* dan angka 2 menunjukkan kotak dialog untuk menampilkan informasi hasil deteksi. Gambaran mengenai tampilan sederhana yang diimplementasikan menggunakan *library* PyQt5 dapat dilihat pada Gambar 4.2.



Gambar 4.2. Implementasi Tampilan Sistem

Program dijalankan melalui terminal Linux pada Raspberry 4 dengan beberapa masukan yang telah didefinisikan di dalam program. Masukan yang diberikan pada program ini adalah berkas model, berkas label, penggunaan TPU, dan perangkat *webcam*. Program menampilkan hasil deteksi rambu lalu lintas menggunakan Model SSD MobileNet-V2 secara *realtime* dari tangkapan gambar *webcam* pada Nomor 1 dan menampilkan informasi terkait deteksi rambu lalu lintas pada nomor 2.

4.2. Pengujian Model Melalui Jupyter Notebook Perangkat Raspberry Pi

Pengujian model melalui Jupyter Notebook menggunakan CPU yang dimiliki oleh Raspberry Pi. Proses pengujian dilakukan dengan memasukkan folder yang berisi data *test* (citra dan anotasi) pada program untuk dievaluasi sesuai kasus uji. Hal ini dilakukan untuk mengetahui hasil evaluasi dari masing-masing kasus uji berdasarkan nilai *learning rate*, nilai *batch*, dan jumlah *epoch*. Penentuan *hyperparameter* dipilih berdasarkan hasil evaluasi terbaik dalam penelitian ini. Berikut ini adalah pengujian yang dilakukan tiap kasus uji.

4.2.1. Kasus Uji I

Kasus uji I menggunakan beberapa *hyperparameter* yang dimasukkan pada proses pelatihan model SSD MobileNet-V2 yaitu nilai *learning rate* 0.0001, nilai *batch* 2, dan jumlah *epoch* maksimal 30000. Hasil pengujian kasus uji I dapat dilihat pada Tabel 4.5.

Tabel 4.5. Hasil Pengujian Kasus Uji I Menggunakan CPU

Tipe Data	<i>Epoch</i>	TP	FP	FN	TN	Akurasi	Presisi	<i>Recall</i>	Waktu (s)
Tanpa Augmentasi	17854	67	13	33	0	0.59	0.84	0.67	0.103
Augmentasi	28282	81	5	27	0	0.72	0.94	0.75	0.1

Hasil pengujian menyajikan perhitungan nilai akurasi, presisi, *recall*, dan rata-rata waktu komputasi dari pelatihan model menggunakan *hyperparameter* nilai *learning rate* 0.0001, nilai *batch* 2. Pelatihan model tanpa augmentasi berhenti pada jumlah *epoch* sebesar 17854, menghasilkan akurasi 59%, presisi 84%, *recall* 67%, dan rata-rata waktu komputasi 0.103s. Pelatihan model dengan augmentasi berhenti pada jumlah *epoch* sebesar 28282, menghasilkan akurasi 72%, presisi 94%, *recall* 75%, dan rata-rata waktu komputasi 0.1s. Pada kasus uji I, proses augmentasi data dapat mempengaruhi peningkatan akurasi model pada *hyperparameter* nilai *learning rate* 0.0001 dan nilai *batch* 2 dengan peningkatan akurasi sebesar 13%.

4.2.2. Kasus Uji II

Kasus uji II menggunakan beberapa parameter yang dimasukkan pada proses pelatihan model SSD MobileNet-V2 yaitu nilai *learning rate* 0.0005, nilai *batch* 2, dan jumlah *epoch* maksimal 30000. Hasil pengujian kasus uji I dapat dilihat pada Tabel 4.6.

Tabel 4.6. Hasil Pengujian Kasus Uji II Menggunakan CPU

Tipe Data	<i>Epoch</i>	TP	FP	FN	TN	Akurasi	Presisi	<i>Recall</i>	Waktu (s)
Tanpa Augmentasi	23014	84	7	22	0	0.74	0.92	0.79	0.104
Augmentasi	26408	77	3	33	0	0.68	0.96	0.7	0.105

Hasil pengujian menyajikan perhitungan nilai akurasi, presisi, *recall*, dan rata-rata waktu komputasi dari pelatihan model menggunakan *hyperparameter* nilai *learning rate* 0.0005, nilai *batch* 2. Pelatihan model tanpa augmentasi berhenti pada jumlah *epoch* sebesar 23014, menghasilkan akurasi 74%, presisi 92%, *recall* 79%, dan rata-rata waktu komputasi

0.104s. Pelatihan model dengan augmentasi berhenti pada jumlah *epoch* sebesar 26408, menghasilkan akurasi 68%, presisi 96%, *recall* 70%, dan rata-rata waktu komputasi 0.105s. Pada kasus uji II, proses augmentasi data tidak mempengaruhi peningkatan akurasi model pada *hyperparameter* nilai *learning rate* 0.0005 dan nilai *batch* 2.

4.2.3. Kasus Uji III

Kasus uji III menggunakan beberapa *hyperparameter* yang dimasukkan pada proses pelatihan model SSD MobileNet-V2 yaitu nilai *learning rate* 0.005, nilai *batch* 2, dan jumlah *epoch* maksimal 30000. Hasil pengujian kasus uji III dapat dilihat pada Tabel 4.7.

Tabel 4.7. Hasil Pengujian Kasus Uji III Menggunakan CPU

Tipe Data	<i>Epoch</i>	TP	FP	FN	TN	Akurasi	Presisi	<i>Recall</i>	Waktu (s)
Tanpa Augmentasi	15202	65	8	40	0	0.58	0.89	0.62	0.108
Augmentasi	23066	68	10	35	0	0.6	0.87	0.66	0.148

Hasil pengujian menyajikan perhitungan nilai akurasi, presisi, *recall*, dan rata-rata waktu komputasi dari pelatihan model menggunakan *hyperparameter* nilai *learning rate* 0.005, nilai *batch* 2. Pelatihan model tanpa augmentasi berhenti pada jumlah *epoch* sebesar 15202, menghasilkan akurasi 58%, presisi 89%, *recall* 62%, dan rata-rata waktu komputasi 0.108s. Pelatihan model dengan augmentasi berhenti pada jumlah *epoch* sebesar 23066, menghasilkan akurasi 60%, presisi 87%, *recall* 66%, dan rata-rata waktu komputasi 0.148s. Pada kasus uji III, proses augmentasi data dapat mempengaruhi peningkatan akurasi model pada *hyperparameter* nilai *learning rate* 0.005 dan nilai *batch* 2 dengan peningkatan akurasi sebesar 2%.

4.2.4. Kasus Uji IV

Kasus uji IV menggunakan beberapa *hyperparameter* yang dimasukkan pada proses pelatihan model SSD MobileNet-V2 yaitu nilai *learning rate* 0.0001, nilai *batch* 4, dan jumlah *epoch* maksimal 30000. Hasil pengujian kasus uji IV dapat dilihat pada Tabel 4.8.

Tabel 4.8. Hasil Pengujian Kasus Uji IV Menggunakan CPU

Tipe Data	<i>Epoch</i>	TP	FP	FN	TN	Akurasi	Presisi	<i>Recall</i>	Waktu (s)
Tanpa Augmentasi	10902	62	13	38	0	0.55	0.83	0.62	0.11
Augmentasi	21288	77	11	25	0	0.68	0.88	0.75	0.096

Hasil pengujian menyajikan perhitungan nilai akurasi, presisi, *recall*, dan rata-rata waktu komputasi dari pelatihan model menggunakan *hyperparameter* nilai *learning rate* 0.0001, nilai *batch* 4. Pelatihan model tanpa augmentasi berhenti pada jumlah *epoch* sebesar 10902, menghasilkan akurasi 55%, presisi 83%, *recall* 62%, dan rata-rata waktu komputasi 0.11s. Pelatihan model dengan augmentasi berhenti pada jumlah *epoch* sebesar 21288, menghasilkan akurasi 68%, presisi 88%, *recall* 75%, dan rata-rata waktu komputasi 0.096s. Pada kasus uji IV, proses augmentasi data dapat mempengaruhi peningkatan akurasi model pada *hyperparameter* nilai *learning rate* 0.0001 dan nilai *batch* 4 dengan peningkatan akurasi sebesar 13%.

4.2.5. Kasus Uji V

Kasus uji V menggunakan beberapa *hyperparameter* yang dimasukkan pada proses pelatihan model SSD MobileNet-V2 yaitu nilai *learning rate* 0.0005, nilai *batch* 4, dan jumlah *epoch* maksimal 30000. Hasil pengujian kasus uji V dapat dilihat pada Tabel 4.9.

Tabel 4.9. Hasil Pengujian Kasus Uji V Menggunakan CPU

Tipe Data	<i>Epoch</i>	TP	FP	FN	TN	Akurasi	Presisi	<i>Recall</i>	Waktu (s)
Tanpa Augmentasi	16174	84	3	26	0	0.74	0.97	0.76	0.105
Augmentasi	11939	79	6	28	0	0.7	0.93	0.74	0.104

Hasil pengujian menyajikan perhitungan nilai akurasi, presisi, *recall*, dan rata-rata waktu komputasi dari pelatihan model menggunakan *hyperparameter* nilai *learning rate* 0.0005, nilai *batch* 4. Pelatihan model tanpa augmentasi berhenti pada jumlah *epoch* sebesar 16174, menghasilkan akurasi 74%, presisi 97%, *recall* 76%, dan rata-rata waktu komputasi 0.105s. Pelatihan model dengan augmentasi berhenti pada jumlah *epoch* sebesar 11939, menghasilkan akurasi 70%, presisi 93%, *recall* 74%, dan rata-rata waktu komputasi 0.104s. Pada kasus uji V, proses augmentasi data tidak mempengaruhi peningkatan akurasi model pada *hyperparameter* nilai *learning rate* 0.0005 dan nilai *batch* 4.

4.2.6. Kasus Uji VI

Kasus uji VI menggunakan beberapa *hyperparameter* yang dimasukkan pada proses pelatihan model SSD MobileNet-V2 yaitu nilai *learning rate* 0.005, nilai *batch* 4, dan jumlah *epoch* maksimal 30000. Hasil pengujian kasus uji VI dapat dilihat pada Tabel 4.10.

Tabel 4.10. Hasil Pengujian Kasus Uji VI Menggunakan CPU

Tipe Data	<i>Epoch</i>	TP	FP	FN	TN	Akurasi	Presisi	<i>Recall</i>	Waktu (s)
Tanpa Augmentasi	15249	79	3	31	0	0.7	0.96	0.72	0.105
Augmentasi	18997	81	2	30	0	0.72	0.98	0.73	0.102

Hasil pengujian menyajikan perhitungan nilai akurasi, presisi, *recall*, dan rata-rata waktu komputasi dari pelatihan model menggunakan *hyperparameter* nilai *learning rate* 0.005, nilai *batch* 4. Pelatihan model tanpa augmentasi berhenti pada jumlah *epoch* sebesar 15249, menghasilkan akurasi 70%, presisi 96%, *recall* 72%, dan rata-rata waktu komputasi 0.105s. Pelatihan model dengan augmentasi berhenti pada jumlah *epoch* sebesar 18997, menghasilkan akurasi 72%, presisi 98%, *recall* 73%, dan rata-rata waktu komputasi 0.102s. Pada kasus uji VI, proses augmentasi data dapat mempengaruhi peningkatan akurasi model pada *hyperparameter* nilai *learning rate* 0.005 dan nilai *batch* 4 dengan peningkatan akurasi sebesar 2%.

4.2.7. Kasus Uji VII

Kasus uji VII menggunakan beberapa *hyperparameter* yang dimasukkan pada proses pelatihan model SSD MobileNet-V2 yaitu nilai *learning rate* 0.0001, nilai *batch* 8, dan jumlah *epoch* maksimal 30000. Hasil pengujian kasus uji VII dapat dilihat pada Tabel 4.11.

Tabel 4.11. Hasil Pengujian Kasus Uji VII Menggunakan CPU

Tipe Data	<i>Epoch</i>	TP	FP	FN	TN	Akurasi	Presisi	<i>Recall</i>	Waktu (s)
Tanpa Augmentasi	26220	70	11	32	0	0.62	0.86	0.69	0.113
Augmentasi	28478	84	6	23	0	0.74	0.93	0.79	0.1

Hasil pengujian menyajikan perhitungan nilai akurasi, presisi, *recall*, dan rata-rata waktu komputasi dari pelatihan model menggunakan *hyperparameter* nilai *learning rate* 0.0001, nilai *batch* 8. Pelatihan model tanpa augmentasi berhenti pada jumlah *epoch* sebesar 26220, menghasilkan akurasi 62%, presisi 86%, *recall* 69%, dan rata-rata waktu komputasi 0.113s. Pelatihan model dengan augmentasi berhenti pada jumlah *epoch* sebesar 28478, menghasilkan akurasi 74%, presisi 93%, *recall* 79%, dan rata-rata waktu komputasi 0.1s. Pada kasus uji VII, proses augmentasi data dapat mempengaruhi peningkatan akurasi model

pada *hyperparameter* nilai *learning rate* 0.0001 dan nilai *batch* 8 dengan peningkatan akurasi sebesar 12%.

4.2.8. Kasus Uji VIII

Kasus uji VIII menggunakan beberapa *hyperparameter* yang dimasukkan pada proses pelatihan model SSD MobileNet-V2 yaitu nilai *learning rate* 0.0005, nilai *batch* 8, dan jumlah *epoch* maksimal 30000. Hasil pengujian kasus uji VIII dapat dilihat pada Tabel 4.12.

Tabel 4.12. Hasil Pengujian Kasus Uji VIII Menggunakan CPU

Tipe Data	Epoch	TP	FP	FN	TN	Akurasi	Presisi	Recall	Waktu (s)
Tanpa Augmentasi	25000	81	7	25	0	0.72	0.92	0.76	0.106
Augmentasi	26087	91	2	20	0	0.81	0.98	0.82	0.103

Hasil pengujian menyajikan perhitungan nilai akurasi, presisi, *recall*, dan rata-rata waktu komputasi dari pelatihan model menggunakan *hyperparameter* nilai *learning rate* 0.0005, nilai *batch* 8. Pelatihan model tanpa augmentasi berhenti pada jumlah *epoch* sebesar 25000, menghasilkan akurasi 72%, presisi 92%, *recall* 76%, dan rata-rata waktu komputasi 0.106s. Pelatihan model dengan augmentasi berhenti pada jumlah *epoch* sebesar 26087, menghasilkan akurasi 81%, presisi 98%, *recall* 82%, dan rata-rata waktu komputasi 0.103s. Pada kasus uji VIII, proses augmentasi data dapat mempengaruhi peningkatan akurasi model pada *hyperparameter* nilai *learning rate* 0.0005 dan nilai *batch* 8 dengan peningkatan akurasi sebesar 9%.

4.2.9. Kasus Uji IX

Kasus uji IX menggunakan beberapa parameter yang dimasukkan pada proses pelatihan model SSD MobileNet-V2 yaitu nilai *learning rate* 0.005, nilai *batch* 8, dan jumlah *epoch* maksimal 30000. Hasil pengujian kasus uji IX dapat dilihat pada Tabel 4.13.

Tabel 4.13. Hasil Pengujian Kasus Uji IX Menggunakan CPU

Tipe Data	Epoch	TP	FP	FN	TN	Akurasi	Presisi	Recall	Waktu (s)
Tanpa Augmentasi	15617	80	7	26	0	0.71	0.92	0.75	0.156
Augmentasi	23638	90	3	20	0	0.8	0.97	0.82	0.122

Hasil pengujian menyajikan perhitungan nilai akurasi, presisi, *recall*, dan rata-rata waktu komputasi dari pelatihan model menggunakan *hyperparameter* nilai *learning rate*

0.005, nilai *batch* 8. Pelatihan model tanpa augmentasi berhenti pada jumlah *epoch* sebesar 15617, menghasilkan akurasi 71%, presisi 92%, *recall* 75%, dan rata-rata waktu komputasi 0.156s. Pelatihan model dengan augmentasi berhenti pada jumlah *epoch* sebesar 23638, menghasilkan akurasi 80%, presisi 97%, *recall* 80%, dan rata-rata waktu komputasi 0.122s. Pada kasus uji IX, proses augmentasi data dapat mempengaruhi peningkatan akurasi model pada *hyperparameter* nilai *learning rate* 0.005 dan nilai *batch* 8 dengan peningkatan akurasi sebesar 9%.

4.2.10. Penentuan *Hyperparameter* Terbaik

Penentuan *hyperparameter* terbaik dilakukan dengan membandingkan hasil evaluasi tiap kasus uji. Hasil evaluasi terbaik ditinjau dari nilai akurasi tertinggi pada pengujian kasus uji dengan beberapa *hyperparameter* masukan. Nilai *learning rate* yang digunakan yaitu 0.0001, 0.0005, dan 0.005. Nilai *batch* yang digunakan yaitu 2, 4, dan 8. Jumlah *epoch* maksimal yang digunakan yaitu 30000 serta penerapan augmentasi data. Evaluasi dilakukan mengetahui kinerja model terhadap perangkat Raspberry Pi 4.

Hasil evaluasi model SSD MobileNet-V2 yang memiliki nilai akurasi terbaik ditunjukkan oleh Kasus Uji VIII dengan penambahan augmentasi serta *hyperparameter* nilai *learning rate* 0.0005, nilai *batch* 8, dan jumlah *epoch* 26087. *Hyperparameter* ini menghasilkan akurasi 81 %, presisi 98 %, *recall* 82 %, dan rata-rata waktu komputasi 0.103s. *Hyperparameter* ini digunakan untuk pengujian menggunakan perangkat komputasi Google Coral USB (TPU) yang terhubung dengan Raspberry Pi 4 melalui *port* USB 3.0. Pengujian ini dilakukan untuk mengetahui kinerja model terhadap kecepatan waktu komputasi yang dihasilkan oleh pemrosesan deteksi rambu lalu lintas melalui perangkat TPU. Hasil pengujian parameter terpilih menggunakan CPU dan TPU dapat dilihat pada Tabel 14.

Tabel 4.14. Perbandingan Pengujian *Hyperparameter* Terbaik Menggunakan CPU dan TPU

Komputasi	TP	FP	FN	TN	Akurasi	Presisi	<i>Recall</i>	Waktu (s)
CPU	91	2	20	0	0.81	0.98	0.82	0.103
TPU	91	2	20	0	0.81	0.98	0.82	0.012

Hasil pengujian menggunakan TPU memiliki keunggulan dari segi waktu komputasi dibandingkan dengan CPU. Pada CPU memiliki rata-rata waktu komputasi sebesar 0.103 kemudian dipercepat menggunakan TPU menjadi 0.012 s. Rata-rata waktu komputasi dipercepat sebanyak 8 kali lipat menggunakan komputasi TPU dengan akurasi sebesar 81%.

4.3. Pengujian Alat Deteksi Rambu Lalu Lintas Secara *Realtime*

Pada penelitian ini dilakukan penambahan citra yang digunakan untuk pengujian model secara *realtime* melalui masukan *webcam*. Alasan penambahan citra dikarenakan ukuran objek pada *dataset* GTSDDB terlalu kecil sehingga sulit tertangkap oleh *webcam*. Citra tambahan berisi potongan rambu lalu lintas yang merepresentasikan *dataset* GTSDDB. Citra ini dapat diunduh pada *website* <https://www.bast.de/DE/Verkehrstechnik/Fachthemen/v1-verkehrszeichen/vz-download.html>. Citra tambahan berukuran 712x712 piksel dan berjumlah 43 citra dengan pembagian 4 kelas.

Pengujian alat deteksi rambu lalu lintas secara *realtime* menggunakan citra yang dicetak dengan ukuran objek sebesar 7.5x7.5cm. Pengujian sederhana dilakukan di ruangan tertutup dengan latarbelakang dinding berwarna polos. Proses pengujian rambu lalu lintas dilakukan secara tunggal maupun gabungan dari beberapa rambu lalu lintas. Pengujian alat menggunakan model dengan *hyperparameter* terbaik yang telah dievaluasi pada proses pengujian model melalui Jupyter Notebook. Parameter terbaik ditunjukkan dengan nilai *learning rate* 0.0005, nilai *batch* 8, dan jumlah *epoch* 26087. Pengujian ini dilakukan dengan meletakkan alat deteksi rambu lalu lintas dengan cetakan rambu lalu lintas sejauh 75cm dari penepatan alat. Skema ini digunakan untuk mengetahui kinerja model dengan *hyperparameter* terbaik terhadap deteksi secara *realtime* baik pengujian CPU maupun TPU. Hasil pengujian secara *realtime* menggunakan CPU dan TPU dapat dilihat pada Tabel 4.15

Tabel 4.15. Hasil Pengujian Secara *Realtime* Menggunakan CPU dan TPU

Skenario	CPU					TPU				
	TP	FP	FN	TN	Waktu (s)	TP	FP	FN	TN	Waktu (s)
1	1	-	-	-	0.115	1	-	-	-	0.017
2	1	-	-	-	0.132	1	-	-	-	0.016
3	1	-	-	-	0.099	1	-	-	-	0.019
4	1	-	-	-	0.102	1	-	-	-	0.017
11	2	-	-	-	0.125	2	-	-	-	0.016
22	2	-	-	-	0.111	2	-	-	-	0.019
33	2	-	-	-	0.103	2	-	-	-	0.019
44	1	-	-	-	0.115	1	-	-	-	0.024
12	2	-	-	-	0.099	2	-	-	-	0.017
13	2	-	-	-	0.102	2	-	-	-	0.016
14	1	-	1	-	0.099	1	-	1	-	0.02
23	2	-	-	-	0.099	2	-	-	-	0.018

Skenario	CPU					TPU				
	TP	FP	FN	TN	Waktu (s)	TP	FP	FN	TN	Waktu (s)
24	2	-	-	-	0.106	2	-	-	-	0.027
34	2	-	-	-	0.111	2	-	-	-	0.017
111	3	-	-	-	0.144	3	-	-	-	0.021
222	3	-	-	-	0.092	3	-	-	-	0.016
333	3	-	-	-	0.095	3	-	-	-	0.018
444	2	-	1	-	0.099	2	-	1	-	0.02
123	3	-	-	-	0.093	3	-	-	-	0.017
124	2	-	1	-	0.098	2	-	1	-	0.018
234	3	-	-	-	0.099	3	-	-	-	0.018
341	2	-	1	-	0.108	2	-	1	-	0.017
1111	4	-	-	-	0.117	4	-	-	-	0.019
2222	3	-	1	-	0.114	3	-	1	-	0.019
3333	4	-	-	-	0.108	4	-	-	-	0.016
4444	3	-	1	-	0.098	3	-	1	-	0.017
1234	3	-	1	-	0.113	3	-	1	-	0.018
Total	60	0	7	0	2.896	60	0	7	0	0.496
Akurasi	0.896					0.896				
Presisi	1.00					1.00				
<i>Recall</i>	0.896					0.896				
Waktu (s)	0.107					0.018				

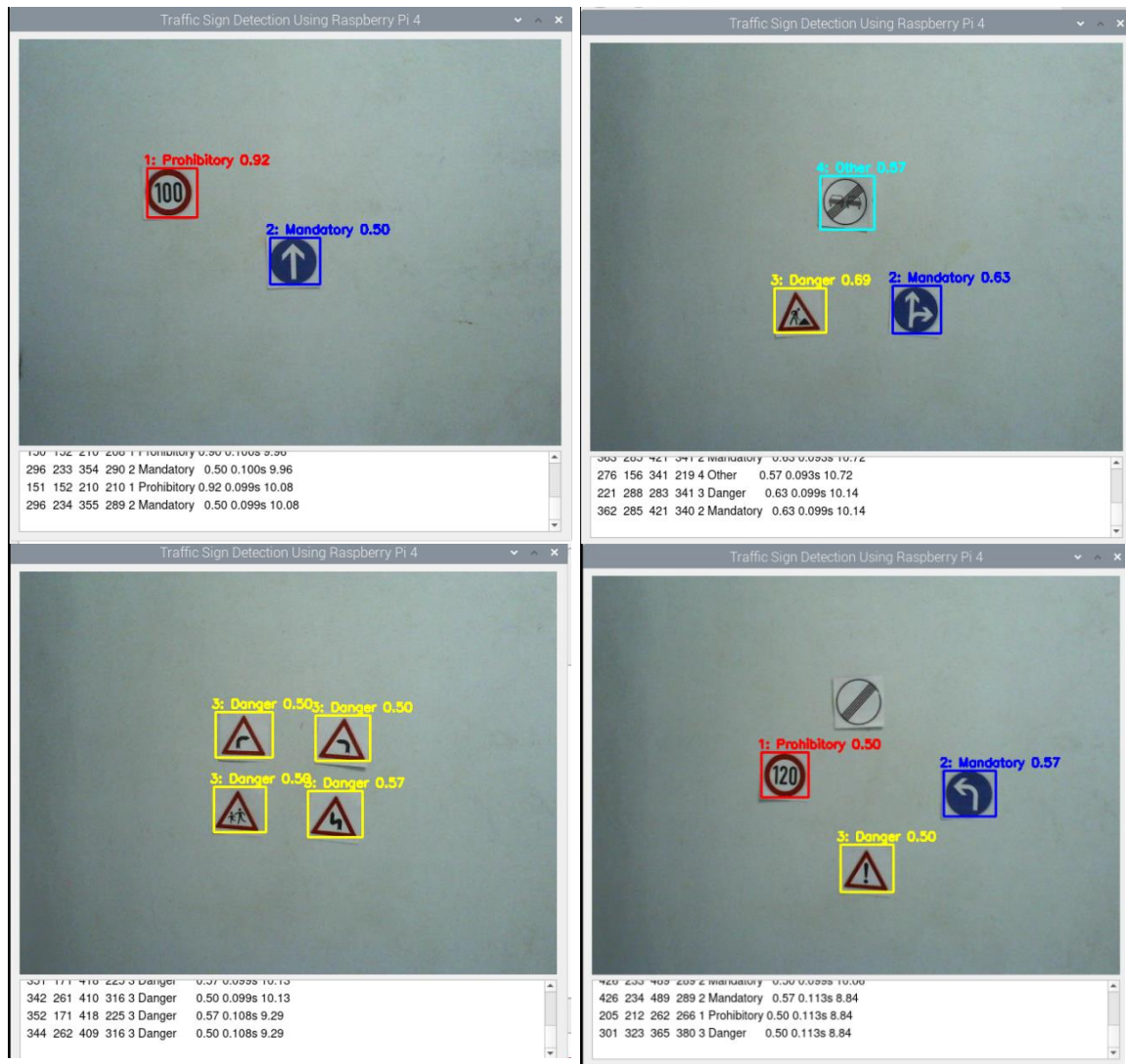
Pengujian rambu lalu lintas dilakukan secara tunggal maupun gabungan dengan kombinasi terhadap 4 buah kelas yang ditunjukkan pada kolom skenario. Jumlah digit menunjukkan banyaknya rambu digunakan saat pengujian dan angka menunjukkan kelas yang digunakan saat pengujian. Kedua pengujian melalui CPU dan TPU didapatkan hasil evaluasi yang sama dengan nilai akurasi 89.6%, presisi 100 %, dan *recall* 89.6%. Faktor pembeda dari kedua pengujian terletak pada rata-rata waktu komputasi, CPU memiliki rata-rata waktu komputasi 0.107s dan TPU memiliki rata-rata waktu komputasi 0.018 TPU membantu proses komputasi model SSD MobileNet-V2 sehingga kecepatan waktu komputasi menjadi 6 kali lipat lebih cepat dibandingkan dengan komputasi menggunakan CPU dengan akurasi yang tetap. Gambar di bawah ini adalah beberapa hasil pengujian alat deteksi rambu lalu lintas secara *realtime* melalui input dari *webcam*.



Gambar 4.3. Hasil Deteksi Rambu Lalu Lintas Secara Tunggal

Hasil deteksi rambu lalu lintas yang dilakukan secara tunggal mampu mendeteksi 4 buah kelas rambu yang berbeda sehingga alat dapat berjalan dengan baik pada kondisi satu rambu di hadapan *webcam*. Skenario ini berlaku pada komputasi menggunakan CPU maupun TPU.

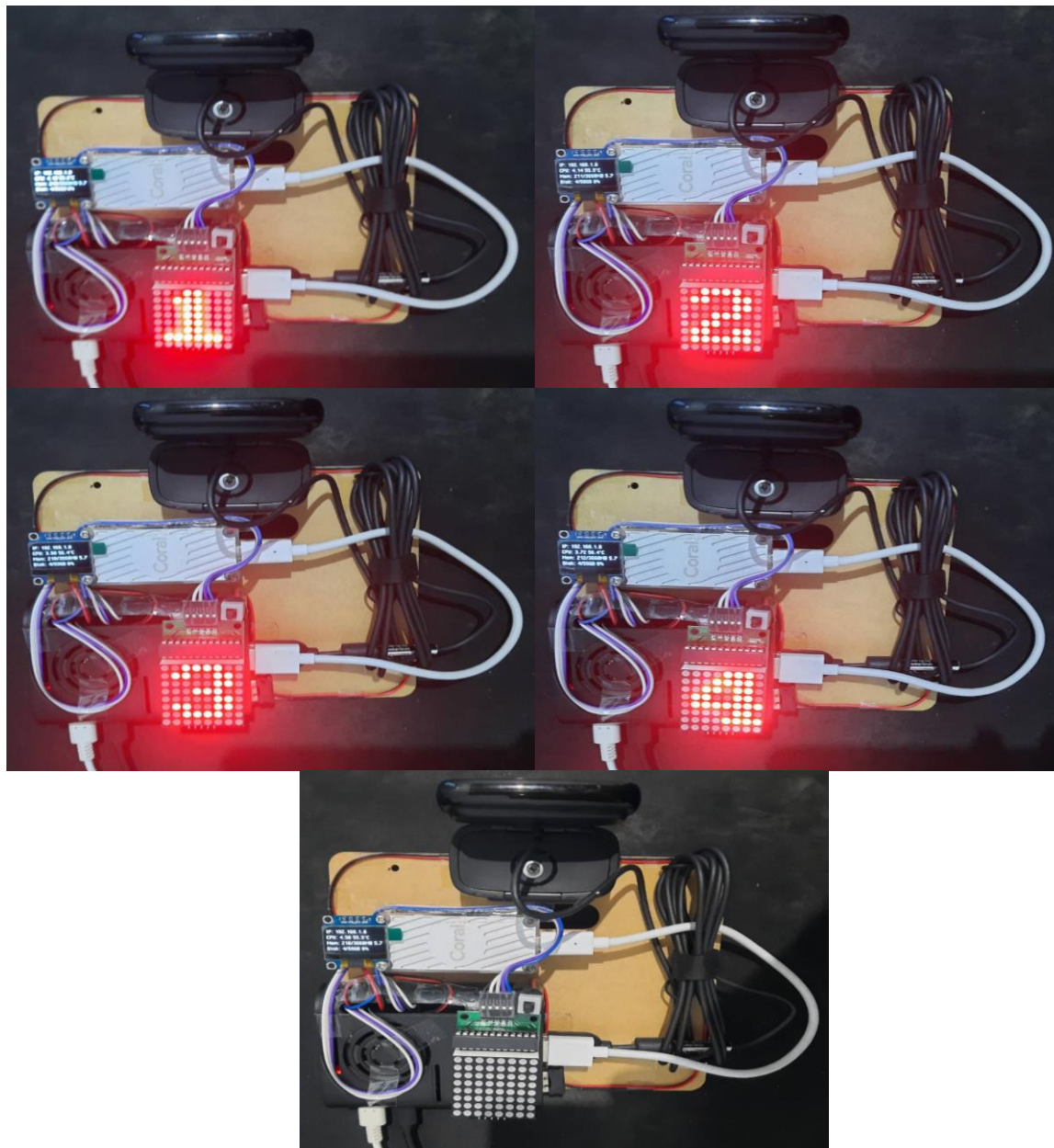
Skenario lain dilakukan dengan pengujian rambu lalu lintas secara gabungan dengan kombinasi beberapa rambu lalu lintas sesuai dengan Tabel 4.15. Gambar di bawah ini merupakan hasil pengujian rambu lalu lintas secara gabungan.



Gambar 4.4. Hasil Deteksi Rambu Lalu Lintas Secara Gabungan

Pada pengujian dengan kondisi rambu lalu lintas secara gabungan, didapatkan hasil yang beragam. Pada saat kondisi ada 2 atau 3 rambu lalu lintas alat dapat berjalan dengan baik dan mampu mendeteksi rambu lalu lintas yang memiliki kelas kelas berbeda. Permasalahan muncul ketika alat berada pada kondisi 4 rambu lalu lintas. Alat berjalan baik ketika pengujian 4 buah rambu lalu lintas dengan kelas yang sama, tetapi tidak berjalan baik ketika 4 buah rambu lalu lintas yang berbeda. Hal ini disebabkan oleh beberapa faktor yang mempengaruhi. Pertama, kondisi pengujian dilakukan pada ruangan tertutup sehingga sumber cahaya menjadi salah satu faktor utama dalam pengujian. Kedua, spesifikasi *webcam* mempengaruhi pengujian alat dikarenakan komponen ini merupakan bagian vital dalam alat deteksi. Untuk mendapatkan hasil yang maksimal memerlukan *webcam* / kamera yang lebih baik kemudian sudut kamera terhadap objek rambu lalu lintas perlu diperhatikan.

Alat deteksi memiliki lampu indikator yang bertujuan untuk menampilkan kelas dari hasil deteksi rambu lalu lintas. Di bawah ini adalah kondisi lampu indikator dalam berbagai kondisi yang sedang berlangsung.



Gambar 4.5. Kondisi Lampu LED Saat Proses Deteksi

BAB V

KESIMPULAN DAN SARAN

Bab ini membahas kesimpulan dari penelitian sistem deteksi rambu lalu lintas dan saran untuk pengembangan pada penelitian selanjutnya.

5.1. Kesimpulan

Kesimpulan dalam penelitian sistem deteksi rambu lalu lintas sesuai dengan pengujian yang telah dilakukan adalah sebagai berikut.

1. Parameter terbaik ditunjukkan dengan nilai *learning rate* 0.0005, nilai *batch* 8, dan jumlah *epoch* 26087. Parameter ini menghasilkan akurasi 81%, presisi 98%, *recall* 82%, dan rata-rata waktu komputasi 0.103 s pada pengujian Jupyter Notebook menggunakan CPU Raspberry Pi. Hasil pengujian dari parameter terbaik melalui Jupyter Notebook kemudian diujikan menggunakan TPU menghasilkan akurasi 81%, presisi 98%, *recall* 84.6%, dan rata-rata waktu komputasi 0.012 s. Pengujian menggunakan TPU memiliki waktu komputasi yang lebih cepat sebesar 8 kali lipat.
2. Hasil pengujian alat secara *realtime* dilakukan dengan meletakkan alat sejauh 75cm dari rambu lalu lintas. Hasil pengujian menggunakan CPU menghasilkan akurasi 89.6%, presisi 100%, *recall* 89.6%, dan rata-rata waktu komputasi 0.107 s. Hasil pengujian menggunakan TPU memiliki hasil evaluasi yang sama akan tetapi ada perbedaan dalam rata-rata waktu komputasi 0.018s. Hal ini TPU meningkatkan performa deteksi rambu lalu lintas sebesar 6 kali lipat dari pemrosesan melalui CPU.
3. Pengujian alat menggunakan CPU memiliki waktu komputasi yang lebih lambat dibandingkan dengan TPU dikarenakan perangkat Raspberry Pi menjalankan banyak proses yang memerlukan peran CPU. Adanya TPU mengurangi beban CPU sehingga waktu komputasi tidak mengganggu peran CPU.
4. Pengujian alat belum bisa dilakukan secara portabel dikarenakan memerlukan beberapa modul tambahan seperti layar LCD untuk menampilkan tampilan hasil deteksi, modul speaker untuk mengeluarkan suara rambu kelas, dan modul baterai yang disesuaikan dengan kebutuhan daya yang diperlukan oleh alat.

5.2. Saran

Hasil dari penelitian ini memiliki beberapa kekurangan dari segi pengembangan model deteksi rambu lalu lintas. Adapun beberapa saran dalam pengembangan model deteksi rambu lalu lintas adalah sebagai berikut.

1. Untuk meningkatkan kinerja model dapat dilakukan dengan mengubah bagian ekstraksi fitur menggunakan varian MobileNet yang lebih baru yaitu MobileNet-V3.
2. Untuk meningkatkan waktu pemrosesan deteksi dapat dilakukan dengan mengubah bagian detektor objek menggunakan varian *Single Shot Detector* (SSD) yang lebih ringan yaitu SSD-Lite.
3. Untuk mengoptimalkan model dapat dilakukan dengan mengubah algoritma pengoptimalan lain seperti ADAM (*Adaptive Moment Estimation*).

Hasil dari penelitian ini memiliki beberapa kekurangan dari segi pengembangan alat deteksi rambu lalu lintas. Adapun beberapa saran dalam pengembangan alat deteksi rambu lalu lintas adalah sebagai berikut.

1. Tampilan menggunakan *library* PyQt5 cukup memakan CPU pada Raspberry Pi sehingga tampilan memungkinkan berjalan lambat ketika banyak proses yang sedang berjalan pada CPU. Untuk mengatasi permasalahan ini dapat dilakukan dengan pengembangan tampilan menggunakan *library* yang dapat memanfaatkan GPU Raspberry Pi seperti *library* pi3d atau kivy.
2. Pengoptimalan dari sisi pengkodean dapat diubah ke dalam bahasa pemrograman C++ karena bahasa pemrograman tersebut memiliki kemampuan yang lebih baik dibandingkan bahasa pemrograman Python. Alternatif dapat dilakukan dengan menerapkan prinsip pengkodean *multithreading* atau *multiprocessing*.
3. Penambahan beberapa komponen lain seperti layar LCD, modul speaker, dan modul baterai. Layar LCD digunakan untuk menampilkan sistem deteksi, modul speaker digunakan untuk menghasilkan suara ketika rambu lalu lintas terdeteksi, dan modul baterai digunakan untuk sumber daya listrik portabel dengan memperhatikan kebutuhan daya yang diperlukan oleh semua alat. *Webcam* dapat diganti dengan modul kamera yang memiliki spesifikasi lebih tinggi sehingga kualitas gambar menjadi lebih baik.

DAFTAR PUSTAKA

- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., ... Zheng, X. (2016). TensorFlow: A system for large-scale machine learning. *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016*, 265–283.
- Adhikari, B. P. (2018). *Bishwo Prakash Adhikari Camera Based Object Detection for Indoor Scenes. March*, 64. <https://dspace.cc.tut.fi/dpub/handle/123456789/26084>
- Agarwal, K. (2018). Object Detection in Refrigerators using Tensorflow by. *Gastrointestinal Endoscopy*, 10(1), 279–288. <http://dx.doi.org/10.1053/j.gastro.2014.05.023><https://doi.org/10.1016/j.gie.2018.04.013><http://www.ncbi.nlm.nih.gov/pubmed/29451164><http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5838726><http://dx.doi.org/10.1016/j.gie.2013.07.022>
- Aggarwal, C. C. (2018). Neural Networks and Deep Learning. In *Machine Learning*. <https://doi.org/10.7551/mitpress/13811.003.0007>
- Alsing, O. (2018). *Mobile Object Detection using TensorFlow Lite and Transfer Learning*.
- Arcos-García, Á., Álvarez-García, J. A., & Soria-Morillo, L. M. (2018). Evaluation of deep neural networks for traffic sign detection systems. *Neurocomputing*, 316, 332–344. <https://doi.org/10.1016/j.neucom.2018.08.009>
- Athanasiadis, I., Mousoulitis, P., & Petrou, L. (2018). *A Framework of Transfer Learning in Object Detection for Embedded Systems*. <http://arxiv.org/abs/1811.04863>
- Ayi, M. (2020). *RMNV2: REDUCED MOBILENET V2 AN EFFICIENT LIGHTWEIGHT MODEL FOR HARDWARE DEPLOYMENT*. 3(2017), 54–67. <http://repositorio.unan.edu.ni/2986/1/5624.pdf>
- Badan Pengembangan Sumber Daya Manusia Kementrian PUPR. (2016). Pengenalan Rekayasa Keselamatan Jalan. *Journal of Chemical Information and Modeling*, 53(9), 1689–1699.
- Ben abdel ouahab, I., Elaachak, L., Elouaai, F., & Bouhorma, M. (2021). A Smart Surveillance Prototype Ensures the Respect of Social Distance During COVID19. In *Lecture Notes in Networks and Systems* (Vol. 183). Springer International Publishing. https://doi.org/10.1007/978-3-030-66840-2_91
- Chiu, Y. C., Tsai, C. Y., Ruan, M. Da, Shen, G. Y., & Lee, T. T. (2020). Mobilenet-SSDv2: An Improved Object Detection Model for Embedded Systems. *2020 International Conference on System Science and Engineering, ICSSE 2020*, 0–4. <https://doi.org/10.1109/ICSSE50014.2020.9219319>
- Dang, L., Pang, P., & Lee, J. (2020). Depth-wise separable convolution neural network with

- residual connection for hyperspectral image classification. *Remote Sensing*, 12(20), 1–20. <https://doi.org/10.3390/rs12203408>
- Fan, H., Liu, S., Ferianc, M., Ng, H. C., Que, Z., Liu, S., Niu, X., & Luk, W. (2018). A Real-Time Object Detection Accelerator with Compressed SSDLite on FPGA. *Proceedings - 2018 International Conference on Field-Programmable Technology, FPT 2018*, 2, 17–24. <https://doi.org/10.1109/FPT.2018.00014>
- Géron, A. (n.d.). *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly Media.
- Google. (2022). *10 Years of Raspberry Pi - History of Raspberry Pi and element14 Community - Documents - Raspberry Pi - element14 Community*. <https://community.element14.com/products/raspberry-pi/w/documents/27523/10-years-of-raspberry-pi---history-of-raspberry-pi-and-element14-community>
- Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., & Shuai, B. (2017). *Recent Advances in Convolutional Neural Networks*. 1–38.
- Hadi, D. S. (2014). *Modul grafika komputer* (Issue November). [http://setiawanhadi.unpad.ac.id/Welcome to Setiawan Hadi homepage_files/pendidikan/Ganjil1516/CG/Modul Grafika Komputer.pdf](http://setiawanhadi.unpad.ac.id/Welcome%20to%20Setiawan%20Hadi/homepage_files/pendidikan/Ganjil1516/CG/Modul%20Grafika%20Komputer.pdf)
- Herawati, H. (2019). Karakteristik Dan Penyebab Kecelakaan Lalu Lintas Di Indonesia Tahun 2012. *Warta Penelitian Perhubungan*, 26(3), 133. <https://doi.org/10.25104/warlit.v26i3.875>
- Hope, T., Resheff, Y. S., & Lieder, I. (2017). Learning TensorFlow: A Guide to Building Deep Learning Systems. In *Deep Learning With Tensorflow*.
- Houben, S., Stallkamp, J., Salmen, J., Schlipsing, M., & Igel, C. (2013). Detection of traffic signs in real-world images: The German traffic sign detection benchmark. *Proceedings of the International Joint Conference on Neural Networks*. <https://doi.org/10.1109/IJCNN.2013.6706807>
- Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. <http://arxiv.org/abs/1704.04861>
- Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *32nd International Conference on Machine Learning, ICML 2015*, 1, 448–456.
- Khan, A. I., & Al-Habsi, S. (2020). Machine Learning in Computer Vision. *Procedia Computer Science*, 167(2019), 1444–1451. <https://doi.org/10.1016/j.procs.2020.03.355>
- Khan, S., Rahmani, H., Shah, S. A. A., & Bennamoun, M. (2018). A Guide to Convolutional Neural Networks for Computer Vision. In *Synthesis Lectures on Computer Vision* (Vol. 8, Issue 1). <https://doi.org/10.2200/s00822ed1v01y201712cov015>
- Lamprousi, V. (2021). *Application of Machine Learning Algorithms for Post Processing of*

Reference Sensors Utilization of Deep Learning Methods for Object Detection to.


- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9905 LNCS, 21–37. https://doi.org/10.1007/978-3-319-46448-0_2
- Magalhães, S. A., Castro, L., Moreira, G., Dos Santos, F. N., Cunha, M., Dias, J., & Moreira, A. P. (2021). Evaluating the single-shot multibox detector and yolo deep learning models for the detection of tomatoes in a greenhouse. *Sensors*, 21(10), 1–24. <https://doi.org/10.3390/s21103569>
- McAndrew, A. (2004). *An Introduction to Digital Image Processing with Matlab Notes for SCM2511 Image Processing 1 Semester 1*.
- Melinte, D. O., Travediu, A. M., & Dumitriu, D. N. (2020). Deep convolutional neural networks object detector for real-time waste identification. *Applied Sciences (Switzerland)*, 10(20), 1–18. <https://doi.org/10.3390/app10207301>
- Mordvintsev, A., & Abid, K. (2017). OpenCV-Python Tutorials Documentation. *OpenCV Python Documentation*, 269. <https://media.readthedocs.org/pdf/opencv-python-tutroals/latest/opencv-python-tutroals.pdf>
- Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S. (2018). *Activation Functions: Comparison of trends in Practice and Research for Deep Learning*. 1–20. <http://arxiv.org/abs/1811.03378>
- Padilla, R., Netto, S. L., & Da Silva, E. A. B. (2020). A Survey on Performance Metrics for Object-Detection Algorithms. *International Conference on Systems, Signals, and Image Processing*, 2020-July, 237–242. <https://doi.org/10.1109/IWSSIP48289.2020.9145130>
- Patterson, J., & Gibson, A. (2005). *Deep Learning: A Practitioner's Approach*.
- Perhubungan, M., & Indonesia, R. (2014). *Menteri perhubungan republik indonesia*.
- Pichaikutty, P. (2020). *Detection of curbside storm drain from street level images using Faster R-CNN*.
- Pinto de Aguiar, A. S., Neves dos Santos, F. B., Feliz dos Santos, L. C., de Jesus Filipe, V. M., & Miranda de Sousa, A. J. (2020). Vineyard trunk detection using deep learning – An experimental device benchmark. *Computers and Electronics in Agriculture*, 175(May), 105535. <https://doi.org/10.1016/j.compag.2020.105535>
- Rajendran, S. P., Shine, L., R., P., & Vijayaraghavan, S. (2019). Real Time Traffic Sign Recognition using Yolov3 based Detector. *Ayan*, 8(5), 55.
- Reithaug, A. (2018). Employing Deep Learning for Fish Recognition. *Western Norway University of Applied Science*, June.
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A.,


















- Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3), 211–252. <https://doi.org/10.1007/s11263-015-0816-y>
- Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 4510–4520. <https://doi.org/10.1109/CVPR.2018.00474>
- Shin, D., & Kim, J. (2022). *applied sciences A Deep Learning Framework Performance Evaluation to Use YOLO in Nvidia Jetson Platform*.
- Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1). <https://doi.org/10.1186/s40537-019-0197-0>
- Singh, P., & Manure, A. (2020). Learn TensorFlow 2.0. In *Learn TensorFlow 2.0*. <https://doi.org/10.1007/978-1-4842-5558-2>
- Thompson, J. (2016). *Python's Companion*. 360.
- Tieleman, T., & Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2), 26–31.
- Tramontano, A. G., Ghidoni, S., Bellotto, N., & Jaycee Lock, I. (2018). *Deep Learning Networks for Real-time Object Detection on Mobile Devices*.
- Undang-Undang Republik Indonesia No. 22 Tahun 2009 Tentang Lalu Lintas Dan Angkutan Jalan. (2009). 12–42.
- Wani, M. A., Bhat, F. A., Afzal, S., & Khan, A. I. (2019). *Advances in Deep Learning* (Vol. 57). <https://doi.org/10.1007/978-981-13-6794-6>
- Zhang, Z., Zhou, X., Chan, S., Chen, S., & Liu, H. (2017). Faster R-CNN for small traffic sign detection. *Communications in Computer and Information Science*, 773, 155–165. https://doi.org/10.1007/978-981-10-7305-2_14
- Zhao, W., Wang, Z., & Yang, H. (2018). *Traffic Sign Detection Based on Faster R-CNN in Scene Graph*. 154(Meees), 35–42. <https://doi.org/10.2991/meees-18.2018.8>

LAMPIRAN-LAMPIRAN

Lampiran 1. Representasi Citra Rambu Lalu Lintas (GTSDB)

Tabel 1. Representasi Dataset

Id Citra	Citra	Kelas		Id Citra	Citra	Kelas
0		<i>1 Prohibitory</i>		13		<i>4 Other</i>
1		<i>1 Prohibitory</i>		14		<i>4 Other</i>
2		<i>1 Prohibitory</i>		15		<i>1 Prohibitory</i>
3		<i>1 Prohibitory</i>		16		<i>1 Prohibitory</i>
4		<i>1 Prohibitory</i>		17		<i>4 Other</i>
5		<i>1 Prohibitory</i>		18		<i>3 Danger</i>
6		<i>4 Other</i>		19		<i>3 Danger</i>
7		<i>1 Prohibitory</i>		20		<i>3 Danger</i>
8		<i>1 Prohibitory</i>		21		<i>3 Danger</i>
9		<i>1 Prohibitory</i>		22		<i>3 Danger</i>
10		<i>1 Prohibitory</i>		23		<i>3 Danger</i>
11		<i>3 Danger</i>		24		<i>3 Danger</i>
12		<i>4 Other</i>		25		<i>3 Danger</i>

Id Citra	Citra	Kelas		Id Citra	Citra	Kelas
26		3 <i>Danger</i>		35		2 <i>Mandatory</i>
27		3 <i>Danger</i>		36		2 <i>Mandatory</i>
28		3 <i>Danger</i>		37		2 <i>Mandatory</i>
29		3 <i>Danger</i>		38		2 <i>Mandatory</i>
30		3 <i>Danger</i>		39		2 <i>Mandatory</i>
31		3 <i>Danger</i>		40		2 <i>Mandatory</i>
32		4 <i>Other</i>		41		4 <i>Other</i>
33		2 <i>Mandatory</i>		42		4 <i>Other</i>
34		2 <i>Mandatory</i>				

Lampiran 2. Kode Program Main (mainqt.py)

```
from helper import *
from PyQt5.QtCore import QTimer, QPoint, pyqtSignal
from PyQt5.QtWidgets import QApplication, QMainWindow, QTextEdit,
QLabel
from PyQt5.QtWidgets import QWidget, QAction, QVBoxLayout, QHBoxLayout
from PyQt5.QtGui import QFont, QPainter, QImage, QTextCursor
import queue as Queue

class Window(QMainWindow):
    text_update = pyqtSignal(str)
    def __init__(self, parent=None):
        QMainWindow.__init__(self, parent)
        QMainWindow.setFixedSize(self, 660, 600)
        self.central = QWidget(self)
        self.textbox = QTextEdit(self.central)
        self.textbox.setFont(FONT)
        self.textbox.setMaximumSize(640, 100)
        self.textbox.setReadOnly(True)
        self.textbox.setAlignment(Qt.AlignLeft)
        self.text_update.connect(self.append_text)
        sys.stdout = self
        print("Output:")

        self.vlayout = QVBoxLayout()
        self.displays = QHBoxLayout()
        self.disp = ImageWidget(self)
        self.displays.addWidget(self.disp)
        self.vlayout.addLayout(self.displays)
        self.vlayout.addWidget(self.textbox)
        self.central.setLayout(self.vlayout)
        self.setCentralWidget(self.central)

        exitAction = QAction('&Exit', self)
        exitAction.setShortcut('Ctrl+Q')
        exitAction.triggered.connect(self.close)
        self.addAction(exitAction)

        TITLE = "Traffic Sign Detection Using Raspberry Pi 4"
        self.setWindowTitle(TITLE)
        self.setWindowIcon(QIcon('Undip.png'))
        self.show()

    def start(self):
        self.timer = QTimer(self)
        self.timer.timeout.connect(lambda:
            self.show_image(img_queue, self.disp))
        self.timer.start(DISPLAY_MSEC)
        self.capture_thread = threading.Thread(target=main,
            args=(camera_num, img_queue))
        self.capture_thread.start()

    def show_image(self, imageq, display):
        if not imageq.empty():
            image = imageq.get()
            if image is not None and len(image) > 0:
                img = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
                self.display_image(img, display)
```

```

def display_image(self, img, display):
    qimg = QImage(img.data, img.shape[1], img.shape[0], FORMAT)
    display.setImage(qimg)

def write(self, text):
    self.text_update.emit(str(text))

def flush(self):
    pass

def append_text(self, text):
    cur = self.textbox.textCursor()
    cur.movePosition(QTextCursor.End)
    s = str(text)
    while s:
        head, sep, s = s.partition("\n")
        cur.insertText(head)
        if sep:
            cur.insertBlock()
    self.textbox.setTextCursor(cur)

def closeEvent(self, event):
    global capture
    capture = False
    self.capture_thread.join()

class ImageWidget(QWidget):
    def __init__(self, parent=None):
        super(ImageWidget, self).__init__(parent)
        self.image = None

    def setImage(self, image):
        self.image = image
        self.setMinimumSize(image.size())
        self.update()

    def paintEvent(self, event):
        qp = QPainter()
        qp.begin(self)
        if self.image:
            qp.drawImage(QPoint(0, 0), self.image)
        qp.end()

def main(cam_num, queue):
    frame_rate_calc = 1
    time1 = 0
    freq = cv2.getTickFrequency()
    cap = WebcamVideoStream(src=CAM_NUM, res=RES, fps=FPS,
    api= API).start()
    time.sleep(1)
    while capture:
        try:
            start = time.perf_counter()
            _, image = cap.read()
            frame_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
            frame_resized = cv2.resize(frame_rgb, (w, h))
            input_data = np.expand_dims(frame_resized, axis=0)
            results = detect_objects(interpreter, input_data,
            threshold)

```



```

        image = annotate_objects(image, results, RES[0], RES[1],
                                labels, time1, frame_rate_calc)
        time1 = (time.perf_counter()-start)
        frame_rate_calc = 1/time1
        image = cv2.resize(image, IMG_SIZE)
        if image is not None and queue.qsize() < 5:
            queue.put(image)
        else:
            time.sleep(DISP_MSEC / 1000.0)
    except:
        print("Failed")
        break
cap.stop()
del cap
time.sleep(2)

if __name__ == '__main__':
    img_queue = Queue.Queue()
    FORMAT = QImage.Format_RGB888
    DISP_MSEC = 1
    FONT = QFont("Helvetica", 10)
    capture = True

    IMG_SIZE = (640, 480)
    RES = (640, 480)

    CAM_NUM = 0
    API = cv2.CAP_V4L2
    FPS = 30

    USE_TPU = 0
    NUM_THREAD = 4
    threshold = 0.5

    GRAPH_FOLDER = 'export/SSD/5/1501/tflite/'
    GRAPH_NAME = 'detect_model.tflite'
    LABEL_NAME = 'data/TFLite_label.txt'

    result = searchFile(GRAPH_FOLDER)
    interpreter = make_interpreter(result, NUM_THREAD)
    time.sleep(2)

    _, h, w, _ = interpreter.get_input_details()[0]['shape']

    labels = load_tflite_label(LABEL_NAME)
    print(labels)
    app = QApplication(sys.argv)
    win = Window()
    win.start()
    sys.exit(app.exec_())

```

Lampiran 3. Kode Program Helper (helper.py)

```
import numpy as np
import sys, time, threading, cv2, os, re, platform, importlib, imutils
from imutils.video import VideoStream
from tfllite_runtime.interpreter import Interpreter
from tfllite_runtime.interpreter import load_delegate
from led import *

def check_platform():
    return platform.system()

def searchFile(folder):
    result = []
    for root, dirs, files in os.walk(folder):
        for file in files:
            if file.endswith(".tflite"):
                result.append(os.path.join(os.getcwd(), root, file))

    return result

def make_interpreter(result, NUM_THREAD):
    platform = check_platform()
    edgetpu = None
    try:
        if platform == "Windows":
            edgetpu = load_delegate('edgetpu.dll')
        if platform == "Linux":
            edgetpu = load_delegate('libedgetpu.so.1')
            print("halo")
    except:
        pass
    if edgetpu is None:
        print("Edge TPU Not Detected")
        NUM_THREAD = 2
        interpreter = Interpreter(model_path=result[0],
                                num_threads=NUM_THREAD)
    else:
        print("Edge TPU Detected")
        interpreter = Interpreter(model_path=result[1],
                                num_threads=NUM_THREAD, experimental_delegates=[edgetpu])
    interpreter.allocate_tensors()
    return interpreter

def load_tflite_label(path):
    folder_path = os.path.join(os.getcwd(), path)
    with open(folder_path, 'r', encoding='utf-8') as f:
        lines = f.readlines()
        ret = {}
        for row_number, content in enumerate(lines):
            pair = re.split(r'[:\s]+', content.strip(), maxsplit=1)
            if len(pair) == 2 and pair[0].strip().isdigit():
                ret[int(pair[0])] = pair[1].strip()
            else:
                ret[row_number] = pair[0].strip()
    return ret
```

```

def set_input_tensor(interpreter, image):
    """Sets the input tensor."""
    tensor_index = interpreter.get_input_details()[0]['index']
    input_tensor = interpreter.tensor(tensor_index)()[0]
    input_tensor[:, :] = image

def get_output_tensor(interpreter, index):
    output_details = interpreter.get_output_details()[index]
    tensor =
    np.squeeze(interpreter.get_tensor(output_details['index']))
    return tensor

def detect_objects(interpreter, image, threshold):
    set_input_tensor(interpreter, image)
    interpreter.invoke()
    boxes = get_output_tensor(interpreter, 0)
    classes = get_output_tensor(interpreter, 1)
    scores = get_output_tensor(interpreter, 2)
    count = int(get_output_tensor(interpreter, 3))
    results = []
    for i in range(count):
        if scores[i] >= threshold:
            result = {
                'bounding_box': boxes[i],
                'class_id': classes[i],
                'score': scores[i]
            }
            results.append(result)
    return results

def annotate_objects(frame, results, WIDTH, HEIGHT, labels, time1,
frame_rate_calc):
    color_box = [(0,0,255), (255,0,0), (0,255,255), (255,255,0)]
    hide_led()
    for obj in results:
        ymin, xmin, ymax, xmax = obj['bounding_box']
        ymin = int(max(1, (ymin * HEIGHT)))
        xmin = int(max(1, (xmin * WIDTH)))
        ymax = int(min(HEIGHT, (ymax * HEIGHT)))
        xmax = int(min(WIDTH, (xmax * WIDTH)))
        cl = int(obj['class_id']+1)
        classid =labels[cl-1]
        score = obj['score']
        print('%4d %4d %4d %4d %d %-11s %.2f %fs %.1f' % (xmin,
ymin, xmax, ymax, cl, classid, score, time1, frame_rate_calc))
        cv2.rectangle(frame, (xmin, ymin), (xmax, ymax),
color_box[cl-1], 2)
        cv2.putText(frame, '%d: %s %.2f' % (cl, classid, score ),
(xmin-5, ymin-5), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
color_box[cl-1], 2)
        show_led(cl)
    return frame

```

```

class WebcamVideoStream:
    def __init__(self, src=0, res=(640,480), fps=30,
api=cv2.CAP_V4L2,name="WebcamVideoStream"):
        self.stream = cv2.VideoCapture(src, api)
        self.stream.set(cv2.CAP_PROP_FRAME_WIDTH, res[0])
        self.stream.set(cv2.CAP_PROP_FRAME_HEIGHT, res[1])
        self.stream.set(cv2.CAP_PROP_FPS, fps)
        (self.grabbed, self.frame) = self.stream.read()
        self.name = name
        self.stopped = False

    def start(self):
        # start the thread to read frames from the video stream
        t = Thread(target=self.update, name=self.name, args=())
        t.daemon = True
        t.start()
        return self

    def update(self):
        # keep looping infinitely until the thread is stopped
        while True:
            if self.stopped:
                return
            (self.grabbed, self.frame) = self.stream.read()

    def read(self):
        return True,self.frame

    def stop(self):
        self.stopped = True
        self.stream.release()

```

Lampiran 4. Kode Program LED (led.py)

```
from luma.led_matrix.device import max7219
from luma.core.interface.serial import spi, noop
from luma.core.render import canvas
from luma.core.legacy import text

import time

serial = spi(port=0, device=0, gpio=noop())
device = max7219(serial, rotate=2)

def show_led(cl):
    classid = str(cl)
    with canvas(device) as draw:
        text(draw, (1, 0), classid, fill="white")
    time.sleep(1/1000)

def hide_led():
    with canvas(device) as draw:
        text(draw, (0, 0), "", fill="white")
    time.sleep(1/1000)

def poweroff_led():
    device.cleanup()
```