TUGAS AKHIR – KM184801

PENGENALAN PLAT NOMOR KENDARAAN BERMOTOR DENGAN METODE YOLO-DARKNET

Dyah Ayu Amini
NRP 0611160000025

Dosen Pembimbing
Dr. Budi Setiyono, S.Si, MT

DEPARTEMEN MATEMATIKA
Fakultas Sains Dan Analitika Data
Institut Teknologi Sepuluh Nopember
Surabaya 2020

“Halaman ini sengaja dikosongkan”

TUGAS AKHIR – KM184801

PENGENALAN PLAT NOMOR KENDARAAN BERMOTOR DENGAN METODE YOLO-DARKNET

Dyah Ayu Amini
NRP 0611160000025

Dosen Pembimbing
Dr. Budi Setiyono, S.Si, MT

DEPARTEMEN MATEMATIKA
Fakultas Sains Dan Analitika Data
Institut Teknologi Sepuluh Nopember
Surabaya 2020

“Halaman ini sengaja dikosongkan”

FINAL PROJECT – KM184801

NUMBER PLATE RECOGNITION ON VEHICLE USING YOLO-DARKET

Dyah Ayu Amini
NRP 0611160000025

Supervisor
Dr. Budi Setiyono, S.Si, MT

DEPARTMENT OF MATHEMATICS
Faculty Of Science And Data Analytics
Sepuluh Nopember Institute of Technology
Surabaya 2020

“Halaman ini sengaja dikosongkan”

LEMBAR PENGESAHAN

PENGENALAN PLAT NOMOR KENDARAAN BERMOTOR DENGAN METODE YOLO-DARKNET

NUMBER PLATE RECOGNITION ON VEHICLE USING YOLO-DARKNET

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
Untuk memperoleh gelar Sarjana Matematika
Pada bidang studi Ilmu Komputer
Program Studi S-1 Departemen Matematika
Fakultas Sains dan Analitika Data
Institut Teknologi Sepuluh Nopember Surabaya

Oleh:
DYAH AYU AMINI
NRP. 06111640000025

Menyetujui,
Dosen Pembimbing

Budi Setiyono, S.Si, MT
NIP. 19720207 199702 1 001

Mengetahui,
Kepala Departemen Matematika
FSAD ITS

Subchan, Ph.D
NIP. 19710513 199702 1 001

Surabaya, 19 Agustus 2020

Halaman ini sengaja dikosongkan

PENGENALAN PLAT NOMOR KENDARAAN BERMOTOR DENGAN METODE YOLO-DARKNET

Nama	:	Dyah Ayu Amini
NRP	:	06111640000025
Departemen	:	Matematika FSAD ITS
Dosen Pembimbing	:	Budi Setiyono, S.Si, MT

Abstrak

Pengenalan karakter merupakan salah satu tahap dalam sistem pengenalan plat nomor. Pengenalan karakter dilakukan untuk mendapatkan data karakter teks. Metode yang diusulkan adalah YOLOv3 (*You Only Look Once*) dan sebagai *feature extractor* digunakan Darknet-53. Pada penelitian ini, data yang digunakan adalah gambar plat nomor yang berasal dari hasil ekstraksi dan *cropping* video kendaran bermotor yang telah diambil menggunakan *handphone* dan kamera. *testing* dilakukan dengan dua model berbeda yaitu model yang didapat dengan tambahan *preprocessing* data dan model yang didapat tanpa adanya *preprocessing* data. *Preprocessing* data dilakukan untuk memperbaiki kualitas citra plat nomor.

Testing dilakukan pada dataset citra plat nomor tanpa gangguan dan dataset citra plat nomor dengan gangguan berupa penambahan dan pengurangan intensitas warna (*brightness*) pada citra.

Untuk model yang didapat dari *training* data tanpa *preprocessing*, nilai akurasi pengenalan plat nomor paling tinggi yang diperoleh sebesar 80% dan akurasi pengenalan karakter sebesar 97.1%. Sementara itu untuk model yang didapat dari *training* data dengan *preprocessing*, akurasi pengenalan plat nomor paling tinggi yang didapat sebesar 88% dan akurasi pengenalan karakter sebesar 98.2%.

Kata Kunci : Pengenalan Plat Nomor, YOLO-Darknet, Pengenalan Karakter, *Preprocessing*.

Halaman ini sengaja dikosongkan

NUMBER PLATE RECOGNITION ON VEHICLE USING YOLO-DARKNET

Name	:	Dyah Ayu Amini
NRP	:	06111640000025
Department	:	Mathematics FSDA-ITS
Supervisor	:	Budi Setiyono, S.Si, MT

Abstract

Character recognition is one of the stages in the number plate recognition system. Character recognition is performed to get text character data. The method that used in this research is YOLO (You Only Look Once). The type of YOLO used is YOLOv3 and the network architecture used is Darknet-53. In this research, the data used are number plate images derived from the extracting and cropping of vehicle videos that have been taken using mobile phones and cameras.

Testing is done with two different models. The first is the model obtained with additional preprocessing and the second is the model obtained without preprocessing. Preprocessing is done to improve the quality of image plates that small and blurry. The test is carried out on various types of datasets, the first is the number plate image dataset without interference and the second is the number plate image dataset with addition and reduction of brightness.

For the model obtained without preprocessing, the highest accuracy of number plate recognition was 80% *and character recognition accuracy was* 97.1%*. Meanwhile, for the model obtained by preprocessing, the highest accuracy of plat recognition was* 88%*, and the accuracy of character recognition was* 98.2%*.*

Key Words : *Plate Recognition, YOLO-Darknet, Character Recognition, Preprocessing.*

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Alhamdulillah, puji syukur dipanjatkan kepada Allah SWT atas segala rahmat dan karunia-Nya sehingga penyusunan Tugas Akhir yang berjudul :

PENGENALAN PLAT NOMOR KENDARAAN BERMOTOR DENGAN METODE YOLO-DARKNET

dapat terselesaikan dengan baik dan tepat waktu sebagai salah satu syarat kelulusan dalam menempuh program studi S1 departemen Matematika FSAD ITS Surabaya.

Tugas Akhir ini dapat terselesaikan dengan baik dan tepat waktu berkat dukungan dari banyak pihak. Sehubungan dengan hal tersebut, penulis bermaksud menyampaikan ucapan terima kasih kepada :

1. Orang tua tercinta, Bapak Anang Zubaidi dan Ibu Sumiyati, atas segala doa, kasih sayang, perhatian, bimbingan, wejangan, dan dukungan (baik materi maupun non materi) yang tiada henti.

2. Adik-adik tersayang, Hana Ajeng Pratiwi, Elisa Kartika Andini, dan Retno Elyana Putri, atas segala kasih sayang, kebersamaan, doa, semangat, motivasi, dan perhatian yang tiada henti.

3. Bapak Budi Setiyono, S.Si, MT, selaku dosen pembimbing yang telah memberikan bimbingan, saran, kritik, dan motivasi selama proses pengerjaan Tugas Akhir.

4. Bapak Drs. Sadjidon, M.Si, Bapak Drs. Daryono Budi Utomo, M.Si, dan Ibu Dr. Dwi Ratna Sulistyaningrum, M.T, selaku dosen penguji yang sudah memberikan masukan yang membangun.

5. Bapak Subchan, Ph.D, selaku Kepala Departemen Matematika FSAD ITS, yang telah memberikan motivasi dan dukungan kepada penulis.

6. Ibu Dr. Dwi Ratna Sulistyaningrum, M.T selaku Sekretaris Bidang Akademik Departemen Matematika FSAD ITS, yang telah membantu penulis dalam hal menyelesaikan kebutuhan administrasi selama periode pengambilan Tugas Akhir.

7. Seluruh Bapak dan Ibu Dosen Departemen Matematika ITS, yang telah memberikan ilmu dan nasihat selama perkuliahan.

8. Seluruh Staf Tata Usaha Departemen Matematika ITS yang telah membantu dalam proses mengurus administrasi selama perkuliahan.

9. Seluruh teman-teman mahasiswa Departemen Matematika ITS angkatan 2016 (LEMNISCATE), atas kebersamaan, canda tawa, suka duka, dan pengalaman lainnya yang tak terlupakan oleh penulis.

Penulis juga mengharapkan kritik dan saran dari berbagai pihak untuk perbaikan isi Tugas Akhir ini. Segala kritik dan saran akan penulis terima dan akan disambut dengan senang hati.

Surabaya, 19 Agustus 2020

Dyah Ayu Amini

DAFTAR ISI

Halaman

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

Pada bab pendahuluan ini, dipaparkan latar belakang masalah, rumusan masalah, batasan masalah, tujuan, dan manfaat penulisan dari Tugas Akhir.

1.1 Latar Belakang Masalah

Plat nomor adalah salah satu jenis identifikasi kendaraan bermotor. Plat nomor juga disebut plat registrasi kendaraan. Bentuknya berupa potongan plat logam atau plastik yang dipasang pada kendaraan bermotor sebagai identifikasi resmi. Biasanya plat nomor jumlahnya sepasang, untuk dipasang di depan dan belakang kendaraan. Berbagai metode pengenalan plat nomor kendaraan telah dilakukan. Secara umum, algoritma-algoritma tersebut dikembangkan dari 3 langkah, yakni pencarian area plat nomor, segmentasi karakter dari plat nomor dan pengenalan dari setiap karakter [6]. Sistem pengenalan plat nomor kendaraan bermotor merupakan sebuah aplikasi yang menggantikan fungsi pengelihatan manusia dalam hal mengenali plat nomor kendaraan bermotor.

Salah satu tahap penting dalam pengenalan plat nomor adalah tahap pengenalan karakter. Dalam tahap pengenalan karakter ini akan dihasilkan suatu keluaran berupa karakter teks. Tahap sebelum pengenalan karakter dilakukan preprocessing kepada citra citra plat nomor yang akan diidentifikasi. Pengenalan karakter yang umum disebut sebagai OCR (*Optical Character Recognition*) termasuk pada pengenalan pola. Pengertian pengenalan pola sendiri merupakan suatu sistem yang mencoba untuk membaca / mengenali apakah citra masukan yang diterima cocok dengan salah satu citra yang telah ditentukan. Pengaplikasian pengenalan pola diantaranya seperti pendeteksi sidik jari, tulisan, tanda tangan, bahkan wajah seseorang. Tujuan dari pengenalan karakter adalah untuk mendukung perkembangan teknologi yang semakin pesat dalam bentuk digital. Sistem aplikasi OCR yang akan dibangun ini adalah

khusus untuk mengenali karakter pada citra gambar yang berasal dari proses ekstraksi dan *cropping* video yang memuat plat nomor kendaraan bermotor.

Tujuan dari pengenalan karakter adalah untuk mendukung perkembangan teknologi yang semakin pesat dalam bentuk digital. Sehingga bila terdapat suatu data fisik yang dikehendaki menjadi bentuk digital, maka sistem pengenal karakter ini dapat dimanfaatkan [7].

Beberapa metode untuk pengenalan plat nomor telah digunakan, penelitian yang dilakukan Hui Li, Chunhua Shen (2016) berjudul "Reading Car License Plates Using Deep Convolutional Neural Networks and LSTMs", penelitian ini menangani masalah deteksi dan pengenalan plat nomor pada sebuah gambar mobil. Keuntungan utama dari metode yang digunakan dalam penelitian ini adalah ini segmentasi yang mudah [12] namun waktu yang dibutuhkan masih cukup lama. Selanjutnya penelitian juga dilakukan oleh Syed Zain Masood, Guang Shu, Afshin Dehghan dan Enrique G. Ortiz (2017). yang berjudul "License Plate Detection and Recognition Using Deeply Learned Convolutional Neural Networks", sistem ini dibangun dengan menggunakan rangkaian Convolutional Neural Networks (CNN) yang saling terkait dengan akurat dan efisien. Pada penelitian ini CNN dilatih dan diatur agar mendapatkan hasil yang baik ketika diterapkan pada kondisi yang berbeda (misalnya pencahayaan, oklusi, dll.). Dan dapat bekerja di berbagai jenis plat nomor (misalnya ukuran, latar belakang, font, dll) [11] namun kelemahan nya adalah waktu *training* yang diperlukan masih cukup lama.

Metode YOLO diperkenalkan pertama kali oleh Redmon et al. (2016) dalam judul You Only Look Once : Unified, Real-Time Detection. Dalam penelitiannya, selain arsitekturnya yang sederhana, dikatakan pula bahwa YOLO sangat cepat dalam mengidentifikasi objek dan akurasi rata-rata yang didapatkan cukup tinggi dengan presentase mencapai angka 88% dalam ImageNet 2012 Validation [1]. Karena itu, pada penelitian ini metode YOLO *(You Only Look*

Once)-Darknet dipilih untuk melakukan pengenalan plat nomor.

1.2 Rumusan Masalah

Rumusan masalah yang digunakan pada Tugas Akhir ini adalah sebagai berikut.

1. Bagaimana cara menerapkan metode YOLO-Darknet untuk mengenali plat nomor sehingga gambar plat nomor yang berasal dari ekstraksi dan *cropping* video dapat diubah menjadi teks?

2. Bagaimana tingkat akurasi yang diperoleh saat metode YOLO-Darknet diterapkan untuk mengenali plat nomor?

1.3 Batasan Masalah

Batasan masalah yang digunakan padal Tugas Akhir ini adalah sebagai berikut.

1. Data pada penelitian ini berupa gambar yang berasal dari ekstraksi dan *cropping* video. Video diambil menggunakan *handphone* dan kamera pada siang hari dengan kondisi cerah. Video diambil di jalur satu arah dengan arah kendaraan mendekati kamera.

2. Citra plat nomor yang digunakan adalah plat nomor kendaraan dengan warna dasar plat hitam yang berlaku di Indonesia.

3. Citra karakter plat nomor yang digunakan adalah karakter standar dari plat nomor dan bukan karakter modifikasi.

4. Sistem ini menangani pengenalan plat nomor berdasarkan karakter huruf dan angka (A-Z dan 0-9).

1.4 Tujuan

Adapun tujuan penyusunan Tugas Akhir ini adalah untuk menerapkan metode YOLO-Darknet untuk mengenali plat nomor sehingga gambar plat nomor yang didapat dengan cara ekstraksi dan *cropping* video dapat dikenali sebagai teks. Selain itu adalah untuk mengetahui tingkat akurasi yang diperoleh saat metode YOLO-Darknet diterapkan untuk mengenali plat nomor.

1.5 Manfaat

Pengenalan plat nomor kendaraan bermotor dapat membantu mempercepat proses identifikasi kendaraan bermotor sehingga data kendaraan bermotor yang sudah teridentifikasi dapat disimpan dan digunakan sesuai kebutuhan.

BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

2.1.1 Penelitian Hui Li dan Penelitian Syed Zain Masood

Metode yang telah digunakan untuk mengenali plat nomor salah satunya adalah penelitian yang dilakukan Hui Li, Chunhua Shen (2016) berjudul "Reading Car License Plates Using Deep Convolutional Neural Networks and LSTMs", penelitian ini menangani masalah deteksi dan pengenalan plat nomor pada sebuah gambar mobil. Keuntungan utama dari metode yang digunakan dalam penelitian ini adalah ini segmentasi yang mudah [12] namun waktu yang dibutuhkan masih cukup lama. Selanjutnya penelitian juga dilakukan oleh Syed Zain Masood, Guang Shu, Afshin Dehghan dan Enrique G. Ortiz (2017). yang berjudul "License Plate Detection and Recognition Using Deeply Learned Convolutional Neural Networks", sistem ini dibangun dengan menggunakan rangkaian Convolutional Neural Networks (CNN) yang saling terkait dengan akurat dan efisien. Pada penelitian ini CNN dilatih dan diatur agar mendapatkan hasil yang baik ketika diterapkan pada kondisi yang berbeda (misalnya pencahayaan, oklusi, dll.). Dan dapat bekerja di berbagai jenis plat nomor (misalnya ukuran, latar belakang, font, dll) [11] namun kelemahan nya adalah waktu *training* yang diperlukan masih cukup lama.

2.1.2 Penelitian Christya Aji Putra

Dalam lingkup ITS penelitian tentang pengenalan plat nomor juga telah dilakukan. Salah satunya adalah penelitian yang dilakukan oleh Christya Aji Putra pada tahun 2015. dengan judul "Pengenalan Karakter Plat Nomor Kendaraan Bermotor Menggunakan Extreme Learning Machine". Penelitian ini membahas tentang pengenalan karakter pada plat nomor dengan menggunakan

metode *Extreme Learning Machine* yang merupakan jaringan syaraf tiruan *feedforward* dengan satu *hidden layer* [4].

Kelemahan yang terdapat dari metode ini adalah jumlah *nodes hidden layer* yang ditentukan secara *try and error* sehingga tidak diketahui secara pasti berapa jumlah yang seharusnya digunakan.

2.1.3 Penelitian Rayson Laroca

Pada tahun 2018 Rayson Laroca et al. melakukan penelitian yang menerapkan metode YOLO untuk melakukan deteksi dan pengenalan plat nomor. Penelitian ini diberi judul "A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector" [13].

Pada penelitian ini dibahas masalah deteksi dan pengenalan plat nomor dengan data masukan yang digunakan adalah video. Terdapat beberapa tahapan yang dilakukan pada penelitian ini. Tahap pertama adalah deteksi kendaraan dan plat nomor dengan menggunakan metode Fast-YOLO dan YOLOv2 dimana Fast-YOLO membutuhkan waktu lebih cepat untuk melakukan deteksi yaitu mencapai 47 *frame* tiap detik dibandingkan dengan YOLOv2 yang hanya mencapai 35 *frame* tiap detik sementra untuk akurasi sementara untuk akurasi yang didapat, Fast-YOLO memperoleh akurasi sebesar 97.3% dan YOLOv2 memperoleh akurasi sebesar 99%. Tahap selanjutnya adalah segmentasi karakter yang dilakukan dengan metode CNN dilanjutkan dengan pengenalan karakter yang juga dilakukan dengan metode CNN.

2.1.4 Penelitian Hendry dan Rung-Ching Chen

Hendry dan Rung-Ching Chen pada tahun 2019 telah melakukan penelitian menggunakan metode YOLO. Penelitian yang dilakukan berjudul "Automatic License Plate Recognition via sliding-window-darknet-YOLO Deep Learning" [5]. Pada penelitian ini dibahas

masalah deteksi dan pengenalan plat nomor dengan metode YOLO (You Only Look Once) yang diterapkan pada gambar plat nomor.

Pada penelitian ini dilakukan modifikasi pada arsitektur jaringan YOLO dan membentuk tinyYOLO dengan 13 *convolutional layers*. Deteksi dan pengenalan dilakukan untuk mengenali plat nomor Taiwan. Data yang digunakan berasal dari AOLP (Aplication Oriented Lisence Plate) Dataset berupa plat nomor yang memuat enam digit karakter. Data latih yang digunakan berjumlah 581 gambar sementara data uji yang digunakan berjumlah 100 gambar.Sistem ini membutuhkan waktu 0.8 hingga 1 detik untuk melakukan deteksi dan pengenalan pada setiap input gambar. Akurasi yang didapat pada saat deteksi plat nomor mencapai 98.22% sementara untuk pengenalan plat nomor akurasi yang didapat hanya mencapai 78%.

2.1.5 Perkembangan metode YOLO

Metode YOLO diperkenalkan pertama kali oleh Redmon et al. (2016) dengan judul ”You Only Look Once : Unified, Real-Time” Detection. Dalam penelitiannya dikatakan bahwa YOLO memiliki arsitektur yang sederhana yaitu hanya dengan menerapkan *single neural network* untuk menentukan *Bounding box* dan probabilitas kelas dari objek yang dideteksi. Selain itu dikatakan pula bahwa YOLO sangat cepat sehingga dapat digunakan untuk melakukan proses deteksi secara *real time* [1].

Terdapat tiga versi YOLO yang telah dikembangkan, versi pertama YOLO yang lebih dikenal dengan nama YOLOv1 memiliki arsitektur jaringan yang terdiri dari 24 *convolutional layers* yang diikuti oleh 2 *fully connected layers* dengan dimensi gambar masukan yang digunakan adalah 224×224 dan kecepatan untuk melakukan deteksi yaitu mencapai 45 *frame* tiap detik [1]. Adapun kelemahan dari YOLOv1 adalah sulitnya mendeteksi objek yang berukuran kecil selain itu arsitektur jaringan juga mengalami kesulitan dalam melakukan deteksi objek ketika dimensi gambar uji coba berbeda dengan dimensi dari gambar yang dilatih [1].

Versi kedua dari YOLO yang lebih dikenal dengan nama YOLO9000 dipublikasikan oleh Joseph Redmon dan Ali Farhadi pada akhir tahun 2016. YOLOv2 menggunakan gambar masukan dengan dimensi 448×448, selain itu arsitektur jaringan yang digunakan adalah 19 *convolutional layers* dan 5 *max pooling layers* serta sebuah *softmax layer* untuk melakukan klasifikasi objek, arsitektur jaringan ini disebut dengan Darknet-19. Pada YOLOv2 *training* dilakukan dengan menggunakan gambar dengan dimensi yang berbeda dan berkisar antara 320×320 hingga 608×608 yang memungkinkan jaringan untuk mempelajari dan memprediksi objek dengan berbagai dimensi dengan lebih akurat. Selain itu salah satu perubahan paling penting dalam YOLOv2 adalah pengenalan *anchor boxes* yang digunakan untuk melakukan prediksi kotak pembatas atau *bounding box* [2].

Perbaikan dari YOLOv2 dinamakan dengan YOLOv3. Pada versi ketiga ini arsitektur jaringan yang digunakan terdiri dari 53 *convolutional layers* yang dikenal dengan nama Darknet-53. Dikatakan bahwa YOLOv3 lebih baik dalam melakukan deteksi dengan skala yang berbeda. Selain itu terdapat prediksi *bounding box* yang memberikan nilai objek pada setiap *bounding box* [3].

2.2 Pengenalan Karakter

Pengenalan karakter dapat disebut juga sebagai OCR (*Optical Character Recognition*). Sisem pengenalan karakter termasuk pada pengenalan pola. Sistem ini dapat digunakan untuk mengenali tulisan atau karakter teks yang terdapat pada sebuah citra. Pengertian pengenalan pola sendiri merupakan suatu sistem yang mencoba untuk membaca atau mengenali apakah citra masukan yang diterima cocok dengan salah satu citra yang telah ditentukan. Pengaplikasian pengenalan pola dapat ditemukan pada sistem pendeteksi sidik jari, tulisan, tanda tangan, bahkan wajah seseorang [4].

Sistem pengenalan karakter teks memungkinkan komputer dapat melakukan sebuah tugas yang bisa dilakukan manusia yaitu

membaca teks. Pengenalan karakter teks mulai dikembangkan pada awal tahun 1950-an. Para peneliti mencoba untuk membaca citra yang terdapat tulisan atau karakter teks. Pada mulanya, proses pengenalan karakter berjalan lambat. Kemudian, kemajuan teknologi digital membuat proses pengenalan karakter menjadi semakin cepat [7].

Tujuan dari pengenalan karakter adalah untuk mendukung perkembangan teknologi yang semakin pesat dalam bentuk digital. Sehingga bila terdapat suatu data fisik yang dikehendaki menjadi bentuk digital, maka sistem pengenal karakter ini dapat dimanfaatkan [7].

Mesin OCR cenderung membuat banyak kesalahan dalam pengenalan karakter ketika kualitas citra rendah. Hal ini disebabkan baik oleh variasi luas tipe karakter dalam citra tersebut. Agar kemampuan pengenalan karakter dapat berjalan baik, maka citra masukan harus berkualitas baik. Ketika citra masukan berkualitas baik, maka pengenalan karakter dapat berjalan dengan baik. Kualitas citra yang baik ditentukan oleh komposisi susunan karakter pada citra dan besar atau kecilnya noise yang terdapat pada citra.

Secara umum, prinsip kerja dari aplikasi OCR adalah sebagai berikut:

1. *Input* berupa citra yang berisi karakter teks yang akan dikenali, citra ini bisa berupa foto, hasil *scan* dokumen, dan lain-lain.

2. *File* citra tersebut diproses menggunakan perangkat lunak aplikasi pengenalan teks. Aplikasi ini melakukan proses pengenalan terhadap karakter yang ada pada *file* citra tersebut.

3. Keluaran dari perangkat lunak aplikasi pengenalan karakter ini berupa data karakter yang sudah siap untuk diolah lebih lanjut.

2.3 Plat Nomor Kendaraan Bermotor

Plat nomor kendaraan bermotor merupakan sebuah bukti registrasi sebuah kendaraan bermotor. Setiap kendaraan bermotor yang digunakan di Indonesia wajib meregistrasikan kendaraannya. Bukti registrasi kendaraan bermotor antara lain dokumen kepemilikan kendaraan bermotor, surat tanda nomor kendaraan bermotor, dan tanda nomor kendaraan bermotor. Tanda nomor kendaraan bermotor disebut juga dengan plat nomor [9].

Plat nomor adalah salah satu jenis identifikasi kendaraan bermotor. Plat nomor juga disebut plat registrasi kendaraan, atau dikenal sebagai plat izin (license plate). Bentuknya berupa potongan plat logam atau plastik yang dipasang pada kendaraan bermotor sebagai identifikasi resmi. Biasanya plat nomor jumlahnya sepasang, untuk dipasang di depan dan belakang kendaraan. Namun ada jenis kendaraan tertentu yang hanya membutuhkan satu plat nomor, biasanya untuk dipasang di bagian belakang [9].

Plat nomor memiliki nomor seri yakni susunan huruf dan angka yang dikhususkan bagi kendaraan tersebut. Nomor ini di Indonesia disebut nomor polisi, dan biasa dipadukan dengan informasi lain mengenai kendaraan bersangkutan, seperti warna, merk, model, tahun pembuatan, nomor identifikasi kendaraan/Vehicle Identification Number (VIN) serta nama dan alamat pemiliknya. Semua data ini juga tertera dalam Surat Tanda Nomor Kendaraan Bermotor (STNK) yang merupakan surat bukti bahwa nomor polisi itu memang ditetapkan bagi kendaraan tersebut [9].

Sebuah plat nomor tersusun didalamnya serangkaian huruf dan angka yang menunjukan kode daerah dan nomor seri kendaraan tersebut. Hal ini menunjukan bahwa setiap plat nomor bersifat unik. Plat nomor kendaraan memiliki warna dasar hitam dan tulisan putih untuk kendaraan pribadi, warna dasar merah dan tulisan putih untuk kendaraan pemerintahan, warna dasar kuning dan tulisan hitam untuk kendaraan umum. Selain terdapat nomor seri yang bersifat unik, sebuah plat nomor juga memiliki informasi lain berupa

bulan dan tahun masa pajak kendaraan berakhir. Tulisan nomor seri dicetak lebih besar daripada tulisan informasi pajak pada plat nomor [9]. Gambar 2.1 menunjukkan ilustrasi plat nomor kendaraan bermotor.

Gambar 2.1. Citra Plat Nomor Kendaraan

2.4 Citra Digital

Citra adalah representasi, kemiripan atau imitasi dari suatu objek atau benda [17]. Secara matematis, citra dinyatakan sebagai suatu fungsi kontinu dari intensitas cahaya pada bidang dua dimensi. Citra dibedakan menjadi dua yaitu citra kontinu diperoleh dari sistem optik yang menerima sinyal analog (mata manusia dan kamera analog) dan citra diskrit (digital) dihasilkan melalui proses digitalisasi terhadap citra kontinu.

2.4.1 Segmentasi Citra

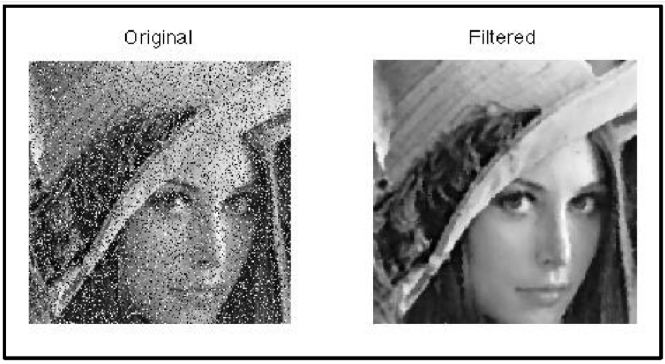
Segmentasi citra merupakan urutan proses yang ditujukan untuk mendapatkan objek-objek yang terkandung didalam citra atau membagi citra kedalam beberapa daerah dengan setiap objek atau daerah memiliki kemiripan atribut. Pada citra yang mengandung hanya satu objek, objek dibedakan dari latar belakangnya. Pada citra yang mengandung sejumlah objek, proses untuk memilah semua objek tentu saja lebih kompleks. Penerapan segmentasi yaitu untuk membuat citra beraras keabuan yang dipisahkan antara *foreground* dan *background* [8].

2.4.2 Derajat Keabuan Citra

Sesuai dengan nama yang melekat, citra jenis ini menangani gradasi warna hitam dan putih, yang menghasilkan efek warna abu-abu. Pada jenis Gambar ini, warna dinyatakan dengan intensitas. Dalam hal ini, intensitas berkisar antara 0 sampai dengan 255. Nilai 0 menyatakan hitam dan nilai 255 menyatakan putih [4].

2.4.3 *Mean Filtering Image*

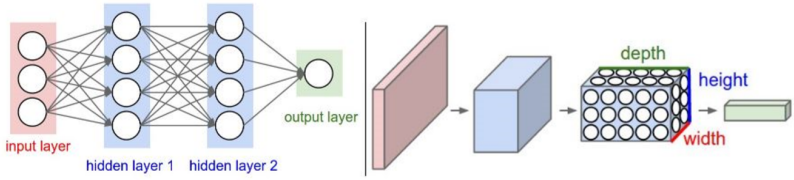
Mean filter digunakan untuk melakukan penghalusan pada citra yang memiliki gangguan atau noise. Mean filter adalah mengganti nilai pixel pada posisi koordinat tertentu dengan rata-rata nilai *pixel* tetangga. Luasan jumlah pixel tetangga ditentukan sebagai masking yang berukuran 2×2 piksel, 3×3 piksel, 4×4 piksel, dan seterusnya [4]. Ilustrasi pada Gambar 2.2 menjelaskan hasil penerapan *mean filter* pada sebuah citra.



Gambar 2.2. Proses Filtering pada citra [4]

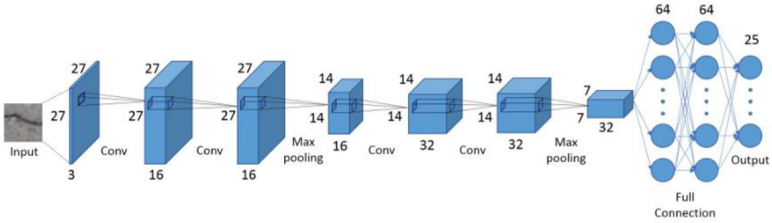
2.5 *Convolutional Neural Network* (CNN)

Metode CNN merupakan pengembangan dari Metode *Multilayer Perceptron* (MLP) yang didesain untuk mengolah data dua dimensi. Seperti yang terlihat pada Gambar 2.3, cara kerja CNN memiliki kesamaan pada MLP, namun dalam CNN setiap neuron dipresentasikan dalam bentuk tiga dimensi, tidak seperti MLP yang setiap neuron hanya berukuran satu dimensi [10].



Gambar 2.3. Dimensi MLP (kiri) dan CNN (kanan)[10]

Terdapat beberapa arsitektur CNN yang umum digunakan. Arsitektur tersebut yaitu LeNet, AlexNet, ZF Net, GoogLeNet, VGGNet dan ResNet. Dapat dilihat pada contoh arsitektur CNN pada Gambar 2.4, CNN terdiri dari tiga jenis layer, yaitu *convolutional layer* (Conv), *pooling layer* (Max pooling) dan *fully-connected layer* (Full Connection). Tumpukan lapisan tersebut membentuk arsitektur dari CNN.

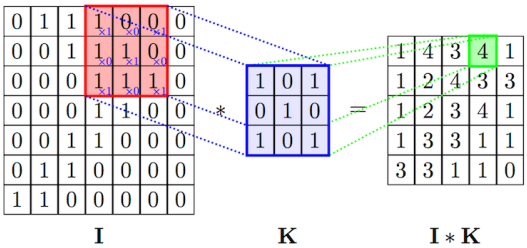


Gambar 2.4. Contoh Arsitektur CNN [10]

Fungsi dasar dari CNN di atas dapat dipecah menjadi empat bidang utama, yaitu [10] :

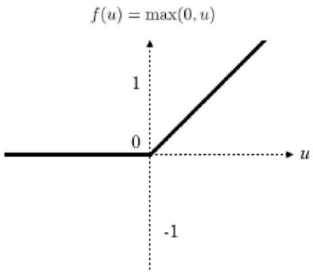
Input layer. Layer yang akan menyimpan nilai piksel dari citra masukan. Pada Gambar 2.4, ukuran dari data input yaitu $27 \times 27 \times 3$, artinya 27×27 merupakan ukuran piksel citra dan 3 merupakan banyak *chanel* citra, yaitu *Red, Green, Blue*.

Convolutional layer. Layer ini akan menentukan keluaran neuron yang terhubung ke *input layer* melalui perhitungan skalar produk antara bobot dan daerah yang terhubung dengan *input*. Ilustrasi dari proses dalam *convolutional layer* dapat dilihat pada Gambar 2.5. Pada ilustrasi tersebut digunakan 1 *zero padding*, yaitu penambahan 1 baris nilai nol disepanjang garis batas input. Dalam proses konvolusi juga digunakan *Rectified Linear Unit* (biasa disingkat menjadi 'ReLu') bertujuan untuk menerapkan fungsi aktivasi *'elementwise'* seperti sigmoid ke *output* dari aktivasi yang dihasilkan oleh lapisan sebelumnya, grafik ReLu, dapat dilihat pada Gambar 2.6.



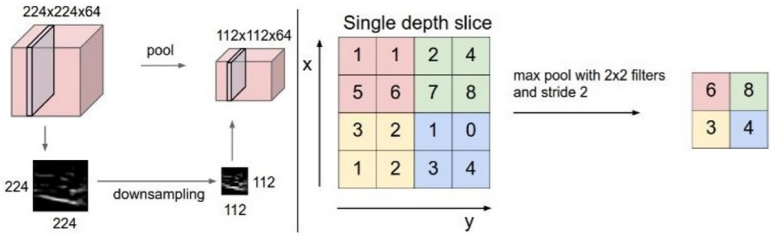
Gambar 2.5. Ilustrasi Perhitungan Convolutional Layer [15]

Pooling layer. Layer ini akan melakukan *downsampling* di sepanjang dimensi spasial dari *input* yang diberikan, selanjutnya mengurangi jumlah parameter dalam aktivasi tersebut. *Pooling layer* mengoperasikan peta aktivasi ke seluruh *input*, dan menggunakan fungsi "MAX". Di sebagian besar CNN, *max-pooling layer* menggunakan kernel dengan dimensi 2×2 dengan *stride* 2 di



Gambar 2.6. *ReLu Activation* [10]

sepanjang dimensi spasial *input*, artinya berpindah sebanyak 2 langkah dalam pergerakan kernelnya. Hal ini mengakibatkan ukuran *input* turun sampai 25% dari ukuran aslinya. Ilustrasi dari proses ini disajikan pada Gambar 2.7. Pada ilustrasi tersebut digunakan kernel dengan dimensi 2×2 dengan *stride* 1.



Gambar 2.7. *Max Pooling* [10]

Fully-connected layer. Layer ini akan melakukan tugas yang sama dengan Jaringan Syaraf standar dan berusaha menghasilkan nilai kelas dari aktivasi, yang digunakan untuk klasifikasi. Lapisan ini sejalan dengan cara neuron yang diatur dalam JST.

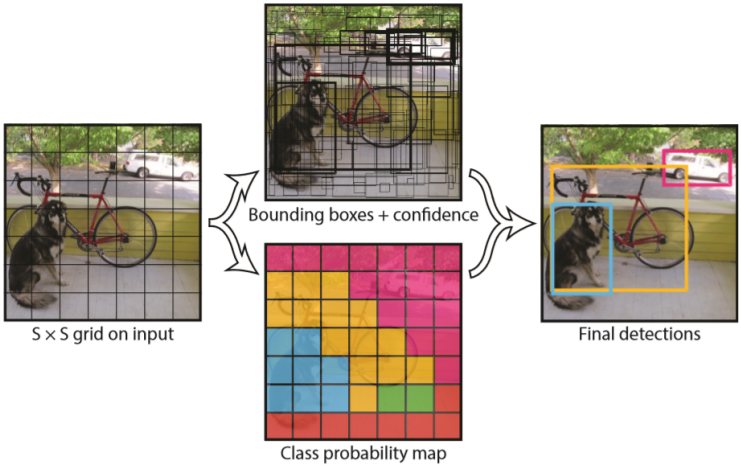
2.6 *You Only Look Once* (YOLO)

YOLO merupakan *Real Object Detection* yang baru-baru ini sangat populer untuk dikembangkan. Kebanyakan sistem deteksi sebelumnya menggunakan pengklasifikasian atau *localizer* untuk melakukan deteksi dengan menerapkan model ke gambar di beberapa lokasi dan skala dan memberi nilai pada gambar sebagai bahan untuk pendeteksian. YOLO menggunakan pendekan yang sangat berbeda dengan metode sebelumnya, yakni menerapkan jaringan syaraf tunggal *(Single Neural Network)* pada keseluruhan gambar. Jaringan ini akan membagi gambar menjadi wilayah-wilayah kemudian memprediksi kotak pembatas dan probabilitas, untuk setiap kotak wilayah pembatas ditimbang probabilitasnya untuk mengklasifikasian sebagai objek atau bukan [13]. Pada Gambar 2.8 dijelaskan ilustrasi deteksi dari metode YOLO.

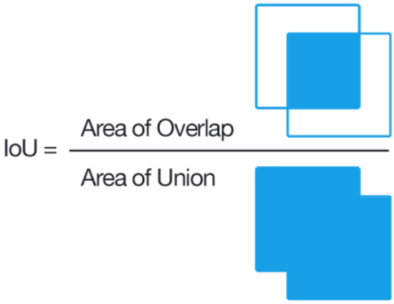
2.6.1 *Intersection over Union* (IoU)

Ketika mendeteksi letak objek, algoritma deteksi objek harus dipertimbangkan. Beberapa algoritma pendeteksian mungkin memerlukan pendeteksian letak objek dengan akurasi tinggi, sementara yang lain lebih toleran terhadap kesalahan dalam penempatan kotak pembatas. Keakuratan kotak biasanya diukur dengan menggunakan *Intersection over Union* (IoU). IoU menghitung area perpotongan antara kotak prediksi objek dan kotak kebenaran dasar (*ground truth*) dan membaginya dengan area gabungan dari mereka. Perumusan IoU tersebut dapat diilustarasikan pada Gambar 2.10.

Saat mengevaluasi algoritma pendeteksian objek, ambang IoU sebesar $0,5$ biasanya digunakan untuk menentukan apakah deteksi benar [15]. Namun, nilai $IoU = 0,5$ mempunyai area yang cukup longgar. Sehingga diingkan nilai IoU yang lebih besar dari $0,5$ [14].



Gambar 2.8. Model sistem yang mendeteksi sebagai permasalahan regresi. Sistem ini membagi gambar menjadi kotak berukuran S x S dan untuk setiap sel kotak ini memprediksi kotak B yang terikat, tingkat kepercayaan, dan probabilitas kelas C. Prediksi ini diformulasikan sebagai berikut $S \times S \times (B * 5 + C)$ tensor [1].



Gambar 2.9. Ilustrasi Perhitungan IoU

2.6.2 *Loss Function*

Untuk sel dengan grid tunggal, metode YOLO memprediksi beberapa kotak pembatas. Untuk menghitung fungsi kerugian, digunakan satu kotak pembatas sebagai representasi dari objek yang lain. Untuk memilih satu di antara kotak pembatas, digunakan nilai IoU tinggi. Kotak dengan IoU tinggi akan menjadi representasi dari objek lain [18]. Berbagai macam fungsi kerugian diantaranya :

1. Fungsi kerugian klasifikasi (*Classification loss function*) Jika suatu objek terdeteksi, kerugian klasifikasi di setiap sel adalah kesalahan kuadrat dari probabilitas bersyarat kelas untuk setiap kelas : YOLO dapat mempelajari menggeneralisasi representasi objek saat dilatih tentang gambar alami dan diuji gambar seni, YOLO mengungguli metode metode pendeteksian terbaik seperti DPM dan R-CNN dengan selisih lebar.

2. Fungsi kerugian lokalisasi (*Localization loss function*) Kerugian lokalisasi diukur dari kesalahan dalam prediksi lokasi dan ukuran kotak pembatas. Disini hanya dihitung kotak yang responsibel untuk mendeteksi objek.

3. Fungsi kerugian kepercayaan (*Confidence loss function*).

2.7 Fungsi Aktivasi

Fungsi aktivasi atau fungsi transfer merupakan fungsi nonliniear yang memungkinkan sebuah jaringan untuk dapat menyelesaikan permasalahan permasalahan non trivial. Setiap fungsi aktivasi mengambil sebuah nilai dan melakukan operasi matematika. Pada arsitektur CNN, fungsi aktivasi terletak pada perhitungan akhir keluaran feature map atau sesudah proses perhitungan konvolusi atau pooling untuk menghasilkan suatu pola fitur. Beberapa macam fungsi aktivasi yang sering digunakan dalam penelitian antara lain fungsi sigmoid, tanh, *Rectified Liniear Unit* (ReLU), Leaky ReLU (LReLU) dan Parametric ReLU [16].

BAB III
METODE PENELITIAN

Pada bab ini akan dijelaskan mengenai metodologi penelitian yang dilakukan untuk menyelesaikan Tugas Akhir ini. Metodologi penelitian berfungsi sebagai acuan sehingga penelitian dapat berjalan sistematis. Adapun secara umum pengerjaan tugas akhir ini dapat ditunjukkan melalui blog diagram pada gambar 3.1.

Gambar 3.1. Diagram metode penelitian

3.1 Studi Literatur

Pada tahap ini dilakukan pengumpulan referensi yang menunjang pengerjaan Tugas Akhir ini. Referensi tersebut dapat berupa

literatur, jurnal ilmiah, tugas akhir dan artikel – artikel lain dari internet yang bersifat relevan mengenai pengenalan plat nomor serta metode You Only Look Once (YOLO) dan penerapannya.

3.2 Pengumpulan Data

Pada tahap ini dilakukan pengumpulan data yang diperlukan dalam pengerjaan tugas akhir ini. Data yang diperlukan dalam tugas akhir ini berupa gambar plat nomor kendaraan bermotor yang didapat dari hasil ekstraksi dan *cropping* video. Adapun video yang digunakan diambil dengan *handphone* dan kamera pada siang hari dengan cuaca cerah.

3.3 Perancangan dan Implementasi Program

Pada tahap ini dilakukan perancangan program yang dibuat untuk menyelesikan Tugas Akhir. program ini dibuat menggunakan OpenCV dan Python, serta tkinter untuk membuat tampilan *Graphical User Interface* (GUI).

Proses utama dalam implementasi pengenalan plat nomor pada Tugas Akhir ini adalah *pre-processing* atau persiapan data dan perancangan model yang akan dijelaskan sebagai berikut:

Tahap *Pre-processing* data dilakukan sebelum proses *training* dan *testing*. Tahap *preprocessing* untuk proses *training* terdiri dari dua proses yaitu proses penambahan filter untuk memperbaiki kualitas citra gambar yang digunakan. Hal ini terjadi karena data citra diperoleh dari proses ekstraksi dan *cropping* video, sehingga berukuran kecil dan sedikit *blur*. Proses selanjutnya yaitu anotasi data. Proses anotasi data yaitu proses memberi keterangan setiap data citra berupa informasi kotak pembatas dan kelas objek pada citra. Sedangkan tahap *pre-processing* untuk proses *testing* tidak hanya melawati proses tersebut, namun juga melewati proses pemberian ganguan pada citra, yaitu *bright adjustment*.

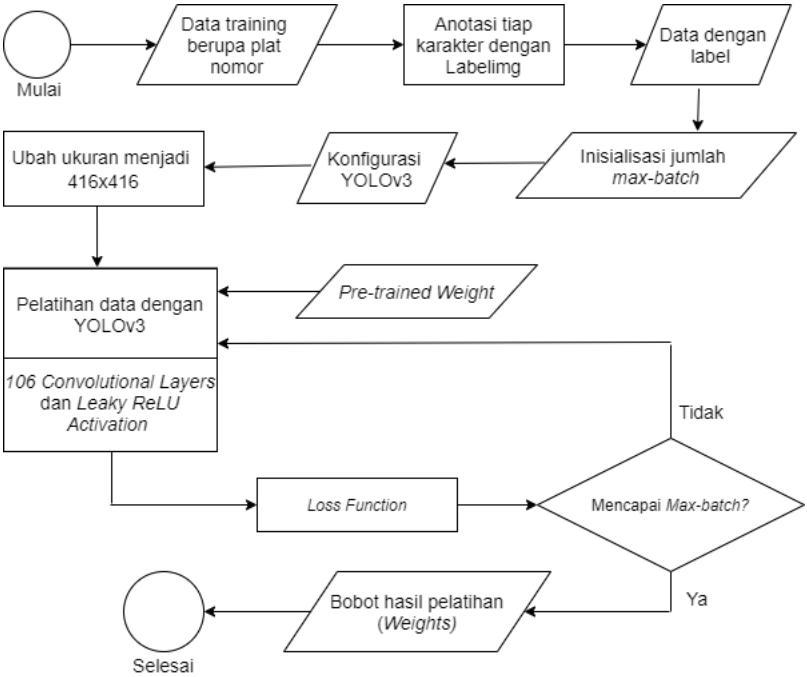
Pada tahap perancangan model, Metode yang digunakan untuk proses pengenalan plat nomor pada penelitian ini adalah

YOLOv3. Langkah langkah yang gunakan akan dijelaskan sebagai berikut :

1. Menginputkan data citra RGB hasil *pre-processing*.
2. Mengubah ukuran citra masukan menjadi 416×416.
3. Mendapatkan data *pre-trained weight* sebagai bobot awal dan model jaringan YOLO.
4. Menyiapkan file konfigurasi YOLOv3.
5. Melakukan proses *training* dengan bobot awal dan model YOLOv3 menggunakan data *training*. Arsitektur YOLOv3 mempunyai arsitektur 106 *convolutional layer* dengan *Leaky ReLU Activation*.
6. Melakukan proses *testing* terhadap data uji dengan menggunakan bobot hasil proses *training*.
7. Menghitung akurasi kinerja metode YOLOv3 dalam mengenali plat nomor.

3.4 *Training* dan *Testing*

Pada tahap ini dilakukan *training* dan uji coba program yang telah dibuat yaitu *training* dan *testing* data. Adapun untuk diagram *training* dapat ditunjukkan melalui blog diagram pada Gambar 3.2. Sementara itu untuk diagram uji coba dapat ditunjukkan melalui blog diagram pada Gambar 3.3. Pada blok diagram ujicoba yang ditampilkan pada Gambar 3.3 terdapat tahapan pengenalan karakter dengan metode YOLO yang lebih jelas akan ditampilkan pada Gambar 3.4.



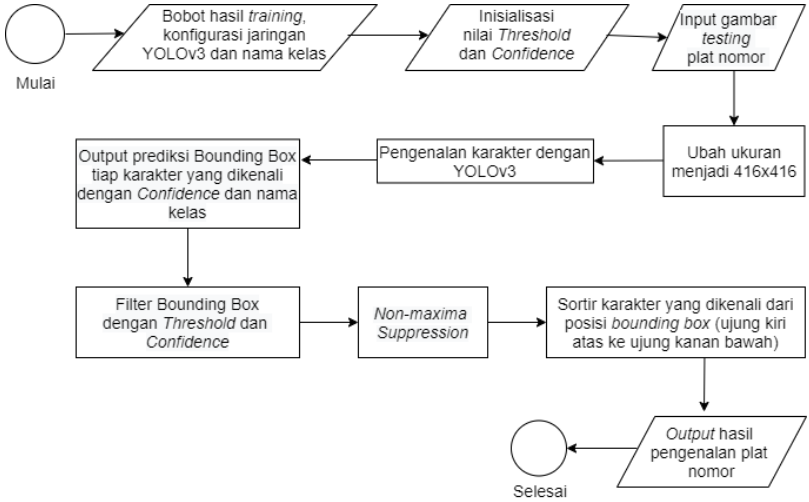
Gambar 3.2. Diagram *training* data

3.5 Penarikan Kesimpulan

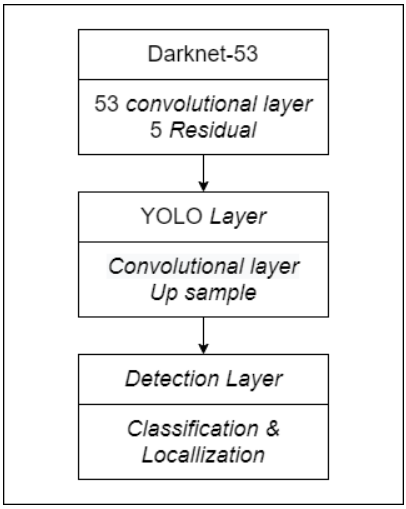
Pada tahap ini, akan dilakukan penarikan kesimpulan dari hasil pembahasan sebelumnya dan juga akan diberikan saran mengenai hal-hal yang dapat dikembangkan untuk penelitian selanjutnya.

3.6 Pembuatan Laporan

Bagian terakhir dalam Tugas Akhir ini adalah pembuatan laporan seluruh tahapan atau proses yang sudah dilakukan.



Gambar 3.3. Diagram *testing*



Gambar 3.4. Pengenalan karakter dengan YOLO

BAB IV
PERANCANGAN DAN IMPLEMENTASI PROGRAM

Pada bab ini akan dijelaskan rancangan program untuk mengenali plat nomor menggunakan metode YOLO-Darknet. Dengan versi YOLO yang dipilih adalah YOLOv3 dan sebagai *feature extractor* digunakan Darknet-53. Kemudian akan dijelaskan implementasi dari rancangan tersebut pada sebuah perangkat lunak untuk mengetahui kinerja dari metode yang digunakan.

4.1 Perancangan Data

Pengenalan objek berbasis citra digital memerlukan data latih untuk mendapat bobot yang akan digunakan untuk mengenali objek. Selain itu, diperlukan juga data untuk proses *testing* untuk menguji keakuratan dari metode. Data yang telah terkumpul dalam penelitian ini berjumlah 170 data. Data tersebut terdiri dari gambar plat nomor yang berasal dari ekstraksi dan *cropping* video. Kemudian akan dibagi dengan perbandingan $85 : 15$ yaitu 85% untuk proses *training* sebanyak 145 data dan 15% untuk proses *testing* yaitu sebanyak 25 data.

4.1.1 Kebutuhan Perangkat Lunak

Untuk mengimplementasikan metode yang digunakan pada pengerjaan tugas akhir ini, diperlukan perangkat keras dan perangkat lunak. Adapun rincian dari perangkat keras dan perangkat lunak yang digunakan akan disajikan pada Tabel 4.1.

4.1.2 Data Awal

Data latih dan data uji yang digunakan dalam penelitian ini terdiri dari gambar plat nomor yang berasal dari hasil ekstraksi dan *cropping* video. Video kendaraan yang sedang melaju diambil secara manual menggunakan *handphone* dan kamera. Video diambil

pada waktu siang hari dengan kondisi cerah. Video diambil di jalur satu arah dengan arah kendaraan mendekati kamera. Dari video yang didapat perlu dilakukan ekstraksi dan *cropping* untuk memisahkan gambar plat nomor dengan *background*. Gambar plat nomor yang didapatkan rata-rata berukuran 80×20 *pixel*. Gambar 4.1 adalah gambar plat nomor yang didapat setelah melakukan ekstraksi dan *cropping* video. Terlihat bahwa ukuran plat nomor yang didapat cukup kecil dan sedikit blur.

Tabel 4.1. Kebutuhan Perancangan

Perangkat Keras	1. Intel Celeron N4000, up to 2.6GHz 2. Memory 4.00GB RAM
Perangkat Lunak	1. Sistem operasi Microsoft Windows 10 Pro 64-bit 2. Python 3.8.1 3. OpenCV 4.1.2.30 4. labelimg 1.7.0 5. Google Collaboration

4.1.3 Data Proses *training*

Untuk melakukan pengenalan karakter pada plat nomor diperlukan sejumlah data yang digunakan untuk proses *training*. Jumlah data latih yang digunakan pada penelitian ini adalah 100 gambar

Gambar 4.1. Citra hasil ekstraksi dan *cropping*

plat nomor hasil deteksi dan *cropping* video. Total jumlah karakter pada plat nomor yang digunakan utuk proses *training* ini berjumlah 801 buah. Proses *training* dilakukan dua kali. Yaitu *training* dengan menggunakan data asli tanpa melakukan *preprocessing* dan *training* dengan data yang dilakukan proses *preprocessing* terlebih dahulu. *Preprocessing* yang dilakukan adalah menerapkan filter untuk memperbaiki kualitas gambar plat nomor.

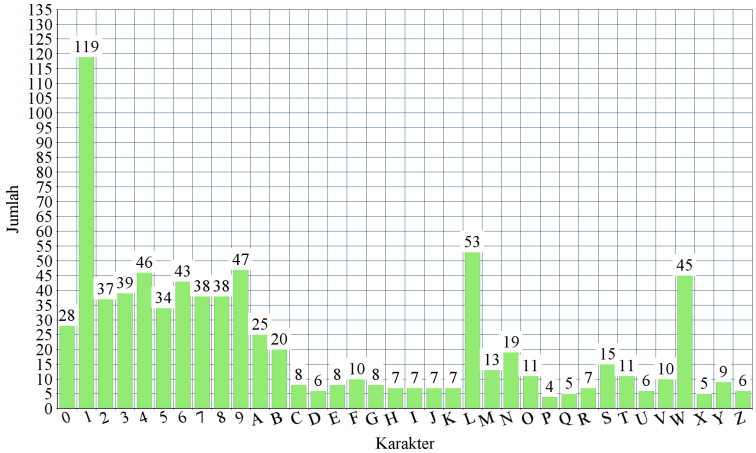
Kemudian dilanjutkan dengan proses anotasi data atau pemberian label pada data. Jumlah label kelas yang digunakan adalah 36 yaitu terdiri dari huruf (A-Z) dan angka $(0 - 9)$. 36 kelas dipilih karena sistem yang akan dibangun bertujuan untuk mengenali plat nomor dan plat nomor terdiri dari 26 karakter huruf besar dan 10 angka. *Preprocessing* data lebih lanjut akan dijelaskan pada subbab 4.2.1. Adapun rincian dari karakter pada data yang digunakan pada saat proses *training* dapat dilihat pada Tabel 4.2. Dan untuk mengetahui sebaran data latih yang digunakan akan ditampilkan grafik pada Gambar 4.2.

4.1.4 Data Proses *testing*

Sebanyak 25 data baru (data lain yang tidak digunakan untuk proses *training*) digunakan untuk proses *testing*. Data tersebut

Tabel 4.2. Data proses *training*

JENIS KARAKTER	JUMLAH	JENIS KARAKTER	JUMLAH
0	28	I	7
1	119	J	7
2	37	K	7
3	39	L	53
4	46	M	13
5	34	N	19
6	43	O	11
7	38	P	4
8	38	Q	5
9	47	R	7
A	25	S	15
B	20	T	11
C	8	U	6
D	6	V	10
E	8	W	45
F	10	X	5
G	8	Y	9
H	7	Z	6



Gambar 4.2. Sebaran data latih

terdiri dari dataset tanpa gangguan dan dataset dengan gangguan. Terdapat 2 jenis gangguan yang diberikan pada dataset, yaitu gangguan dengan penambahan intensitas warna dan pengurangan intensitas warna pada citra. Gangguan diberikan dalam beberapa tingkatan. Hal ini dilakukan sebagai antisipasi ketika data yang diambil mengalami gangguan pencahayaan seperti terlalu gelap atau terlalu terang. Dataset tanpa gangguan yaitu dataset yang berisi 25 data citra dan keseluruhan datanya tidak diberi gangguan. Namun perlu diketahui bahwa data uji yang digunakan untuk menguji tiap model adalah berbeda. Untuk model yang didapat dari hasil *training* menggunakan data tanpa dilakukan *preprocessing* maka data uji yang digunakan juga tanpa dilakukan *preprocessing* terlebih dahulu. Sedangkan untuk model yang didapat dari hasil *training* menggunakan data dengan dilakukan *preprocessing* terlebih dahulu,

maka data uji yang digunakan juga dilakukan proses *preprocessing* terlebih dahulu. Hal ini dilakukan untuk mengetahui keakuratan model dalam mendeksi pola data *training*. Tabel 4.3 dan 4.4 menunjukkan rincian dari dataset yang digunakan saat proses *testing*. Dapat dilihat pada tabel tersebut, total datset yang digunakan adalah 7 dataset.

Tabel 4.3. Dataset proses *testing* untuk model tanpa *preprocessing*

Jenis Dataset	Parameter	Total
Citra tanpa gangguan	-	25
Citra gangguan terang	Intensitas warna (n) = 25	25
	Intensitas warna (n) = 50	25
	Intensitas warna (n) = 75	25
Citra gangguan gelap	Intensitas warna (n) = -25	25
	Intensitas warna (n) = -50	25
	Intensitas warna (n) = -75	25

4.2 Perancangan Proses

Secara umum proses pengenalan plat nomor dengan metode YOLO-Darknet dapat ditunjukkan dalam diagram alir yang telah disajikan dalam Gambar 3.2 dan Gambar 3.3. Adapun rincian tentang perancangan proses pengenalan plat nomor akan dijelaskan pada subbab ini. Secara garis besar, tahapan pengerjaannya adalah sebagai berikut :

1. Mempersiapkan data baik untuk proses *training* maupun *testing*.
2. Melakukan proses pelabelan atau anotasi pada data.
3. Mempersiapkan hal-hal yang dibutuhkan dalam proses *training*. Diantaranya yaitu memenuhi kebutuhan penggunaan

Tabel 4.4. Dataset proses *testing* untuk model dengan *preprocessing*

Jenis Dataset	Parameter	Total
Citra tanpa gangguan dengan *preprocessing*	-	25
Citra gangguan terang dengan *preprocessing*	Intensitas warna (n) = 25	25
	Intensitas warna (n) = 50	25
	Intensitas warna (n) = 75	25
Citra gangguan gelap dengan *preprocessing*	Intensitas warna (n) = -25	25
	Intensitas warna (n) = -50	25
	Intensitas warna (n) = -75	25

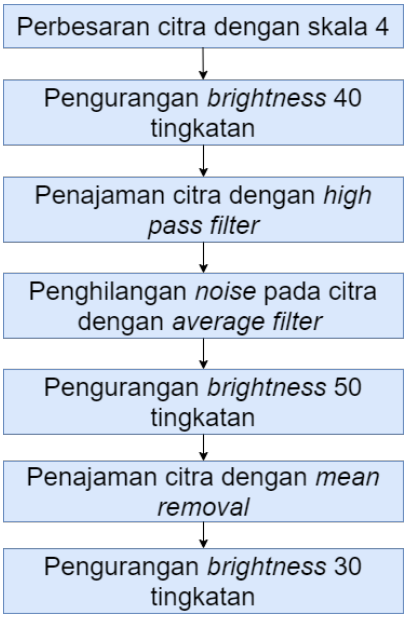
perangkat lunak dan menyiapkan data yang digunakan untuk proses *training*.

4. Melakukan proses *training*
5. Mempersiapkan hal-hal yang dibutuhkan untuk proses *testing*. Diantaranya yaitu mempersiapkan dataset *testing* dan bobot yang didapatkan dari hasil *training*.
6. Melakukan proses *testing*.
7. Melakukan uji keakuratan.

4.2.1 *Preprocessing* Data

Data yang didapatkan dari hasil ekstraksi dan *cropping* video memiliki kualitas yang tidak terlalu baik seperti ukuran citra yang terlalu kecil dan terdapat efek *blur* pada citra. Karena itu perlu dilakukan *preprocessing* data untuk memperbaiki kualitas citra. Tahap *preprocessing* data ini terdiri dari beberapa tahapan diantaranya

adalah *resize* yaitu mengubah ukuran data agar tidak terlalu kecil, penajaman citra, penghilangan *noise*, dan pengurangan intensitas warna. Gambar 4.3 menunjukkan tahapan-tahapan yang dilakukan pada saat *preprocessing* data.



Gambar 4.3. Tahapan *preprocessing* data

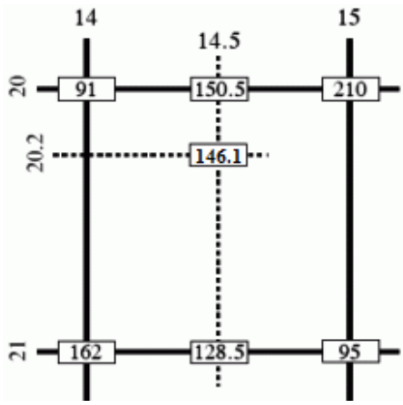
1. ***Resize* Data**

   Pada tahap ini dilakukan perbesaran ukuran citra. Citra yang mula-mula berukuran 80×20 *pixel* diperbesar dengan skala 400% sehingga ukuran citra menjadi 320×80 *pixel*. Perbesaran ukuran citra dilakukan dengan metode interpolasi bilinear. Interpolasi bilinear berarti menerapkan interpolasi linear dalam dua arah. Yang berarti menggunakan nilai 4 tetangga terdekat dan menghitung rata-rata tertimbang untuk meng-

hasilkan output. Adapun persamaan interpolasi bilinear untuk melakukan perbesaran citra ditunjukkan pada persamaan 4.1 [20]. Misalkan kita ingin mencari nilai dari fungsi f pada titik (x, y) dimana nilai fungsi f dalam hal ini adalah intensitas citra. Diasumsikan bahwa kita mengetahui nilai f pada empat titik yaitu $Q_{11} = (x_1, y_1)$, $Q_{12} = (x_1, y_2)$, $Q_{21} = (x_2, y_1)$, $Q_{22} = (x_2, y_2)$.

$$f(x, y_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}),$$
$$f(x, y_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}).$$

$$f(x, y) \approx \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2) \tag{4.1}$$



Gambar 4.4. Ilustrasi perbesaran citra

Sementara itu ilustrasi perhitungan dari gambar 4.4 dengan menggunakan persamaan 4.1 adalah sebagai berikut.

$$I_{20,14.5} = \frac{15-14.5}{15-14} \cdot 91 + \frac{14.5-14}{15-14} \cdot 210 = 150.5,$$
$$I_{21,14.5} = \frac{15-14.5}{15-14} \cdot 162 + \frac{14.5-14}{15-14} \cdot 95 = 128.5,$$

$$I_{20.2,14.5} = \frac{21-20.2}{21-20} \cdot 150.5 + \frac{20.2-20}{21-20} \cdot 128.5 = 146.1.$$

2. **Pengurangan intensitas warna (*brightness*)**

   Pada tahap ini dilakukan pengurangan intensitas warna pada citra sebesar 40 tingkatan. Hal ini dilakukan untuk memperjelas karakter pada plat nomor karena tepi dari objek karakter memiliki intensitas warna yang rendah akibat efek blur. Hal ini terjadi karena citra didapat dari proses ekstraksi dan *cropping*. Dengan dilakukannya pengurangan intensitas warna, tepi karakter yang *blur* akan sedikit memudar dan menyatu dengan warna *background*. Pada tahap ini karena citra terdiri dari 3 *channel* yaitu *red, green, blue* maka pengurangan intensitas warna juga dilakukan terhadap masing-masing *channel* RGB. Sementara itu untuk ilustrasi perhitungan adalah sebagai berikut : Diasumsikan bahwa matriks A adalah sub bagian citra pada salah satu *channel* dengan ukuran 5×5. Dan matriks B adalah hasil pengurangan intensitas warna pada matriks A.

$$A = \begin{bmatrix} 100 & 120 & 190 & 71 & 201 \\ 166 & 211 & 15 & 123 & 111 \\ 214 & 188 & 138 & 51 & 71 \\ 198 & 61 & 34 & 22 & 180 \\ 120 & 13 & 221 & 41 & 113 \end{bmatrix}$$

$$B = A - 40$$

$$B = \begin{bmatrix} 60 & 80 & 150 & 31 & 161 \\ 126 & 181 & 0 & 83 & 71 \\ 174 & 148 & 98 & 11 & 31 \\ 158 & 21 & 0 & 0 & 140 \\ 80 & 0 & 181 & 1 & 73 \end{bmatrix}$$

3. **Penajaman citra (*high pass filtering*)**

   Pada tahap selanjutnya dilakukan penajaman citra. Tahap ini dilakukan untuk membuat tepi objek pada citra lebih terlihat, namun akibat diterapkannya filter ini adalah *noise* pada citra menjadi lebih terlihat. Penajaman citra dilakukan dengan cara mengkonvolusikan *pixel* citra dengan matriks kernel. Matrriks kernel adalah matriks persegi yang berukuran kecil dengan jumlah baris dan kolom berukuran ganjil yang berisi bobot sederhana. Adapun matriks kernel yang digunakan untuk melakukan penajaman citra ditunjukkan pada Gambar 4.5. Matriks B adalah matriks yang diperoleh pada perhitungan di

$$K = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 10 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Gambar 4.5. Kernel penajaman citra

tahap sebelumnya. Hasil perhitungan setelah dikonvolusikan

dengan kernel pada Gambar 4.5 disimpan dalam matriks C.

$$B = \begin{bmatrix} 60 & 80 & 150 & 31 & 161 \\ 126 & 181 & 0 & 83 & 71 \\ 174 & 148 & 98 & 11 & 31 \\ 158 & 21 & 0 & 0 & 140 \\ 80 & 0 & 181 & 1 & 73 \end{bmatrix}$$

$$C = B * K$$

$$C = \begin{bmatrix} 213 & 255 & 255 & 0 & 255 \\ 255 & 255 & 0 & 255 & 255 \\ 255 & 255 & 255 & 0 & 5 \\ 255 & 0 & 0 & 0 & 255 \\ 255 & 0 & 255 & 0 & 255 \end{bmatrix}$$

4. **Penghilangan *Noise* dengan *average filter***

   Pada tahap selanjutnya dilakukan penghilangan *noise* pada citra. Tahap ini dilakukan untuk menghilangkan *noise* pada citra plat nomor yang dihasilkan oleh proses sebelumnya. Sehingga didapatkan citra tanpa *noise* namun sedikit *blur*. Penghilangan *noise* pada citra dilakukan dengan cara mengkonvolusikan *pixel* citra dengan matriks kernel. Adapun matriks kernel yang digunakan untuk melakukan pengaburan citra ditunjukkan pada Gambar 4.6. Matriks C adalah matriks yang diperoleh pada perhitungan di tahap penajaman citra. Hasil perhitungan penghilangan *noise* pada citra dilakukan dengan mengkonvolusikan matriks C dengan kernel pada Gambar

$$K_2 = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Gambar 4.6. Kernel penghilangan *noise* pada citra

4.6. Hasil perhitungan disimpan dalam matriks D.

$$C = \begin{bmatrix} 213 & 255 & 255 & 0 & 255 \\ 255 & 255 & 0 & 255 & 255 \\ 255 & 255 & 255 & 0 & 5 \\ 255 & 0 & 0 & 0 & 255 \\ 255 & 0 & 255 & 0 & 255 \end{bmatrix}$$

$$D = C * K_2$$

$$D = \begin{bmatrix} 109 & 137 & 11311385 & & \\ 165 & 222 & 170 & 142 & 86 \\ 142 & 170 & 113 & 114 & 86 \\ 113 & 170 & 85 & 114 & 57 \\ 57 & 85 & 28 & 85 & 57 \end{bmatrix}$$

5. **Pengurangan intensitas warna (*brightness*)**

   Pada tahap ini kembali dilakukan pengurangan intensitas warna sebesar 50 *pixel*. Hasil perhitungan akan ditampilkan

pada matriks E.

$$D = \begin{bmatrix} 109 & 137 & 11311385 & & \\ 165 & 222 & 170 & 142 & 86 \\ 142 & 170 & 113 & 114 & 86 \\ 113 & 170 & 85 & 114 & 57 \\ 57 & 85 & 28 & 85 & 57 \end{bmatrix}$$

$$E = D - 50$$

$$E = \begin{bmatrix} 59 & 87 & 63 & 63 & 35 \\ 115 & 172 & 120 & 92 & 36 \\ 92 & 120 & 63 & 64 & 36 \\ 63 & 120 & 35 & 64 & 7 \\ 7 & 35 & 0 & 35 & 7 \end{bmatrix}$$

6. **Penajaman citra dengan metode *Mean removal***

   Pada tahap selanjutnya kembali dilakukan penajaman citra dengan metode *Mean removal*. Tahap ini dilakukan dengan tujuan menajamkan kembali citra yang didapat dari tahapan sebelumnya. Penajaman citra dilakukan dengan cara mengkonvolusikan *pixel* citra dengan matriks kernel. Adapun matriks kernel yang digunakan untuk melakukan penajaman citra dengan metode *Mean removal* ditunjukkan pada Gambar 4.7. Matriks E adalah matriks yang diperoleh pada perhitungan di tahap sebelumnya. Hasil perhitungan setelah dikonvolusikan dengan kernel pada Gambar 4.5 disimpan dalam matriks F.

$$E = \begin{bmatrix} 59 & 87 & 63 & 63 & 35 \\ 115 & 172 & 120 & 92 & 36 \\ 92 & 120 & 63 & 64 & 36 \\ 63 & 120 & 35 & 64 & 7 \\ 7 & 35 & 0 & 35 & 7 \end{bmatrix}$$

$$K_3 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Gambar 4.7. Kernel penajaman citra

$$F = E * K_3$$

$$F = \begin{bmatrix} 157 & 254 & 33 & 221 & 124 \\ 255 & 255 & 255 & 255 & 34 \\ 238 & 255 & 0 & 123 & 61 \\ 193 & 255 & 0 & 255 & 0 \\ 0 & 90 & 0 & 202 & 0 \end{bmatrix}$$

7. **Pengurangan intensitas warna (*brightness*)**

   Pada tahap ini kembali dilakukan pengurangan intensitas warna sebesar 30 *pixel*. Hasil perhitungan akan ditampilkan pada matriks G.

$$F = \begin{bmatrix} 157 & 254 & 33 & 221 & 124 \\ 255 & 255 & 255 & 255 & 34 \\ 238 & 255 & 0 & 123 & 61 \\ 193 & 255 & 0 & 255 & 0 \\ 0 & 90 & 0 & 202 & 0 \end{bmatrix}$$

$$G = F - 30$$

$$G = \begin{bmatrix} 127 & 224 & 3 & 191 & 94 \\ 225 & 225 & 225 & 225 & 4 \\ 208 & 225 & 0 & 93 & 31 \\ 163 & 225 & 0 & 225 & 0 \\ 0 & 60 & 0 & 172 & 0 \end{bmatrix}$$

4.2.2 Anotasi Data

Untuk membangun sistem yang dapat mengenali plat nomor, akan dibangun sistem yang dapat mengenali karakter pada plat nomor terlebih dahulu, oleh karena itu keterangan karakter pada setiap gambar plat nomor perlu dibuat. Keterangan yang dimaksud adalah informasi tentang kelas dari objek yang akan dikenali. Jumlah kelas yang digunakan adalah 36 yaitu terdiri dari huruf $(A - Z)$ dan angka $(0 - 9)$. Keterangan ini berfungsi sebagai target atau acuan untuk memperoleh bobot pada saat proses *training* data dan menjadi pembanding nilai *output* pada saat proses *testing*. Proses ini disebut dengan anotasi data atau pelabelan.

Proses anotasi data dimulai dengan menggambar kotak pembatas di setiap objek pada citra yang dalam hal ini adalah karakter, lalu menyimpan keterangan kotak pembatas tersebut dalam suatu file. Isi yang disimpan pada file adalah nilai $c, (x, y), (w, h)$ berturut-turut adalah kelas objek, koordinat titik pusat kotak pembatas, dan ukuran dimensi dari kotak pembatas. Perlu diketahui bahwa koordinat (0,0) pada gambar terletak di pojok kiri atas.

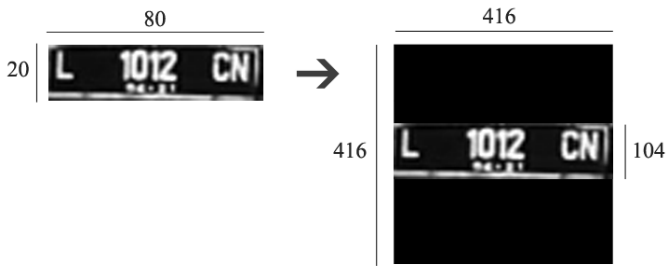
Hal yang perlu diperhatikan karena dataset berupa gambar plat nomor yang dikumpulkan memiliki dimensi yang berbeda-beda, sedangkan pada jaringan YOLOv3 *input* gambar yang digunakan berukuran 416×416. Oleh sebab itu, keterangan kotak pembatas yang telah dibuat pada anotasi data ini juga akan berubah. Untuk menangani hal ini, keterangan kotak pembatas yang telah dibuat akan dibandingkan dengan dimensi citra ukuran asli, hal ini bertujuan untuk menjaga keterangan kotak pembatas dalam dimensi

citra yang berbeda sesuai dengan proporsinya.

Proses menggambar kotak pembatas dilakukan secara akurat dan konsisten karena nilai-nilai yang dihasilkan dari pembentukan kotak pembatas nantinya akan digunakan sebagai target untuk proses *training*.

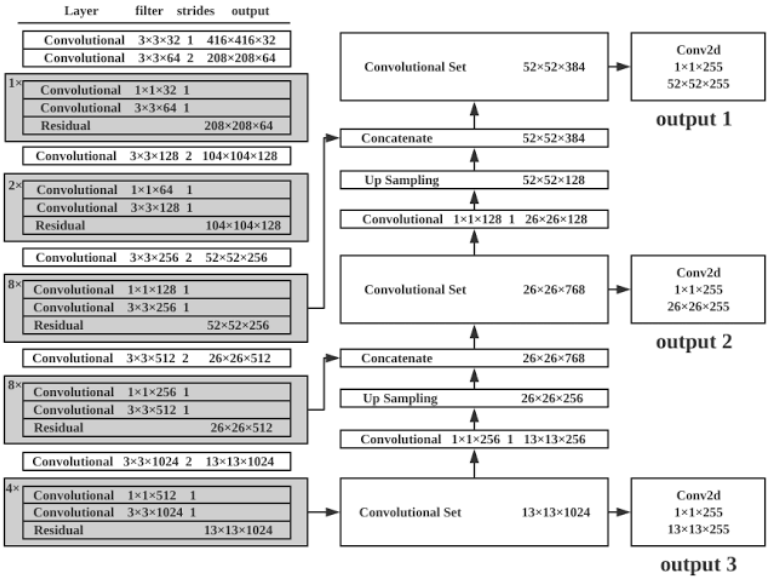
4.2.3 Pengenalan Karakter pada Plat Nomor dengan YOLOv3

Sebelum *training* data dilakukan, hal pertama yang perlu diperhatikan adalah proses *input* gambar, setiap gambar yang masuk akan dilakukan *resizing* menjadi ukuran 416×416. Untuk melakukan *resizing* citra dilakukan dengan menggunakan interpolasi bilinear. Akurasi pendeteksian semakin baik apabila gambar sudah berukuran 416×416. Jika gambar memiliki dimensi yang berbeda maka gambar tersebut akan dilakukan *resizing* dengan ukuran 416×416 dengan tetap menjaga proporsi gambar. Karena gambar masukan memiliki dimensi (80×20) maka hasil *resizing* dari gambar ini adalah (416×104). Karena hasil *resizing* gambar tersebut masih belum sesuai maka dari itu selanjutnya dilakukan *zero padding* pada pinggir citra yang masih kekurangan piksel sehingga hasil dimensi tersebut adalah 416×416. Adapun ilustrasinya akan ditampilkan pada Gambar 4.8.



Gambar 4.8. Proses *input* gambar

YOLOv3 memiliki 106 lapisan *Convolutional layers* yang mendasari arsitektur diikuti *Leaky ReLU activation* dan *batch normalize*, juga terdapat *shortcut*, *route*, dan *upsample*. Adapun arsitektur jaringan diilustrasikan pada Gambar 4.9.



Gambar 4.9. Struktur jaringan [3]

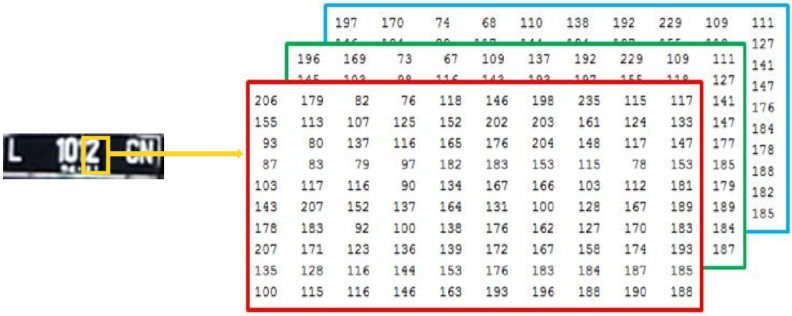
Gambar 4.10 menampilkan tahapan awal dari jaringan YOLOv3 terlihat dari gambar tersebut dilakukan *Convolutional layers* dengan ukuran kernel 3×3, adapun filter yang digunakan adalah 32 dan *stride* yang digunakan adalah 1. Kemudian dilanjutkan dengan menerapkan *Leaky ReLU activation* dan *batch normalize*.

1. ***Convolutional layers***

   Sebagai ilustrasi perhitungan, diberikan contoh kasus jaringan yang digunakan melalui Gambar 4.11. Pada gambar tersebut terdapat tiga lapis matriks berukuran 10×10 yang merupakan

Gambar 4.10. Tahap awal jaringan

representasi sub-bagian dari citra masukkan. Terdapat tiga lapis layer yang berarti citra tersebut mempunyai tiga *channel* warna, yaitu *red, green, blue*.



Gambar 4.11. Ilustrasi Representasi Citra Awal

Selanjutnya ditambahkan *padding*, *padding* yang digunakan adalah 1. Artinya, matriks representasi citra tersebut diberikan *zero-padding* pada matriks terluar sebanyak 1 lapis. Ilustrasi hasilnya dapat dilihat pada Gambar 4.12.

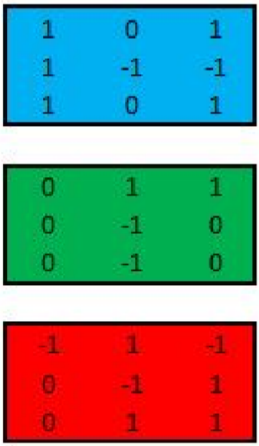
Lapis filter yang digunakan adalah 32 lapis. Salah satu contoh

0	0	0	0	0	0	0	0	0	0	0	0
0	206	179	82	76	118	146	198	235	115	117	0
0	155	113	107	125	152	202	203	161	124	133	0
0	93	80	137	116	165	176	204	148	117	147	0
0	87	83	79	97	182	183	153	115	78	153	0
0	103	117	116	90	134	167	166	103	112	181	0
0	143	207	152	137	164	131	100	128	167	189	0
0	178	183	92	100	138	176	162	127	170	183	0
0	207	171	123	136	139	172	167	158	174	193	0
0	135	128	116	144	153	176	183	184	187	185	0
0	100	115	116	146	163	193	196	188	190	188	0
0	0	0	0	0	0	0	0	0	0	0	0

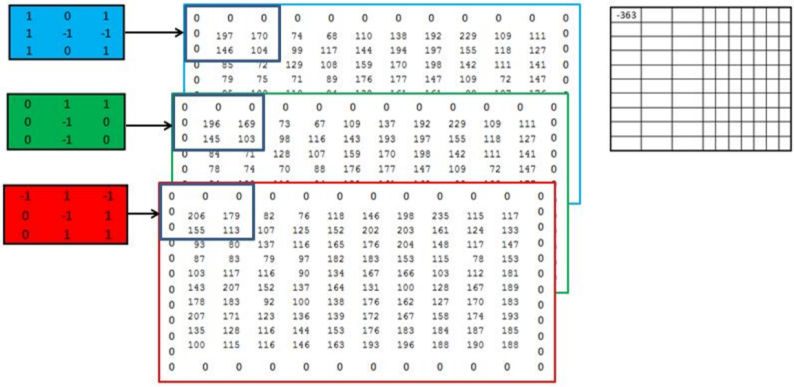
Gambar 4.12. Hasil padding

filter yang digunakan untuk 3 *channel* warna pada matriks representasi citra diberikan pada Gambar 4.13. Selanjutnya setiap layer *channel* warna dikonvolusikan terhadap setiap kernel yang bersesuaian berukuran 3×3. Contoh perhitungan konvolusi untuk titik $(0, 0)$ pada matriks hasil konvolusi R, dengan kernel K dan matriks representasi citra I sebagai berikut.

$$
\begin{aligned}
R(0,0) &= (K(0,0,0) * I(0,0,0)) + (K(0,1,0) * I(0,1,0)) \\
&+ ... + (K(2,2,0) * I(2,2,0)) + (K(0,0,1) * I(0,0,1)) \\
&+ (K(0,1,1) * I(0,1,1)) + ... + (K(2,2,1) * I(2,2,1)) \\
&+ (K(0,0,2) * I(0,0,2)) + (K(0,1,2) * I(0,1,2)) \\
&+ ... + (K(2,2,2) * I(2,2,2)) \\
&= (-1 * 0) + (1 * 0) + ... + (1 * 113) + (0 * 0) + (1 * 0) \\
&+ ... + (1 * 103) + (1 * 0) + (0 * 0) + ... + (1 * 104) \\
&= (-263) + (-314) + 241 \\
&= -363
\end{aligned}
$$

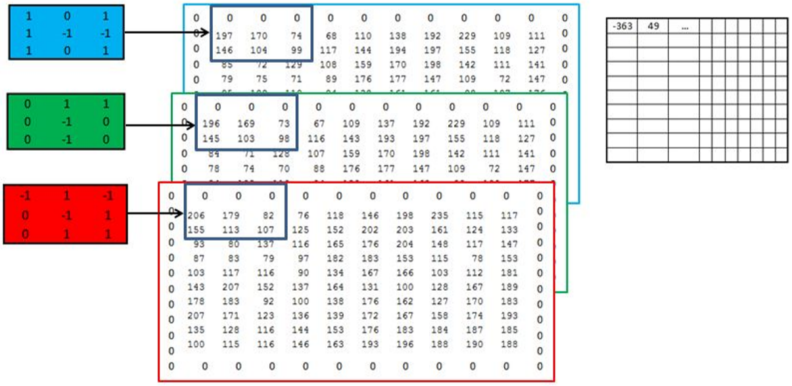


Gambar 4.13. Contoh kernel



Gambar 4.14. Hasil konvolusi kolom 1

Ilustrasi perhitungan $R(0,0)$ ditunjukkan pada Gambar 4.14, dan untuk $R(0,1)$ ditunjukkan pada Gambar 4.15. Proses tersebut dilakukan terus menerus sampai seluruh nilai pada



Gambar 4.15. Hasil konvolusi kolom 2

matriks representasi citra terisi dan menjadi satu layer. Proses tersebut dilakukan sebanyak 32 kali. Hasilnya yaitu ukuran sub-bagian citra menjadi berukuran $10 \times 10 \times 32$.

2. ***Batch Normalization***

   Pada tahap berikutnya adalah menerapkan proses *Batch Normalization* yang berfungsi dalam mempercepat *training Deep Neural Network*. Lapisan batch normalization digunakan untuk menormalkan aktivasi sebuah volume input sebelum meneruskannya ke lapisan berikutnya dalam jaringan. *Batch Normalization* telah terbukti sangat efektif dalam mengurangi jumlah *epoch* yang dibutuhkan untuk melatih jaringan syaraf. Adapun persamaan yang digunakan untuk menghitung *Batch Normalization* ditampilkan pada Gambar 4.16. Sedangkan ilustrasi dari *Batch Normalization* ditampilkan pada Gambar 4.17.

   Sebagai ilustrasi perhitungan pertama-tama sub bagian citra masukan dengan 3 *channel* RGB direpresentasikan berturut-

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

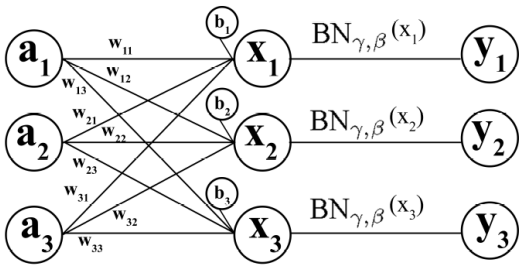
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Gambar 4.16. Algoritma *Batch Normalization* [19]



Gambar 4.17. Ilustrasi *Batch Normalization*

urut sebagai berikut :

$$a_1 = \begin{bmatrix} 6 & 8 & 5 \\ 0 & 9 & 0 \\ 6 & 7 & 1 \end{bmatrix}, a_2 = \begin{bmatrix} 1 & 6 & 5 \\ 2 & 2 & 4 \\ 1 & 5 & 1 \end{bmatrix}, a_3 = \begin{bmatrix} 0 & 4 & 8 \\ 8 & 0 & 1 \\ 5 & 1 & 9 \end{bmatrix}$$

Sedangkan Weights dan Bias yang digunakan adalah :

$$w_{11} = w_{32} = w_{23} \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$w_{21} = w_{12} = w_{33} \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}$$

$$w_{31} = w22 = w_{13} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

$$b_1 = b_2 = b_3 = 1$$

Agar ukuran spasial dari output yang dihasilkan tetap sama dengan ukuran spasial input, maka inputan a_1, a_2, dan a_3 dilakukan *Zero Padding* sebanyak 1 sehingga menghasilkan :

$$a1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 8 & 5 & 0 \\ 0 & 0 & 9 & 0 & 0 \\ 0 & 6 & 7 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, a2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 6 & 5 & 0 \\ 0 & 2 & 2 & 4 & 0 \\ 0 & 1 & 5 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$, a3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 8 & 0 \\ 0 & 8 & 0 & 1 & 0 \\ 0 & 5 & 1 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$x_i = \sum_{n=1}^{3} (a_n * w_n^i + b^i) \tag{4.2}$$

Dengan menggunakan persamaan 4.2 didapatkan :

$$x_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 8 & 5 & 0 \\ 0 & 0 & 9 & 0 & 0 \\ 0 & 6 & 7 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 6 & 5 & 0 \\ 0 & 2 & 2 & 4 & 0 \\ 0 & 1 & 5 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$* \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 8 & 0 \\ 0 & 8 & 0 & 1 & 0 \\ 0 & 5 & 1 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

$$+ \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$x_1 = \begin{bmatrix} 16 & 12 & 12 \\ -21 & 21 & -15 \\ 17 & 12 & -3 \end{bmatrix} + \begin{bmatrix} 10 & 16 & 16 \\ -5 & -3 & -5 \\ 12 & 6 & 6 \end{bmatrix}$$

$$+ \begin{bmatrix} -12 & 15 & 59 \\ 54 & -36 & -14 \\ 31 & -15 & 70 \end{bmatrix} + \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$x_1 = \begin{bmatrix} 15 & 44 & 88 \\ 29 & -17 & -33 \\ 61 & 4 & 80 \end{bmatrix}$$

Adapun untuk menghitung $x2$ dan $x3$ dihitung berdasarkan persamaan 4.2

$$x_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 8 & 5 & 0 \\ 0 & 0 & 9 & 0 & 0 \\ 0 & 6 & 7 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 6 & 5 & 0 \\ 0 & 2 & 2 & 4 & 0 \\ 0 & 1 & 5 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$* \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 8 & 0 \\ 0 & 8 & 0 & 1 & 0 \\ 0 & 5 & 1 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$+ \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$x_2 = \begin{bmatrix} 19 & 29 & 17 \\ -9 & 15 & -3 \\ 17 & 19 & 7 \end{bmatrix} + \begin{bmatrix} -2 & 34 & 28 \\ 1 & -9 & 13 \\ -1 & 30 & -3 \end{bmatrix} + \begin{bmatrix} -12 & 8 & 27 \\ 27 & -14 & -13 \\ 11 & -10 & 34 \end{bmatrix}$$

$$+ \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$x_2 = \begin{bmatrix} 6 & 72 & 73 \\ 20 & -37 & -2 \\ 28 & 40 & 39 \end{bmatrix}$$

$$x_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 8 & 5 & 0 \\ 0 & 0 & 9 & 0 & 0 \\ 0 & 6 & 7 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 6 & 5 & 0 \\ 0 & 2 & 2 & 4 & 0 \\ 0 & 1 & 5 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$* \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 8 & 0 \\ 0 & 8 & 0 & 1 & 0 \\ 0 & 5 & 1 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}$$

$$+ \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$x_3 = \begin{bmatrix} 31 & 44 & 23 \\ -36 & 39 & -30 \\ 32 & 40 & -8 \end{bmatrix} + \begin{bmatrix} -4 & 16 & 10 \\ 4 & -9 & 8 \\ -3 & 16 & -5 \end{bmatrix} + \begin{bmatrix} 0 & 15 & 23 \\ 6 & -9 & -20 \\ 4 & -21 & 19 \end{bmatrix}$$

$$+ \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$x_3 = \begin{bmatrix} 28 & 76 & 57 \\ 25 & 22 & -41 \\ 34 & 78 & 7 \end{bmatrix}$$

Sehingga didapatkan *mini-batch mean* :

$$\frac{1}{3}(\begin{bmatrix} 15 & 44 & 88 \\ 29 & -17 & -33 \\ 61 & 4 & 80 \end{bmatrix} + \begin{bmatrix} 6 & 72 & 73 \\ 20 & -37 & -2 \\ 28 & 40 & 39 \end{bmatrix} + \begin{bmatrix} 28 & 76 & 57 \\ 25 & 22 & -41 \\ 34 & 78 & 7 \end{bmatrix})$$

$$= \begin{bmatrix} 16.33 & 64 & 72.66 \\ 8 & -10.66 & -25.33 \\ 41 & 40.66 & 42 \end{bmatrix}$$

mini-batch mean inilah yang selanjutnya akan digunakan untuk menghitung *mini-batch varian*, *normalize*, serta *scale and shift*.

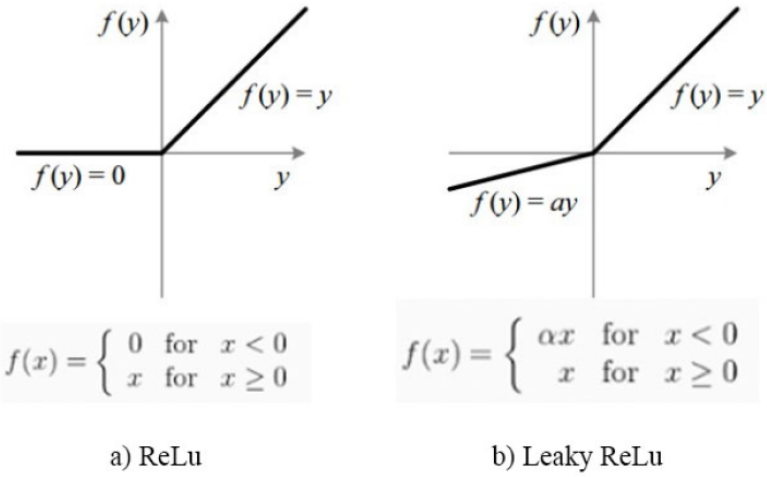
3. ***Leaky ReLU***

   Tahap berikutnya adalah menerapkan fungsi aktifasi *Leaky ReLU*. Pada dasarnya fungsi *Leaky ReLU* muncul untuk menyelesaikan masalah fungsi ReLU. Pada fungsi ReLU semua nilai negatif akan dipetakan menjadi nol, hal tersebut yang menurunkan kemampuan model untuk menyesuaikan atau melatih dari data dengan benar. Hal Itu berarti setiap masukan negatif yang diberikan kepada fungsi aktivasi ReLU mengubah nilai menjadi nol dalam grafik yang pada gilirannya mempengaruhi grafik yang dihasilkan dengan tidak memetakan nilai negatif secara tepat. Perbedaan Grafik fungsi Relu dan *Leaky Relu* dapat dilihat pada Gambar 4.18. *Leaky ReLU* membantu untuk menambah range dari fungsi ReLU. Pada jaringan Yolov3 nilai dari *alfa* yang digunakan adalah 0.1. Proses aktivasi ReLU dapat diilustrasikan seperti pada Gambar 4.19.

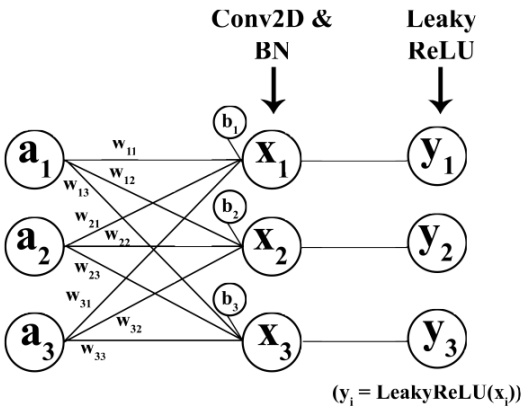
   Sebagai ilustrasi perhitungan digunakan matriks x_1, x_2 dan x_3 yang diperoleh dari tahap *Batch Normalization*.

$$x_1 = \begin{bmatrix} 15 & 44 & 88 \\ 29 & -17 & -33 \\ 61 & 4 & 80 \end{bmatrix}, y_1 = LeakyReLU(x_1)$$

$$y_1 = \begin{bmatrix} 15 & 44 & 88 \\ 29 & -1.7 & -3.3 \\ 61 & 4 & 80 \end{bmatrix}$$



Gambar 4.18. Grafik fungsi Relu dan *Leaky Relu*



Gambar 4.19. Ilustrasi proses *Leaky Relu*

$$x_2 = \begin{bmatrix} 6 & 72 & 73 \\ 20 & -37 & -2 \\ 28 & 40 & 39 \end{bmatrix}, y_2 = LeakyReLU(x_2)$$

$$y_2 = \begin{bmatrix} 6 & 72 & 73 \\ 20 & -3.7 & -0.2 \\ 28 & 40 & 39 \end{bmatrix}$$

$$x_3 = \begin{bmatrix} 28 & 76 & 57 \\ 25 & 22 & -41 \\ 34 & 78 & 7 \end{bmatrix}, y_3 = LeakyReLU(x_3)$$

$$y_3 = \begin{bmatrix} 28 & 76 & 57 \\ 25 & 22 & -4.1 \\ 34 & 78 & 7 \end{bmatrix}$$

4. ***Shortcut Layer***

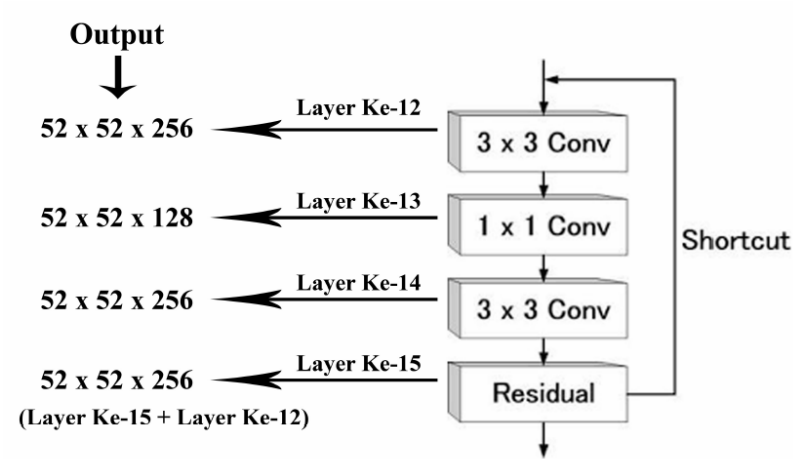
   Pada jaringan YOLOv3 *Shortcut Layer* merupakan sebuah *skip connection*, layer ini mirip dengan yang digunakan di pada jaringan ResNet. Parameter pada *shortcut layer* adalah -3, yang berarti *output* dari layer *shortcut* diperoleh dengan menambahkan hasil peta fitur dari layer sebelumnya dengan peta fitur mundur sebanyak 3 layer dari layer shortcut. Adapun ilustrasi proses *Shortcut Layer* ditampilkan pada Gambar 4.20 sementara perhitungan *Shortcut Layer* ditampilkan pada Gambar 4.21.

5. ***upsample***

   Pada jaringan YOLOv3 proses *upsample* dilakukan dengan menggunakan *input*hasil peta fitur dari *layer* sebelumnya, teknik yang digunakan pada proses ini adalah *bilinear upsampling*dengan nilai faktor 2. Pada proses *upsample* ukurannya menjadi $52 \times 52 \times 256$ (faktor = 2).

6. ***Route***

   Pada proses *route* jaringan YOLOv3 memiliki 2 lapisan atribut yang terdiri dari 1 parameter dan 2 parameter. Ketika atribut layer hanya memiliki 1 parameter, pada proses *route* ini akan



Gambar 4.20. Proses *Shortcut Layer*

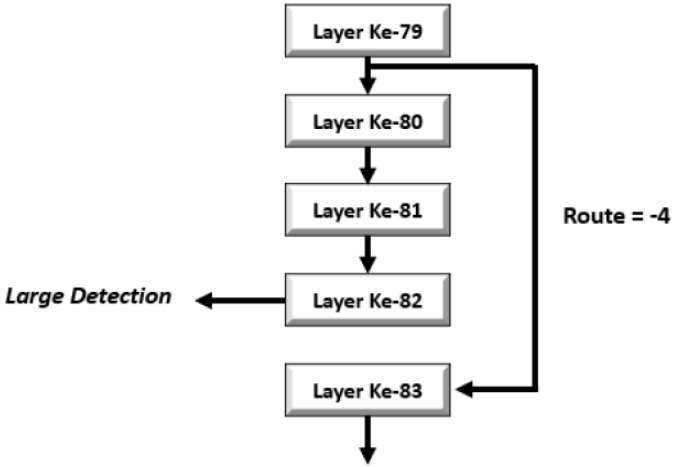
$$a^{[15]} = a^{[14]} + a^{[12]}$$

$$a^{[15]} = \begin{cases} \begin{pmatrix} 1 & 2 & 0 \\ 1 & 0 & 0 \\ 0 & 5 & 1 \end{pmatrix} + \begin{pmatrix} 3 & 1 & 2 \\ 2 & 5 & 5 \\ 1 & 4 & 0 \end{pmatrix} \\ \begin{pmatrix} 2 & 1 & 1 \\ 0 & 1 & 0 \\ 5 & 0 & 3 \end{pmatrix} + \begin{pmatrix} 4 & 1 & 0 \\ 4 & 3 & 1 \\ 3 & 1 & 5 \end{pmatrix} \end{cases}$$

$$a^{[15]} = \begin{cases} \begin{pmatrix} 4 & 3 & 2 \\ 3 & 5 & 5 \\ 1 & 9 & 1 \end{pmatrix} \\ \begin{pmatrix} 6 & 2 & 1 \\ 4 & 4 & 1 \\ 8 & 1 & 8 \end{pmatrix} \end{cases}$$

Gambar 4.21. Ilustrasi perhitungan *Shortcut Layer*

menampilkan peta fitur dari layer yang di indeks oleh nilai parameter tersebut. Dalam jaringan YOLOv3 parameter ini memiliki nilai -4 *(route* = -4*)*, sehingga layer akan menampilkan peta fitur dari lapisan ke-4 sebelum lapisan *route*. Ilustrasi proses dan perhitungan ditampilkan pada Gambar 4.22 dan 4.23.



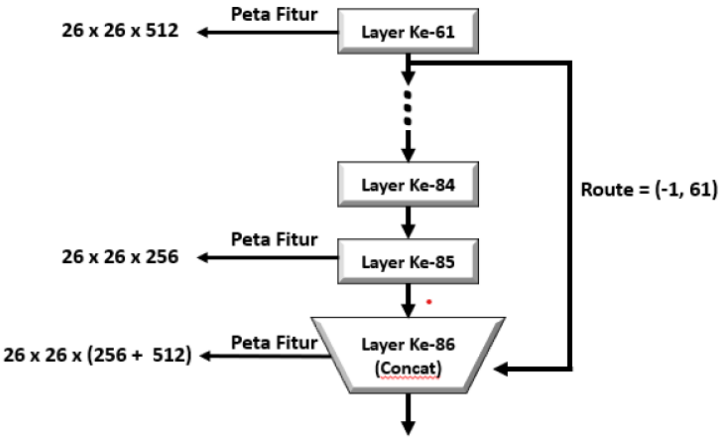
Gambar 4.22. Proses *route* dengan parameter -4

Ketika lapisan memiliki 2 parameter, proses ini akan mengembalikan peta fitur gabungan dari lapisan yang di indeks oleh parameter tersebut. Dalam jaringan YOLOv3 parameter ini memiliki nilai *route = (-1, 61)*, Jika parameter yang muncul adalah *route = (-1, 61)*, maka layer akan menampilkan peta fitur dari lapisan sebelumnya (-1) dan lapisan ke-61, digabungkan sepanjang dimensi kedalaman peta. Ilustrasi proses dan perhitungan ditampilkan pada Gambar 4.24 dan 4.25.

7. **Deteksi pada tiga skala**

$$
\begin{aligned}
a^{[79]} &= \begin{pmatrix} 2 & 4 & 3 \\ 3 & 2 & 4 \\ 1 & 3 & 4 \end{pmatrix} \\
a^{[83]} &= Route(-4) \\
a^{[83]} &= a^{[83-4]} \\
a^{[83]} &= a^{[79]} \\
a^{[83]} &= \begin{pmatrix} 2 & 4 & 3 \\ 3 & 2 & 4 \\ 1 & 3 & 4 \end{pmatrix}
\end{aligned}
$$

Gambar 4.23. Ilustrasi perhitungan *route* dengan parameter -4



Gambar 4.24. Proses *route* dengan parameter -1,61

Berdasarkan desain arsitektur YOLOv3 pada Gambar 4.9 diketahui bahwa jaringan Yolov3 melakukan pendeteksian dengan memprediksi pada 3 skala yang berbeda. Skala pertama terjadi pada lapisan ke-82 dengan $stride = 32$. Karena

$$a^{[61]} = \begin{cases} \begin{pmatrix} 5 & 2 & 8 \\ 1 & 8 & 2 \\ 1 & 2 & 9 \end{pmatrix} \\ \begin{pmatrix} 3 & 6 & 8 \\ 1 & 4 & 5 \\ 2 & 3 & 5 \end{pmatrix} \end{cases}, \quad a^{[85]} = \begin{pmatrix} 9 & 7 & 0 \\ 2 & 3 & 0 \\ 7 & 5 & 5 \end{pmatrix}$$

$$a^{[86]} = Route(-1, 61)$$

$$a^{[86]} = Concat(a^{[85]}, a^{[61]})$$

$$a^{[86]} = \begin{cases} \begin{pmatrix} 5 & 2 & 8 \\ 1 & 8 & 2 \\ 1 & 2 & 9 \end{pmatrix} \\ \begin{pmatrix} 3 & 6 & 8 \\ 1 & 4 & 5 \\ 2 & 3 & 5 \end{pmatrix} \\ \begin{pmatrix} 9 & 7 & 0 \\ 2 & 3 & 0 \\ 7 & 5 & 5 \end{pmatrix} \end{cases}$$

Gambar 4.25. Ilustrasi perhitungan *route* dengan parameter -1,61

ukuran dimensi gambar masukan adalah 416×416, maka ukuran skala pada lapisan ini adalah $(\frac{416}{32} \times \frac{416}{32}) = (13 \times 13)$. Skala kedua terjadi pada lapisan ke-94 dengan $stride = 16$ maka ukuran skala pada lapisan ini adalah $(\frac{416}{16} \times \frac{416}{16}) = (26 \times 26)$. Sementara itu Skala ketiga terjadi pada lapisan ke-106 dengan $stride = 8$. maka ukuran skala pada lapisan ini adalah $(\frac{416}{8} \times \frac{416}{8}) = (52 \times 52)$.

Hal pertama yang perlu diketahui dalam mengintepretasikan output dari YoloV3 ini adalah *input* adalah kumpulan gambar sebanyak m dengan ukuran $(416, 416, 3)$ dan *output* adalah daftar kotak pembatas dengan kelas objek yang dikenali. Setiap kotak pembatas diwakili oleh 6 parameter yaitu (pc, bx, by, bh, bw, c). Jika parameter c diperluas menjadi

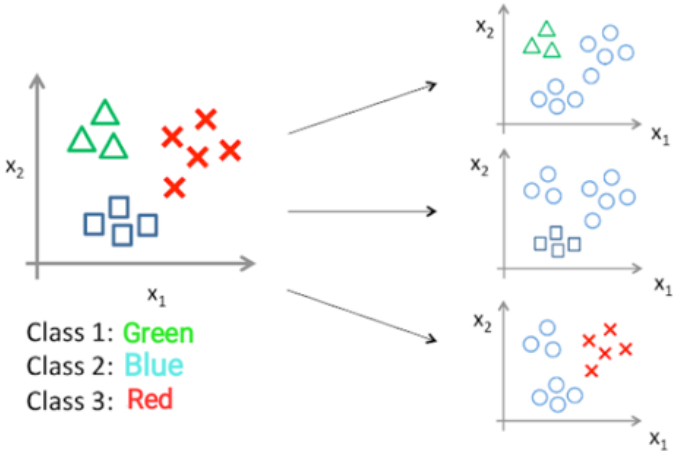
vektor 2-dimensi dengan c adalah kelas objek, maka setiap kotak pembatas diwakili oleh 41 parameter.

Dengan cara yang sama seperti pada semua fitur objek detektor yang telah dipelajari oleh *convolutional layer*, YOLO melakukan prediksi dengan menggunakan lapisan konvolusional (konvolusi = 1×1). Sehingga hal pertama yang harus diperhatikan adalah *output* dari YOLO adalah peta fitur. Karena YOLO telah menggunakan konvolusi 1×1, maka ukuran peta prediksi persis dengan ukuran peta fitur sebelumnya.

Pada YOLOv3 setiap skala akan melakukan prediksi kelas objek menggunakan *regression classification* dan *anchor boxes* yang diperoleh dari *clustering* anotasi data. Setiap skala $(13 \times 13, 26 \times 26, 52 \times 52)$ pada *pixel-pixel*nya memiliki 3 buah *anchor box* sehingga total *anchor box* yang digunakan pada ada sebanyak 9 dengan ukuran yang berbeda-beda sesuai hasil *clustering* dari anotasi data. Setiap *anchor box* memiliki $5 + C$ atribut, dimana nilai 5 merepresentasikan 5 atribut diantaranya Skor Objektifitas (P_0), Koordinat x titik tengah kotak pembatas (t_x), Koordinat y titik tengah kotak pembatas (t_y), lebar kotak pembatas (t_w), tinggi kotak pembatas (t_h) dan skor kelas objek (c). Gambar 4.26 dan 4.27 berisi ilustrasi dan persamaan *regression classification* sementara Gambar 4.28 mengilustrasikan berbagai ukuran *anchor box* yang digunakan.

Selama *training*, YOLOv3 memaksa jaringan untuk menggunakan anchor box yang memiliki dimensi dengan IOU yang lebih tinggi dengan ground truth. Sehingga jaringan akan menghasilkan t_x, t_y, t_w, dan t_h untuk memperbaiki *anchor box* dalam melakukan pengenalan objek. Adapun ilustrasinya akan ditampilkan pada Gambar 4.29

8. **Hasil Keluaran**



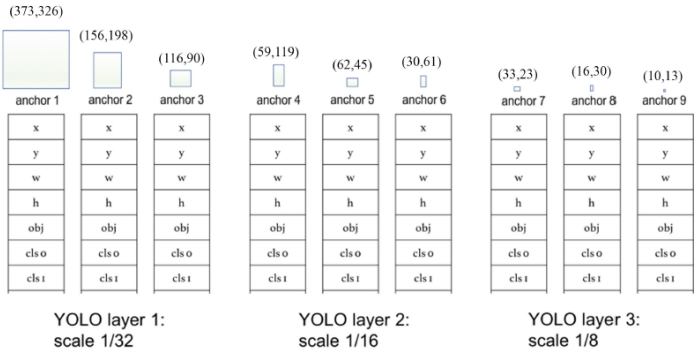
Gambar 4.26. Ilustrasi *regression classification*

$$
\begin{aligned}
& y \in \{0, 1 \ldots n\} \\
& h_\theta^{(0)}(x) = P(y = 0|x;\theta) \\
& h_\theta^{(1)}(x) = P(y = 1|x;\theta) \\
& \cdots \\
& h_\theta^{(n)}(x) = P(y = n|x;\theta) \\
& \text{prediction} = \max_i(h_\theta^{(i)}(x))
\end{aligned}
$$

Gambar 4.27. Persamaan *regression classification*

Pada gambar dengan ukuran 416×416, YOLOv3 memprediksi $((52 \times 52) + (26 \times 26) + 13 \times 13) \times 3 = 10647$ kotak pembatas. Jika hanya ada satu objek yang dikenali dalam satu gambar masukan, maka YOLOV3 akan mengurangi deteksi dari 10647 menjadi 1.

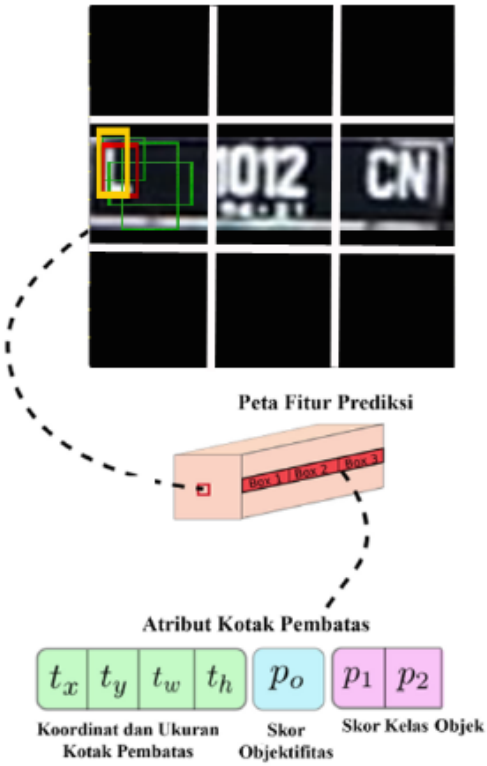
Pertama, YOLOv3 memfilter kotak berdasarkan skor ob-



Gambar 4.28. *Anchor boxes*

jeknya. Umumnya, kotak yang memiliki skor di bawah ambang batas (misalnya di bawah 0, 5) diabaikan. Selanjutnya, *Non-maxima Suppression* (NMS) digunakan untuk menyelesaikan masalah kotak pembatas yang saling tumpang tindih dalam mengenali objek yang sama. Sebagai contoh, semua 3 bounding box sel kotak merah dapat mendeteksi sebuah kotak atau sel yang berdekatan dan mendapatkan hasil deteksi objek yang sama, sehingga NMS digunakan untuk menghapus beberapa deteksi. Secara khusus, YOLOv3 akan melakukan menghapus kotak pembatas dengan skor rendah (artinya, kotak tidak terlalu confidence dalam mendeteksi kelas) kemudian dilanjutkan dengan memilih hanya satu kotak pembatas ketika beberapa kotak saling tumpang tindih dan mendeteksi objek yang sama.

Hal pertama yang dilakukan, YOLOv3 akan menerapkan filter pertama dengan thresholding. YOLOv3 ingin menyingkirkan kotak mana pun yang ”skor” kelasnya kurang dari ambang yang dipilih. Suatu model memberi total angka $13 \times 13 \times 3 \times 41$, dengan masing-masing kotak dijelaskan oleh

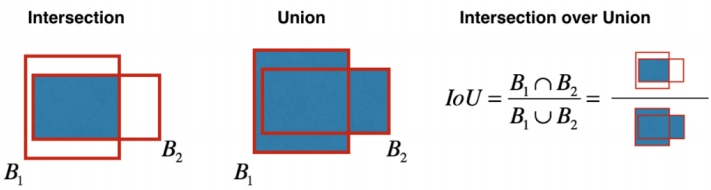


Gambar 4.29. Ilustrasi hasil deteksi

41 angka. Akan lebih mudah untuk mengatur ulang tensor dimensi $(13, 13, 3, 41)$ atau $(13, 13, 123)$ ke dalam variabel berikut antara lain : *box confidence* yaitu tensor dengan ukuran $(13 \times 13, 3, 1)$ yang mengandung pc (probabilitas kepercayaan bahwa ada beberapa objek), kotak pembatas adalah tensor dengan ukuran $(13 \times 13, 3, 4)$ berisi (b_x, b_y, b_h, b_w), dan *box class probs* yaitu tensor dari ukuran $(13 \times 13, 3, 36)$

yang mengandung probabilitas deteksi untuk masing-masing kelas.

Setelah pemfilteran dengan menetapkan nilai skor kelas, YOLOv3 masih memiliki banyak kotak pembatas yang tumpang tindih. Filter kedua untuk memilih kotak yang tepat disebut *Non-maksimum Suppression* atau NMS. NMS menggunakan fungsi yang sangat penting yang disebut *Intersection over Union*, atau IoU. Untuk melakukan perhitungan pertama-tama diperlukan informasi pusat koordinat kotak pembatas (x, y) dan tinggi serta lebar (w, h) dari kotak pembatas, kemudian akan dicari ukuran lebar dan tinggi kotak perpotongan dan didapatkan luas perpotongan. Adapun luas gabungan diperoleh dengan mengurangkan luas kotak gabungan kotak pembatas dengan luas perpotongan. Syarat dapat dicarinya nilai IoU adalah dua kotak pembatas yang dicari nilai IoU-nya tidak saling asing atau harus saling tumpang tindih, karena ketika kedua kotak pembatas tidak saling tumpang tindih maka nilai IoU adalah 0. Adapun ilustrasi mengenai IoU akan ditampilkan pada Gambar 4.30. B_1 dan B_2 melambangkan kotak pembatas yang akan dicari nilai IoUnya.



Gambar 4.30. Ilustrasi perhitungan IoU

Untuk menerapkan *Non-maksimum Suppression*, ada beberapa langkah yang perlu diperhatikan antara lain :

Memilih kotak yang memiliki skor tertinggi, kemudian hitung tumpang tindihnya dengan semua kotak pembatas yang lain,

dan hapus kotak yang tumpang tindih yang lebih dari *iou threshold*, dan ulangi sampai tidak ada lagi kotak dengan skor lebih rendah dari kotak yang dipilih saat ini.

9. ***Loss Function***

   Pada tahap ini akan dicari nilai loss function dari model jaringan YOLOv3 yang telah dibuat. Adapun *Loss Function* pada YOLOv3 diantaranya *Loss Function Localization*, *Loss Function Localization*, dan *Loss Function Classification* yang ditampilkan pada persamaan 4.3.

$$
\begin{aligned}
Loss &= Error_{localization} + Error_{class} + Error_{confidence} \\
Error_{localization} &= \sum_{i=1}^{\max box\_number} [(x_{truth} - \hat{x}_{predict})^2 + (y_{truth} - \hat{y}_{predict})^2 \\
&\quad + (w_{truth} - \hat{w}_{predict})^2 + (h_{truth} - \hat{h}_{predict})^2] \\
Error_{class} &= \sum_{i=1}^{\max box\_number} [-class_{truth} \times \log class_{predict} \\
&\quad - (1 - class_{truth}) \times \log(1 - class_{predict})] \\
Error_{confidence} &= \sum_{i=1}^{\max box\_number} [-confidence_{truth} \times \log confidence_{predict} \\
&\quad - (1 - confidence_{truth}) \times \log(1 - confidence_{predict})]
\end{aligned}
\tag{4.3}
$$

10. ***Back Propagation***

    Backpropagation adalah algoritma pembelajaran untuk memperkecil tingkat error dengan cara menyesuaikan bobotnya berdasarkan perbedaan *output* dan target yang diinginkan. Proses ini digunakan ketika tahap *training* yang bertujuan un-

tuk menyesuaikan kembali tiap *weights* dan *bias* berdasarkan *loss function* yang didapat pada saat *forward pass*

Proses *training* dengan menggunakan *backpropagation* melalui tiga tahapan yaitu proses *feedforward* , *backpropagating–errors* dan pengaturan *(update)* nilai bobot. Setelah proses *training*, dilakukan proses *testing* yang hanya melalui tahapan feedforward.

4.2.4 Pengenalan Plat Nomor

Output dari metode YOLOv3 yang diterapkan adalah *bounding box* pada tiap karakter yang dikenali dengan label kelas dan nilai *confidence*. *Bounding box* memberikan informasi koordinat dari karakter yang dikenali pada suatu gambar plat nomor.

Setelah didapatkan *bounding box* dengan label kelas dan nilai *confidence* dari karakter yang dikenali maka tahapan selanjutnya adalah pembacaan label kelas berdasarkan koordinat *bounding box* dari ujung kiri atas ke kanan bawah. Hal ini dilakukan karena hasil yang diinginkan pada penelitian ini adalah pengenalan plat nomor namun hasil yang didapatkan masih berupa label karakter yang urutannya masih acak berdasarkan karakter yang dikenali terlebih dahulu.

4.3 Implementasi Sistem

Rancangan data dan proses yang telah dilakukan diimplementasikan pada perangkat lunak untuk mengetahui hasil kinerja metode. Pada subbab ini akan disajikan *code* dan penjelasan dari pengerjaan anotasi data, *training*, *testing* dan uji keakuratan.

4.3.1 *Preprocessing* Data

Pada tahap ini dilakukan pengolahan data untuk memperbaiki kualitas dari data yang digunakan. Adapun *code* dari *preprocessing* data disajikan pada Gambar 4.31.

Setelah memilih filter yang dianggap paling baik untuk diterapkan pada data awal, data kemudian dibagi menjadi data yang digunakan untuk proses *training* dan data yang digunakan untuk proses *testing*. Pada data latih selanjutnya dilakukan proses anotasi data sementara pada data uji ditambahkan gangguan gelap dan terang. Adapun *source code* untuk menambahkan gangguan gelap dapat dilihat pada Gambar 4.32, sementara *source code* untuk menambahkan gangguan terang dapat dilihat pada Gambar 4.33.

```
import numpy as np
import cv2
from os import listdir
from os.path import isfile, join

#mypath='/path/to/folder/images'
mypath='C:/Users/Asus/Downloads/yolo-object-detection/tambahan'
onlyfiles = [ f for f in listdir(mypath) if isfile(join(mypath,f)) ]
image = np.empty(len(onlyfiles), dtype=object)
d=0
for n in range(0, len(onlyfiles)):
    image[n] = cv2.imread( join(mypath,onlyfiles[n]) )
    scale_percent = 400
    width = int(image[n].shape[1] * scale_percent / 100)
    height = int(image[n].shape[0] * scale_percent / 100)
    dsize = (width, height)
    output = cv2.resize(image[n], dsize)

    hsv = cv2.cvtColor(output, cv2.COLOR_BGR2HSV)
    h, s, v = cv2.split(hsv)
    v = cv2.add(v,-40)
    v[v > 255] = 255
    v[v < 0] = 0
    final_hsv = cv2.merge((h, s, v))
    img = cv2.cvtColor(final_hsv, cv2.COLOR_HSV2BGR)

    kernel = np.array([[-1,-1,-1], [-1,10,-1], [-1,-1,-1]])
    im = cv2.filter2D(img, -1, kernel)

    kernel2 = np.array([[0.11,0.11,0.11], [0.11,0.11,0.11], [0.11,0.11,0.11]])
    imm = cv2.filter2D(im, -1, kernel2)

    hsv = cv2.cvtColor(imm, cv2.COLOR_BGR2HSV)
    h, s, v = cv2.split(hsv)
    v = cv2.add(v,-50)
    v[v > 255] = 255
    v[v < 0] = 0
    final_hsv = cv2.merge((h, s, v))
    imo = cv2.cvtColor(final_hsv, cv2.COLOR_HSV2BGR)

    kernel3 = np.array([[-1,-1,-1], [-1,10,-1], [-1,-1,-1]])
    imp = cv2.filter2D(imo, -1, kernel3)

    hsv = cv2.cvtColor(imp, cv2.COLOR_BGR2HSV)
    h, s, v = cv2.split(hsv)
    v = cv2.add(v,-50)
    v[v > 255] = 255
    v[v < 0] = 0
    final_hsv = cv2.merge((h, s, v))
    imq = cv2.cvtColor(final_hsv, cv2.COLOR_HSV2BGR)

    filename = "tambahan/filed25 (%d).jpg"%d
    cv2.imwrite(filename, imq)
    d+=1
```

Gambar 4.31. *Code preprocessing data*

```python
import numpy as np
import cv2
from os import listdir
from os.path import isfile, join

#mypath='/path/to/folder/images'
mypath='C:/Users/Asus/Downloads/yolo-object-detection/images/tesfixjpg'
onlyfiles = [ f for f in listdir(mypath) if isfile(join(mypath,f)) ]
image = np.empty(len(onlyfiles), dtype=object)
d=0
for n in range(0, len(onlyfiles)):
    image[n] = cv2.imread( join(mypath,onlyfiles[n]) )
    hsv = cv2.cvtColor(image[n], cv2.COLOR_BGR2HSV)
    h, s, v = cv2.split(hsv)
    v = cv2.add(v,-75)
    v[v > 255] = 255
    v[v < 0] = 0
    final_hsv = cv2.merge((h, s, v))
    img = cv2.cvtColor(final_hsv, cv2.COLOR_HSV2BGR)
    filename = "images/dark75/file (%d).jpg"%d
    cv2.imwrite(filename, img)
    d+=1
```

Gambar 4.32. *Code* untuk menambahkan gangguan gelap

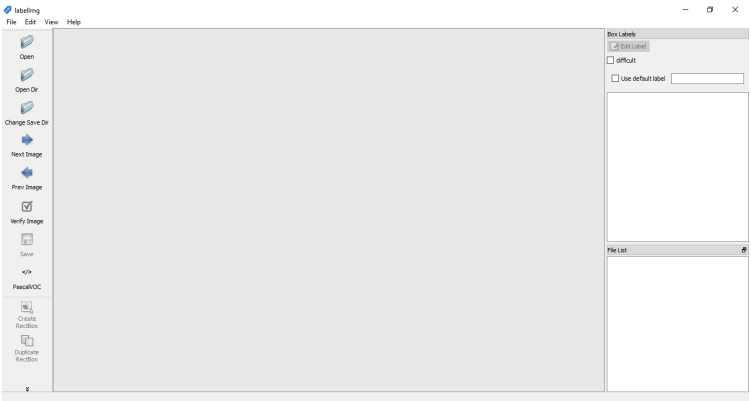
```python
import numpy as np
import cv2
from os import listdir
from os.path import isfile, join

#mypath='/path/to/folder/images'
mypath='C:/Users/Asus/Downloads/yolo-object-detection/images/tesfixjpg'
onlyfiles = [ f for f in listdir(mypath) if isfile(join(mypath,f)) ]
image = np.empty(len(onlyfiles), dtype=object)
d=0
for n in range(0, len(onlyfiles)):
    image[n] = cv2.imread( join(mypath,onlyfiles[n]) )
    hsv = cv2.cvtColor(image[n], cv2.COLOR_BGR2HSV)
    h, s, v = cv2.split(hsv)
    v = cv2.add(v,25)
    v[v > 255] = 255
    v[v < 0] = 0
    final_hsv = cv2.merge((h, s, v))
    img = cv2.cvtColor(final_hsv, cv2.COLOR_HSV2BGR)
    filename = "images/bright25/file (%d).jpg"%d
    cv2.imwrite(filename, img)
    d+=1
```

Gambar 4.33. *Code* untuk menambahkan gangguan terang

4.3.2 Anotasi Data

Untuk keperluan proses *training*, data yang digunakan harus melewati proses pelabelan atau anotasi data.

Proses anotasi data dilakukan dengan menggunakan perangkat lunak labelimg 1.7.0. Perangkat lunak ini dipilih karena dalam melakukan pelabelan data tersimpan sudah sesuai dengan format YOLO. Adapun tampilan perangkat lunak labelimg ditampilkan pada gambar 4.34. Terdapat dua tahap dalam proses anotasi data. Tahap pertama adalah merancang penyimpanan informasi kotak pembatas dan kelas. Informasi tersebut disimpan pada file dengan format .txt. Informasi penting yang diambil dan disimpan adalah kelas objek, koordinat, tinggi, dan lebar objek.



Gambar 4.34. Tampilan muka perangkat lunak labelimg

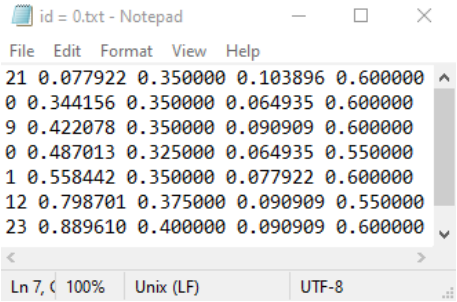
Tahap berikutnya adalah menggambar kotak pembatas setap objek pada citra. Tahap ini digunakan untuk mendefinisikan letak dan kelas objek sebagai informasi objek. Nantinya informasi tersebut akan dikirimkan ke tahap penyimpanan informasi yang telah disiapkan pada tahap pertama.

Proses kerja tahap anotasi data adalah memunculkan citra

dalam folder, menggambar kotak pembatas pada setiap objek, kemudian menyimpan informasi berupa titik kotak pembatas dan kelas objek. Adapun citra setelah dilakukan anotasi direpresentasikan pada Gambar 4.35.

Gambar 4.35. Kotak pembatas pada setiap objek

Hasil penyimpanan file .txt dapat dilihat pada Gambar 4.36 Isi file data anotasi tersebut memberikan informasi yang dibutuhkan tentang setiap objek pada citra. Informasi tersebut adalah kelas objek yang disimpan pada kolom pertama, (b_x, b_y) atau koordinat objek disimpan pada kolom kedua dan ketiga,bh tinggi objek pada kolom keempat, dan b_w lebar objek pada kolom kelima. Dalam contoh data anotasi tersebut terdapat 36 kelas objek yaitu karakter huruf dan angka (A-Z dan 0-9). 36 kelas dipilih karena sistem digunakan untuk mengenali plat nomor.



Gambar 4.36. Isi file data anotasi

4.3.3 *training* Data

training data dilakukan dengan menggunakan *Google Collaboration*, adapun tahapan - tahapannya akan dijelaskan sebagai berikut.

Tahap pertama pada proses *training* data adalah *cloning* dan membangun darknet. *Clone* Darknet untuk *custom dataset* pada github AlexeyAB. Kemudian sesuaikan *Makefile* untuk mengaktifkan OPENCV dan GPU untuk darknet dan kemudian membangun darknet. Adapun *code* pada tahap ini dijelaskan pada Gambar 4.37.

```
[ ]  # clone darknet repo
     !git clone https://github.com/AlexeyAB/darknet
```

```
[ ]  # change makefile to have GPU and OPENCV enabled
     %cd darknet
     !sed -i 's/OPENCV=0/OPENCV=1/' Makefile
     !sed -i 's/GPU=0/GPU=1/' Makefile
     !sed -i 's/CUDNN=0/CUDNN=1/' Makefile
```

```
[ ]  # verify CUDA
     !/usr/local/cuda/bin/nvcc --version
```

```
[ ]  # make darknet (build)
     !make
```

Gambar 4.37. *Cloning* dan membangun Darknet

Langkah selanjutnya adalah mengunggah file dari *Google Drive* ke *Cloud VM*. Diperlukan pemasangan *Google Drive (mount)* ke *Cloud VM* terlebih dahulu agar konten pada Google Drive dapat diakses. Adapun *code* pada tahap ini dijelaskan pada Gambar 4.38.

Tahap selanjutnya adalah menyiapkan data untuk proses *training*. Folder yang berisi data setelah dilakukan anotasi data diubah dalam format .zip kemudian diunggah pada *Google Drive*. Folder yang berisi semua gambar dan anotasi data akan terlihat seperti

```
[ ] %cd ..
    from google.colab import drive
    drive.mount('/content/gdrive')
```

```
[ ]
    !ln -s /content/gdrive/My\ Drive/ /mydrive
    !ls /mydrive
```

Gambar 4.38. *Mount Google Drive* ke *Cloud VM*

ini gambar dibawah ini yang mana setiap gambar harus memiliki file teks dengan nama yang sama. Adapun isi folder *training* ditampilkan pada Gambar 4.39.

Name	Date modified	Type	Size
0 (1).jpg	1/30/2020 5:09 PM	JPG File	2 KB
0 (1).txt	4/14/2020 10:45 PM	Text Document	1 KB
0 (2).jpg	1/30/2020 5:16 PM	JPG File	2 KB
0 (2).txt	4/14/2020 11:03 PM	Text Document	1 KB
0 (3).jpg	1/30/2020 5:17 PM	JPG File	2 KB
0 (3).txt	4/14/2020 11:06 PM	Text Document	1 KB
0 (4).jpg	1/30/2020 5:17 PM	JPG File	2 KB
0 (4).txt	4/14/2020 11:07 PM	Text Document	1 KB
0 (5).jpg	1/30/2020 5:17 PM	JPG File	2 KB
0 (5).txt	4/14/2020 11:08 PM	Text Document	1 KB

Gambar 4.39. Isi folder data *training*

Setelah data latih diunggah pada *Google Drive* maka perlu dipindahkan ke *Cloud VM*. Hal ini dilakukan agar data dapat digunakan pada saat proses *training*. Adapun *code* pada tahap ini dijelaskan pada Gambar 4.40.

Selain data latih diperlukan juga *file-file* lain yang digunakan pada saat proses *training*. *file-file* tersebut adalah *file* dengan format .names, .data, dan .cfg. *file* dengan format .names adalah *file* yang berisi data nama kelas dalam hal ini yaitu karakter (A-Z) dan

```
[ ] !ls /mydrive/yolov3
```

```
[ ] !cp /mydrive/yolov3/obj.zip ../
```

```
[ ] !unzip ../obj.zip -d data/
```

Gambar 4.40. Mengunggah folder data *training* ke *Cloud VM*

(0-9), file dengan format .data adalah file yang berisi jumlah kelas yang akan dilatih dan membuat path untuk menyimpan backup bobot hasil *training*, sementara untuk file dengan format .cfg untuk mengatur konfigurasi *training*. File-file tersebut harus diunggah pada *Google Drive* dan dipindahkan ke *Cloud VM* agar dapat digunakan selama proses *training*. Adapun file dengan format .data akan disajikan pada Gambar 4.41.

```
1  classes =36
2  train = data/train.txt
3  valid = data/test.txt
4  names = data/obj.names
5  backup = /mydrive/yolov3/backup/
```

Gambar 4.41. Isi file obj.data

File konfigurasi terakhir yang diperlukan sebelum melakukan *training* data adalah generate_train.py yang berguna untuk menahan jalur relatif ke semua gambar *training* yang digunakan. adapun isi file generate_train.py ditampilkan pada Gambar 4.42.

Tahap selanjutnya adalah mengunduh *pre-trained weight* untuk lapisan konvolusional dari jaringan YOLOv3 yang berasal dari Pjreddie. Proses ini diperlukan agar proses deteksi menjadi lebih akurat dan proses *training* akan waktu yang diperlukan untuk proses

```python
import os

image_files = []
os.chdir(os.path.join("data", "obj"))
for filename in os.listdir(os.getcwd()):
    if filename.endswith(".jpg"):
        image_files.append("data/obj/" + filename)
os.chdir("..")
with open("train.txt", "w") as outfile:
    for image in image_files:
        outfile.write(image)
        outfile.write("\n")
    outfile.close()
os.chdir("..")
```

Gambar 4.42. Isi file generate_train.py

training akan berkurang.

Tahap terakhir adalah memulai *training* data, waktu yang dibutuhkan untuk *training* bergantung pada banyaknya iterasi dan kecepatan internet yang digunakan. Hasil dari *training* data adalah file dengan format .weight yang menyimpan bobot hasil *training*.

4.3.4 *testing* Data

Proses selanjutnya yang harus dilakukan adalah proses *testing* data untuk menentukan apakah metode yang dipilih dapat mengenali plat nomor dengan baik. Untuk melakukan *testing* perlu disiapkan data uji yang digunakan. Adapun *code* program dalam *python* dan penjelasannya ditampilkan sebagai berikut:

Pertama-tama adalah melakukan *import package* yang dibutuhkan. Beberapa *package* yang dibutuhkan adalah *numpy, argparse, time* dan *os*, serta *tkinter* untuk membangun GUI. Adapun *code import package* ditampilkan pada Gambar 4.43.

Selanjutnya dilakukan *parse* terhadap beberapa *argument*. *Parse argument* diproses pada saat program dijalankan dan membuat kita dapat melakukan perubahan masukan pada terminal. *Code parse argument* ditampilkan pada Gambar 4.44. Penjelasan pada

```
import numpy as np
import argparse
import time
import cv2
import os
```

Gambar 4.43. *Code import package*

beberapa *argument* tersebut adalah sebagai berikut :

1. - -image : adalah jalur untuk membuka gambar masukan untuk proses *testing*. Proses *testing* akan dilakukan dengan metode YOLO-Darknet, dengan versi YOLO yang dipilih adalah YOLOv3.

2. - -yolo : adalah jalur untuk membuka folder yang berisi *file-file* yang dibutuhkan untuk proses pengenalan plat nomor. Beberapa diantaranya adalah coco.names yang berisi nama kelas, yolov3.weight yang berisi bobot hasil *training* yang telah dilakukan, serta yolov3.cfg yang berisi konfigurasi untuk melakukan *testing*.

3. - -confidence : adalah peluang minimum yang digunakan untuk melakukan filter terhadap pengenalan plat nomor, dalam hal ini dipilih 50% atau 0.5 sebagai *confidence*.

4. - -threshold : dipilih 0.3 sebagai *threshold* pada *non-maxima suppression*.

Tahap selanjutnya adalah memuat file coco.names yang berisi nama kelas dan memilih warna acak untuk membedakan tiap kelas. dilanjutkan dengan memuat file yolov3.weight yang berisi bobot hasil *training* yang telah dilakukan, serta yolov3.cfg yang berisi konfigurasi untuk melakukan *testing*. Untuk membaca file yolov3.weight dan yolov3.cfg diperlukan modul *dnn* dari OpenCV. *Code* untuk memuat *file* ditampilkan pada Gambar 4.45.

```
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
    help="path to input image")
ap.add_argument("-y", "--yolo", required=True,
    help="base path to YOLO directory")
ap.add_argument("-c", "--confidence", type=float, default=0.5,
    help="minimum probability to filter weak detections")
ap.add_argument("-t", "--threshold", type=float, default=0.3,
    help="threshold when applying non-maxima suppression")
args = vars(ap.parse_args())
```

Gambar 4.44. *Code parse argument*

```
labelsPath = os.path.sep.join(["coco.names"])
LABELS = open(labelsPath).read().strip().split("\n")

np.random.seed(42)
COLORS = np.random.randint(0, 255, size=(len(LABELS), 3),
    dtype="uint8")

weightsPath = os.path.sep.join(["yolov3.weights"])
configPath = os.path.sep.join(["yolov3.cfg"])


print("[INFO] loading YOLO from disk...")
net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)
```

Gambar 4.45. *Code* memuat *file*

Pada tahap selanjutnya gambar plat nomor yang akan dilakukan uji coba dimasukkan dan didapatkan dimensi gambar, dilanjutkan dengan menghitung nama *output layer* yang dibutuhkan dari YOLO dan membangun *blob* dari gambar masukan. *blob* dapat diartikan sebagai *preprocessing* data atau persiapan data sebelum diolah lebih lanjut. persiapan data yang dilakukan pada tahap ini adalah mengubah ukuran gambar menjadi 416×416. Karena OpenCV mengasumsikan sebuah gambar memiliki format BGR maka diperlukan penukaran R dan B sehinga membentuk format RGB. Kemudian akan dilakukan perhitungan waktu yang diper-

lukan YOLO-Darknet untuk melakukan pengenalan plat nomor. *Code* untuk tahap ini akan ditampilkan pada Gambar 4.46.

```
image = cv2.imread(args["image"])
(H, W) = image.shape[:2]

ln = net.getLayerNames()
ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()]

blob = cv2.dnn.blobFromImage(image, 1 / 255.0, (416, 416),
                             swapRB=True, crop=False)
net.setInput(blob)


start = time.time()
layerOutputs = net.forward(ln)
end = time.time()
print("[INFO] YOLO took {:.6f} seconds".format(end - start))
```

Gambar 4.46. *Code* pengolahan gambar dan perhitungan waktu

Pada tahap ini akan dilakukan inisialisasi beberapa list yang dibutuhkan untuk melakukan visualisasi hasil. Beberapa list tersebut meliputi *boxes* sebagai *bounding box* pada objek yang dikenali, *confidence* sebagai nilai confidence YOLO-Darknet menetapkan objek yang dikenali sebagai objek. Pada tahap sebelumnya nilai minimum dari *confidence* telah dilakukan inisialisasi yaitu 0.5 yang berarti jika nilai confidence dari suatu objek kurang dari 0.5 maka objek tersebut tidak dianggap objek oleh YOLO-Darknet. selanjutnya terdapat list *classIDs* yang berisi label kelas dari objek yang terdeteksi. *Code* untuk tahap ini akan ditampilkan pada Gambar 4.47.

```
boxes = []
confidences = []
classIDs = []
```

Gambar 4.47. *Code* inisialisasi *list*

Selanjutnya dilakukan *looping* pada *layer output* kemudian ekstraksi *classID* dan *confidence*. Hasil dari *confidence* yang diperoleh digunakan untuk melakukan filter terhadap objek yang terdeteksi. *Code looping layer output* akan ditampilkan pada Gambar 4.48.

```
for output in layerOutputs:
    for detection in output:
        scores = detection[5:]
        classID = np.argmax(scores)
        confidence = scores[classID]
```

Gambar 4.48. *Code looping layer output*

Pada tahap ini dilakukan perubahan skala pada koordinat *bounding box* sehingga *bounding box* dapat ditampilkan dengan benar pada gambar awal sebelum dilakukan *preprocessing*. *Code* untuk tahap ini akan ditampilkan pada Gambar 4.49.

```
        if confidence > args["confidence"]:
            box = detection[0:4] * np.array([W, H, W, H])
            (centerX, centerY, width, height) = box.astype("int")

            x = int(centerX - (width / 2))
            y = int(centerY - (height / 2))
```

Gambar 4.49. *Code* untuk mengubah skala *bounding box*

Dilanjutkan dengan ekstraksi koordinat dan dimensi *bounding box*. Pada YOLO-Darknet koordinat dari *bounding box* ditampilkan dalam bentuk (pusat pada sumbu X, pusat pada sumbu Y, lebar, dan tinggi). Kemudian dilakukan *update* pada *boxes, confidence* dan *classIDs*. *Code* ekstraksi *bounding box* akan ditampilkan pada Gambar 4.50.

Sorting dilakukan untuk mendapatkan text karakter plat nomor yang terdeteksi dari kiri ke kanan. *Code sorting* teks akan dita-

```
            boxes.append([x, y, int(width), int(height)])  # ini yan
            confidences.append(float(confidence))
            classIDs.append(classID)
```

Gambar 4.50. *Code* ekstraksi *bounding box*

mpilkan pada Gambar 4.51.

```
detected = []
for box, conf, ids in zip(boxes, confidences, classIDs):
    detected.append({
        'boxes': box,
        'confidences': conf,
        'class_id': ids,
        'karakter':LABELS[ids]
    })

sorted_dict = sorted(detected, key=lambda k: k['boxes'][:2])
```

Gambar 4.51. *Code sorting* teks

Tahap selanjutnya adalah mengaplikasikan *non-maxima suppression* yaitu menghilangkan *overlapping bounding box* dengan cara memilih satu *bounding box* dengan nilai *confidence* yang paling tinggi sehingga tidak terjadi *redundant*. *Code* untuk tahap ini akan ditampilkan pada Gambar 4.52.

```
idxs = cv2.dnn.NMSBoxes([a['boxes'] for a in sorted_dict], [a['confidences'] for a in sorted_dict],
                        args["confidence"], args["threshold"])
```

Gambar 4.52. *Code non-maxima suppression*

Tahap terakhir adalah menampilkan *bounding box* dan nama kelas pada gambar plat nomor. Selain itu akan ditampilkan juga *confidence* tiap karakter yang terdeteksi dan plat nomor dalam bentuk teks. *Code* untuk tahap ini akan ditampilkan pada Gambar 4.53.

```
for item in sorted_dict:
    print(f'{item["karakter"]} dengan confidence: {item["confidences"]}')
text = ''.join([a['karakter'] for a in sorted_dict])
print(text)

if len(idxs) > 0:
    for i in idxs.flatten():
        (x, y) = (sorted_dict[i]['boxes'][0],  sorted_dict[i]['boxes'][1])
        (w, h) = (sorted_dict[i]['boxes'][2], sorted_dict[i]['boxes'][3])

        color = [int(c) for c in COLORS[sorted_dict[i]['class_id']]]
        cv2.rectangle(image, (x, y), (x + w, y + h), color, 2)

        cv2.putText(image, text, (x, y - 5), cv2.FONT_HERSHEY_SIMPLEX,
                    0.5, color, 2)
 # show the output image
 cv2.imshow("Image", image)
 cv2.waitKey(0)
```

Gambar 4.53. *Code* untuk menampilkan hasil pengenalan plat nomor

BAB V
UJI COBA DAN PEMBAHASAN

Pada bab ini akan diberikan pembahasan tentang *testing* metode yang digunakan, yaitu YOLO-Darknet dengan varian YOLO yang dipilih adalah YOLOv3 dan arsitektur di dalamnya adalah Darknet-53. Beberapa pembahasan pada bab ini antara lain dataset *testing*, pembahasan hasil output *testing*, dan pengukuran kinerja metode.

5.1 Hasil *Preprocessing* Data

Kualitas data yang didapatkan dari hasil ekstraksi dan *cropping* video memiliki kualitas yang tidak terlalu baik seperti ukuran citra yang terlalu kecil dan terdapat sedikit efek *blur* pada citra. Karena itu telah diterapkan beberapa filter untuk memperbaiki kualitas citra. Adapun hasil dari penerapan filter yang digunakan dapat dilihat pada Gambar 5.1 dan 5.2.

Gambar 5.1. Hasil penerapan filter pada citra

Dari gambar 5.1 dan 5.2 terlihat bahwa hasil *preprocessing* data terlihat cukup baik. Citra yang awalnya berukuran kecil dan terdapat efek *blur* menjadi lebih jelas sehingga lebih mudah dilakukan pengenalan karakter.

Gambar 5.2. Hasil penerapan filter pada citra

5.2 Dataset *testing*

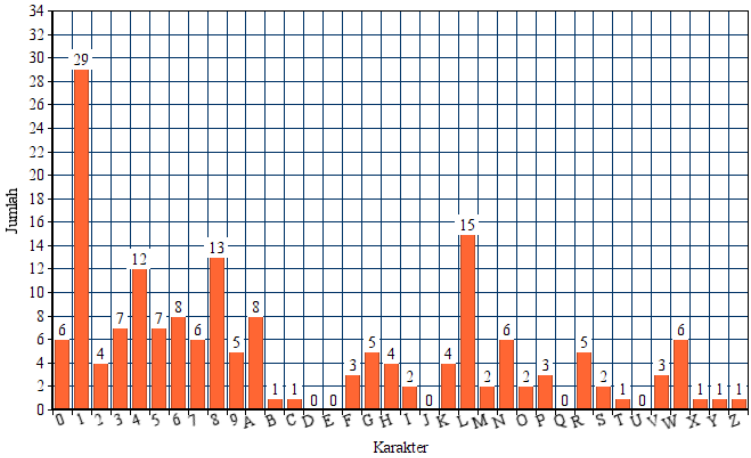
Sebelum melewati proses *testing*, dataset *testing* perlu disiapkan terlebih dahulu. Dataset *testing* digunakan untuk menguji sejauh mana model dapat mengenali objek karakter pada plat nomor. Karena terdapat dua model yang telah dilatih pada proses *training*, maka data uji juga dibedakan menjadi dua kondisi yaitu data uji tanpa *preprocessing* data dan data uji dengan *preprocessing* data. Selanjutnya dari masing-masing data uji, data dibedakan lagi menjadi dataset tanpa gangguan dan dataset dengan gangguan. Tiap dataset terdiri dari 25 plat nomor dengan 171 karakter. Adapun rincian karakter dari dataset yang digunakan dapat dilihat pada Tabel 5.1. Sedangkan sebaran data uji dapat dilihat pada grafik pada Gambar 5.3.

Untuk dataset dengan gangguan yaitu data tanpa gangguan yang diberikan tambahan intensitas warna dan pengurangan intensitas warna. Adapun penambahan dan pengurangan intensitas warna yang digunakan yaitu $25, 50,$ dan 75. Gambar 5.4 menampilkan citra uji coba dengan gangguan yang diberi penambahan intensitas warna untuk data tanpa *preprocessing*, gambar 5.5 menampilkan citra uji coba dengan gangguan yang diberi pengurangan intensitas warna untuk data tanpa *preprocessing*. Adapun Gambar 5.6 dan

Tabel 5.1. Data proses *testing*

NO	GAMBAR PLAT NOMOR	TEKS	JUMLAH KARAK-TER
1		L1434A	6
2		L1168WH	7
3		N511VR	6
4		S1888YO	7
5		N178NTN	7
6		W1914SN	7
7		L8449AH	7
8		L1426OC	7
9		L1780RM	7
10		L1203AB	7
11		P1096X	6
12		L1483HK	7

NO	GAMBAR PLAT NOMOR	TEKS	JUMLAH KARAK-TER
13		N1754AF	7
14		L8115AK	7
15		AG1949F	7
16		L1163PR	7
17		L8146VR	7
18		W1986PK	7
19		L1602MN	7
20		L1713FI	7
21		L1200ZI	7
22		W1540VK	7
23		AG368GH	7
24		L1715W	6
25		L1637PW	7



Gambar 5.3. Sebaran data uji

5.7 menampilkan citra dengan *preprocessing* yang diberi gangguan penambahan dan pengurangan intensitas warna.

5.3 *Batch* Maksimum

Pada saat proses *training* data dilakukan penentuan nilai dari *batch* maksimum yang digunakan. *Batch* maksimum adalah jumlah maksimal iterasi yang dilakukan untuk mendapatkan bobot yang diperlukan pada proses *testing*. Pada penelitian ini nilai *batch* maksimum yang digunakan adalah 1000, 5000, 10000, 15000, 18000, 20000, dan 25000. Pada saat proses *training* data, nilai *batch* maksimum sangat mempengaruhi waktu komputasi yang diperlukan. Saat *batch* maksimum yang digunakan adalah 1000 waktu yang diperlukan pada saat *training* data adalah 2.5 jam, ketika *batch* maksimum yang digunakan adalah 5000 waktu yang diperlukan pada

Gambar asli

Intensitas warna n=25

Intensitas warna n=50

Intensitas warna n=75

Gambar 5.4. Citra tanpa *preprocessing* dengan gangguan penambahan intensitas warna

Gambar asli

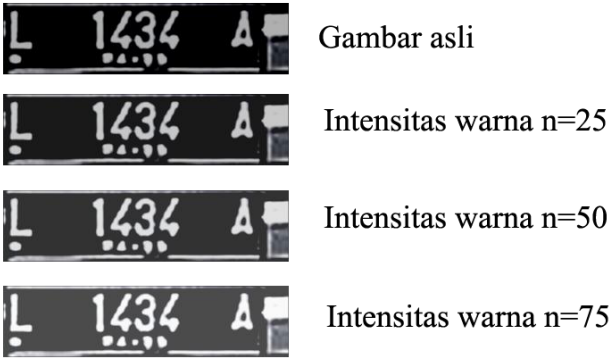
Intensitas warna n=-25

Intensitas warna n=-50

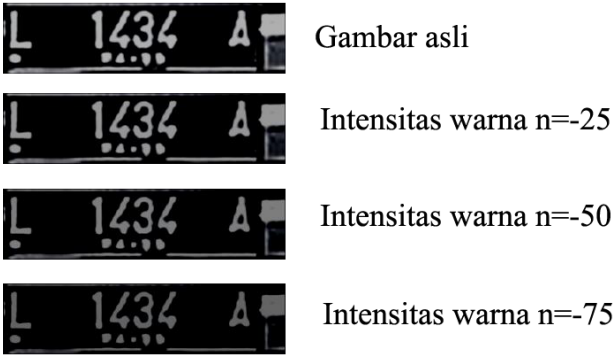
Intensitas warna n=-75

Gambar 5.5. Citra tanpa *preprocessing* dengan gangguan pengurangan intensitas warna

saat *training* data adalah 10.8 jam, ketika *batch* maksimum yang digunakan adalah 10000 waktu yang diperlukan pada saat *training* data adalah 37.6 jam, saat *batch* maksimum yang digunakan adalah



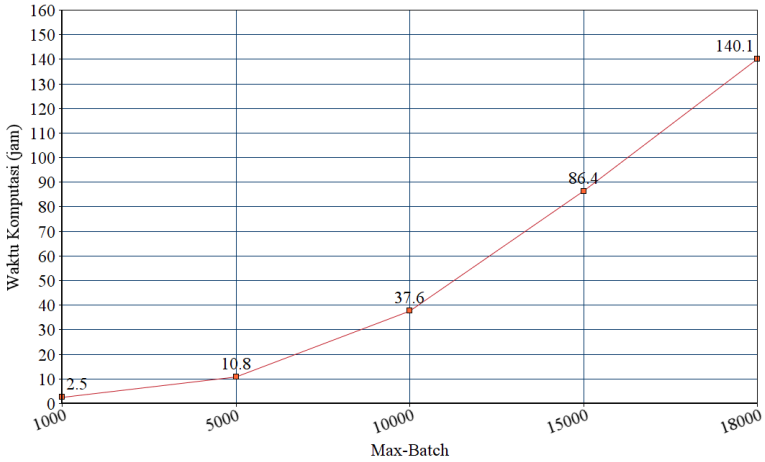
Gambar 5.6. Citra dengan *preprocessing* dengan gangguan penambahan intensitas warna



Gambar 5.7. Citra dengan *preprocessing* dengan gangguan pengurangan intensitas warna

15000 waktu yang diperlukan pada saat *training* data adalah 86.4 jam, dan saat *batch* maksimum yang digunakan adalah 18000 waktu yang diperlukan pada saat *training* data adalah 140.1 jam. Adapun

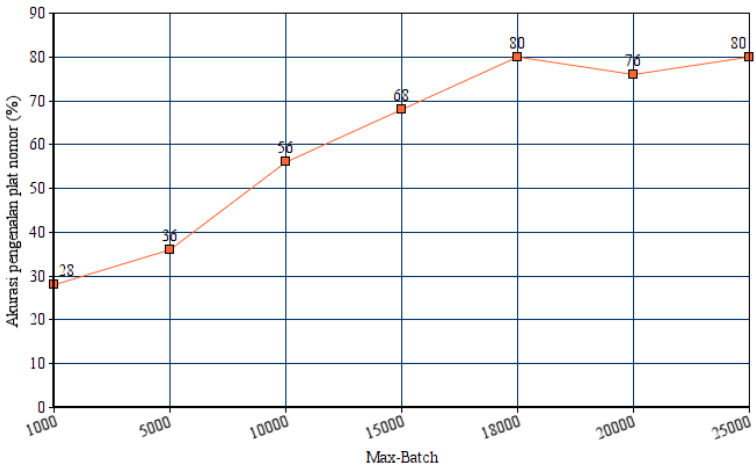
tampilan dalam grafik dapat dilihat pada Gambar 5.8.



Gambar 5.8. Grafik pengaruh *batch* maksimum terhadap waktu komputasi

Selain perbedaan waktu komputasi, akurasi yang didapatkan dari perbedaan *batch* maksimum yang digunakan juga berbeda ketika bobot hasil *training* digunakan untuk melakukan *testing* terhadap data uji citra tanpa gangguan yang berjumlah 25 gambar. Nilai *confidence* yang dipilih adalah 0.3 dan nilai *threshold* yang digunakan adalah 0.3. Akurasi yang diperoleh ketika *batch* maksimum yang digunakan 1000 adalah 28%. Ketika *batch* maksimum yang digunakan adalah 5000, akurasi yang didapat adalah 36%. Akurasi yang diperoleh ketika *batch* maksimum yang digunakan 10000 adalah 56%. Ketika *batch* maksimum yang digunakan adalah 15000, akurasi yang didapat adalah 68%. Akurasi yang diperoleh ketika *batch* maksimum yang digunakan 18000 adalah 80%. Ketika

batch maksimum yang digunakan adalah 20000, akurasi yang didapat adalah 76%. Dan akurasi yang diperoleh ketika *batch* maksimum yang digunakan 25000 adalah 80%. Terlihat bahwa hasil terbaik diperoleh ketika nilai *batch* maksimum adalah 18000. Grafik pada Gambar 5.9 menampilkan pengaruh *batch* maksimum terhadap hasil pengenalan plat nomor.



Gambar 5.9. Grafik pengaruh *batch* maksimum terhadap prosentase plat nomor yang dikenali

5.4 *Confidence*

Pada proses *testing*, diperlukan *confidence* yaitu nilai probabilitas minimum untuk memilah hasil objek yang dikenali. Hasil pengenalan yang memiliki nilai probabilitas lebih kecil dari nilai *confidence* akan dibuang sementara hasil pengenalan yang memiliki

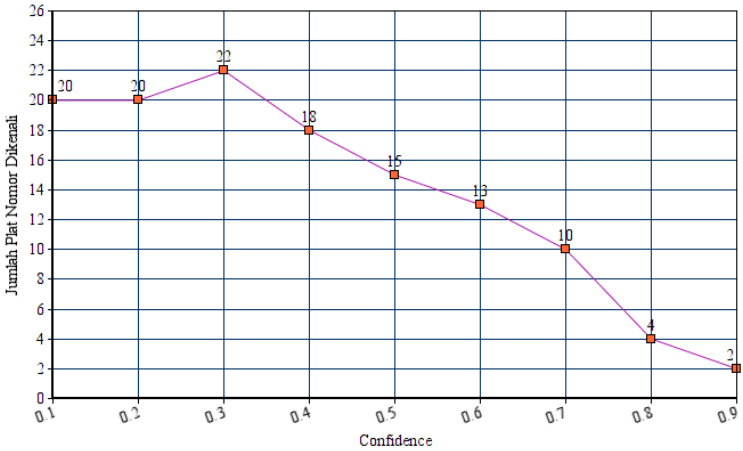
nilai probabilitas lebih besar atau sama dengan nilai *confidence* akan ditampilkan.

testing untuk menentukan nilai *confidence* dengan hasil terbaik dilakukan dengan data uji tanpa gangguan, adapun nilai *confidence* yang digunakan untuk melakukan *testing* adalah $0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8$, dan 0.9. Ketika nilai *confidence* yang digunakan adalah 0.1 jumlah plat nomor yang dikenali berjumlah 20 buah, Ketika nilai *confidence* yang digunakan adalah 0.2 jumlah plat nomor yang dikenali berjumlah 20 buah, Ketika nilai *confidence* yang digunakan adalah 0.3 jumlah plat nomor yang dikenali berjumlah 22 buah, Ketika nilai *confidence* yang digunakan adalah 0.4 jumlah plat nomor yang dikenali berjumlah 18 buah, Ketika nilai *confidence* yang digunakan adalah 0.5 jumlah plat nomor yang dikenali berjumlah 15 buah, Ketika nilai *confidence* yang digunakan adalah 0.6 jumlah plat nomor yang dikenali berjumlah 13 buah, Ketika nilai *confidence* yang digunakan adalah 0.7 jumlah plat nomor yang dikenali berjumlah 10 buah, Ketika nilai *confidence* yang digunakan adalah 0.8 jumlah plat nomor yang dikenali berjumlah 4 buah, dan Ketika nilai *confidence* yang digunakan adalah 0.9 jumlah plat nomor yang dikenali berjumlah 2 buah. Adapun hasil uji coba disajikan pada Gambar 5.10.

Dari Gambar 5.7 terlihat bahwa hasil *testing* dengan hasil terbaik didapatkan ketika nilai *confidence* yang digunakan adalah 0.3. Nilai *confidence* yang terlalu kecil menyebabkan hal-hal yang bukan objek dikenali sebagai objek. Gambar 5.11 menunjukkan kesalahan pengenalan ketika nilai *confidence* yang digunakan 0.2.

5.5 Waktu Komputasi

Untuk mengetahui waktu komputasi yang diperlukan untuk mengenali satu gambar plat nomor, dilakukan uji coba dengan menggunakan *Graphical Processing Unit* (GPU) dan *Central Processing Unit* (CPU). Berikut ini adalah salah satu hasil dari uji coba yang telah telah dilakukan. Salah satu data yang digunakan untuk uji



Gambar 5.10. Hasil ujicoba dengan variasi nilai *confidence*

Gambar 5.11. Kesalahan pengenalan

coba adalah gambar plat nomor tanpa diberi gangguan yang memuat 7 karakter yaitu W1914SN. Adapun gambar dari plat nomor yang dimaksud akan ditunjukkan pada gambar 5.12.

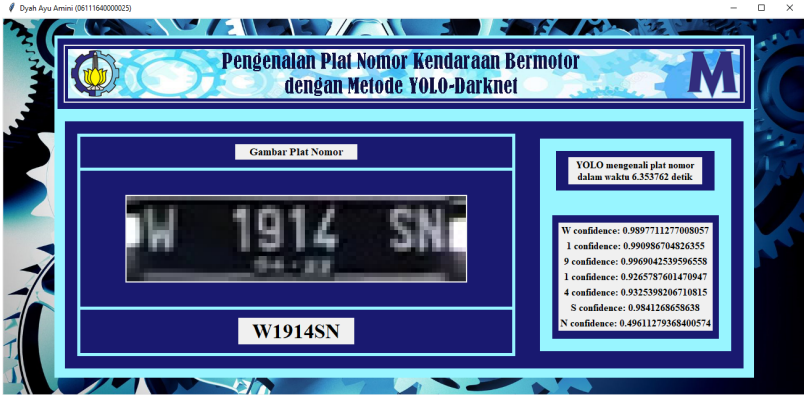
Gambar 5.12. Gambar plat nomor uji coba

Uji coba dilakukan dengan menggunakan perangkat lunak *Google Collaboration*. *Confidence* dan *threshold* yang digunakan berturut-urut adalah 0.3 dan 0.3. waktu yang diperlukan untuk mengenali plat nomor yang berisi 7 karakter tersebut adalah 1.98 detik. Adapun hasil uji coba dengan *Google Collaboration* ditampilkan pada Gambar 5.13.

```
!python3 "/content/drive/My Drive/yolo-object-detection/yolocobacoba.py"

[INFO] YOLO took 1.975502 seconds
W dengan confidence: 0.989039421081543
1 dengan confidence: 0.985913872718811
9 dengan confidence: 0.9967671036720276
1 dengan confidence: 0.9288378357887268
4 dengan confidence: 0.9362310767173767
S dengan confidence: 0.9830037355422974
N dengan confidence: 0.5497026443481445
W1914SN
```

Gambar 5.13. Hasil uji coba plat nomor dengan Google Collaboration

Uji coba selanjutnya dilakukan dengan menggunakan perangkat keras dan perangkat lunak yang ditampilkan pada Tabel 4.1 (selain *Google Collaboration*). *Confidence* dan *threshold* yang digunakan masih sama yaitu 0.3 dan 0.3. Terlihat pada gambar 5.6, waktu yang diperlukan untuk mengenali plat nomor jauh lebih lama yaitu 6.353762 detik. Hal ini terjadi karena uji coba masih dilakukan dengan menggunakan *Central Processing Unit* (CPU), sementara pada Google Collaboration uji coba telah dilakukan dengan menggunakan *Graphical Processing Unit* (GPU). Adapun hasil uji coba ditampilkan pada Gambar 5.14.



Gambar 5.14. Hasil uji coba plat nomor

5.6 Akurasi Uji Coba

Perhitungan akurasi dilakukan untuk mengetahui tingkat akurasi yang didapatkan dari hasil ujicoba metode yang telah dilakukan. Perhitungan akurasi dilakukan dengan cara menghitung jumlah hasil uji coba yang kelasnya dikenali secara tepat. Perhitungan akurasi dilakukan dengan menggunakan persamaan 5.1.

$$Akurasi = \frac{Jumlah\ pengenalan\ benar}{jumlah\ total\ pengenalan} \times 100\% \qquad (5.1)$$

Jumlah pengenalan benar adalah jumlah data uji yang kelasnya dikenali secara benar sesuai dengan kelas sebenarnya. Sedangkan jumlah total pengenalan adalah jumlah seluruh data yang diujikan.

Pada penelitian ini perhitungan akurasi dibagi menjadi dua yaitu perhitungan akurasi berdasarkan jumlah plat nomor yang dikenali dan perhitungan akurasi berdasarkan jumlah karakter yang dikenali.

5.6.1 Data tanpa *preprocessing* dan tanpa gangguan

Berdasarkan *testing* yang telah dilakukan pada data uji tanpa *preprocessing* dan tanpa gangguan, jumlah plat nomor yang dikenali berjumlah 20 buah sementara yang tidak dikenali berjumlah 5 buah plat nomor. Adapun seluruh gambar plat nomor yang diujikan untuk data tanpa gangguan berjumlah 25 buah. Berdasarkan perhitungan akurasi dengan menggunakan persamaan 5.1 didapat akurasi pengenalan plat nomor sebesar 80%

$$Akurasi\ pengenalan\ plat\ nomor = \frac{20}{25} \times 100\%$$

$$Akurasi\ pengenalan\ plat\ nomor = 80\%$$

Sedangkan jumlah karakter yang dikenali berjumlah 166 buah sementara yang tidak dikenali dengan benar berjumlah 5 buah karakter. Kesalahan yang terjadi adalah sering dikenalinya angka 0 sebagai huruf O. Adapun seluruh gambar plat nomor yang diujikan berjumlah 171 buah. Berdasarkan perhitungan akurasi dengan menggunakan persamaan 5.1 didapat akurasi pengenalan karakter sebesar 97.1%

$$Akurasi\ pengenalan\ karakter = \frac{166}{171} \times 100\%$$

$$Akurasi\ pengenalan\ karakter = 97.1\%$$

5.6.2 Data tanpa *preprocessing* dengan Gangguan Penambahan Intensitas Warna

Penambahan intensitas warna pada data uji tanpa *preprocessing* dilakukan dengan 3 tingkatan yang berbeda. Yaitu penambahan intensitas warna sebesar $25, 50$, dan 75.

Setelah melakukan *testing* pada data uji tanpa *preprocessing* dengan gangguan penambahan intensitas warna sebesar 25 didapatkan bahwa jumlah plat nomor yang dikenali sebanyak 18 buah sementara yang tidak dikenali berjumlah 7 buah. Adapun seluruh

gambar plat nomor yang diujikan untuk data dengan penambahan intensitas warna sebesar 25 tingkatan berjumlah 25 buah. Berdasarkan perhitungan akurasi dengan menggunakan persamaan 5.1 didapat akurasi pengenalan plat nomor sebesar 72%

$$Akurasi\ pengenalan\ plat\ nomor = \frac{18}{25} \times 100\%$$

$$Akurasi\ pengenalan\ plat\ nomor = 72\%$$

Sementara itu setelah melakukan *testing* pada data uji dengan gangguan penambahan intensitas warna sebesar 25 jumlah karakter yang dikenali berjumlah 164 buah sementara yang tidak dikenali dengan benar berjumlah 7 buah karakter. Adapun jumlah karakter plat nomor yang diujikan berjumlah 171 buah. Berdasarkan perhitungan akurasi dengan menggunakan persamaan 5.1 didapat akurasi pengenalan karakter sebesar 96%

$$Akurasi\ pengenalan\ karakter = \frac{164}{171} \times 100\%$$

$$Akurasi\ pengenalan\ karakter = 96\%$$

Hasil *testing* pada citra tanpa *preprocessing* dengan gangguan dengan penambahan intensitas warna sebesar 50 didapatkan jumlah plat nomor yang dikenali secara benar berjumlah 15 buah dan jumlah karakter yang dikenali secara benar berjumlah 161 buah. Berdasarkan perhitungan akurasi dengan menggunakan persamaan 5.1 didapat akurasi pengenalan plat nomor sebesar 60% dan akurasi pengenalan karakter sebesar 94.2%

$$Akurasi\ pengenalan\ plat\ nomor = \frac{15}{25} \times 100\% = 60\%$$

$$Akurasi\ pengenalan\ karakter = \frac{161}{171} \times 100\% = 94.2\%$$

Sementara itu hasil *testing* pada citra tanpa *preprocessing* dengan gangguan dengan penambahan intensitas warna sebesar 75 didapatkan jumlah plat nomor yang dikenali secara benar berjumlah

15 buah dan jumlah karakter yang dikenali secara benar berjumlah 161 buah. Berdasarkan perhitungan akurasi dengan menggunakan persamaan 5.1 didapat akurasi pengenalan plat nomor sebesar 60% dan akurasi pengenalan karakter sebesar 94.2%

$$Akurasi\ pengenalan\ plat\ nomor = \frac{15}{25} \times 100\% = 60\%$$

$$Akurasi\ pengenalan\ karakter = \frac{161}{171} \times 100\% = 94.2\%$$

Dari hasil *testing* yang didapatkan dapat disimpulkan untuk model tanpa *preprocessing* semakin besar nilai intensitas warna yang ditambahkan maka hasil akurasi pengenalan yang didapat semakin kecil. Hal ini dikarenakan objek menjadi kurang jelas ketika intensitas warna ditambahkan.

5.6.3 Data tanpa *preprocessing* dengan dengan Gangguan Pengurangan Intensitas Warna

Pengurangan intensitas warna pada data uji tanpa *preprocessing* dilakukan dengan 3 tingkatan yang berbeda. Yaitu pengurangan intensitas warna sebesar $25, 50$, dan 75.

Setelah melakukan *testing* pada data uji tanpa *preprocessing* dengan gangguan pengurangan intensitas warna sebesar 25 didapatkan bahwa jumlah plat nomor yang dikenali sebanyak 19 buah, sementara yang tidak dikenali berjumlah 6 buah. Adapun seluruh gambar plat nomor yang diujikan untuk data dengan pengurangan intensitas warna sebesar 25 berjumlah 25 buah. Berdasarkan perhitungan akurasi dengan menggunakan persamaan 5.1 didapat akurasi pengenalan plat nomor sebesar 76%

$$Akurasi\ pengenalan\ plat\ nomor = \frac{19}{25} \times 100\%$$

$$Akurasi\ pengenalan\ plat\ nomor = 76\%$$

Sementara itu jumlah karakter yang dikenali berjumlah 164 buah sementara yang tidak dikenali dengan benar berjumlah 9 buah

karakter. Adapun jumlah karakter plat nomor yang diujikan berjumlah 171 buah. Berdasarkan perhitungan akurasi dengan menggunakan persamaan 5.1 didapat akurasi pengenalan karakter sebesar 96%

$$Akurasi\ pengenalan\ karakter = \frac{164}{171} \times 100\%$$

$$Akurasi\ pengenalan\ karakter = 96\%$$

Hasil *testing* pada citra tanpa *preprocessing* dengan gangguan pengurangan intensitas warna sebesar 50 didapatkan jumlah plat nomor yang dikenali secara benar berjumlah 12 buah dan jumlah karakter yang dikenali secara benar berjumlah 155 buah. Berdasarkan perhitungan akurasi dengan menggunakan persamaan 5.1 didapat akurasi pengenalan plat nomor sebesar 48% dan akurasi pengenalan karakter sebesar 90.6%

$$Akurasi\ pengenalan\ plat\ nomor = \frac{12}{25} \times 100\% = 48\%$$

$$Akurasi\ pengenalan\ karakter = \frac{155}{171} \times 100\% = 90.6\%$$

Sementara itu hasil *testing* pada citra tanpa *preprocessing* dengan gangguan dengan pengurangan intensitas warna sebesar 75 didapatkan jumlah plat nomor yang dikenali secara benar berjumlah 7 buah dan jumlah karakter yang dikenali secara benar berjumlah 146 buah. Berdasarkan perhitungan akurasi dengan menggunakan persamaan 5.1 didapat akurasi pengenalan plat nomor sebesar 36% dan akurasi pengenalan karakter sebesar 85.3%

$$Akurasi\ pengenalan\ plat\ nomor = \frac{7}{25} \times 100\% = 36\%$$

$$Akurasi\ pengenalan\ karakter = \frac{146}{171} \times 100\% = 85.3\%$$

Dari hasil *testing* yang telah dilakukan terhadap data tanpa *preprocessing* dengan gangguan pengurangan intensitas warna didapat bahwa semakin besar pengurangan intensitas warna, maka akurasi pengenalan akan semakin kecil. Hal ini disebabkan karena semakin besar pengurangan intensitas warna, maka citra akan menjadi lebih gelap sehingga cukup sulit dibedakan dengan *background*.

5.6.4 Data dengan *preprocessing* dan tanpa gangguan

Berdasarkan *testing* yang telah dilakukan pada data uji dengan *preprocessing* dan tanpa gangguan, jumlah plat nomor yang dikenali berjumlah 22 buah, sementara yang tidak dikenali berjumlah 3 buah plat nomor. Adapun seluruh gambar plat nomor yang diujikan untuk data tanpa gangguan berjumlah 25 buah. Berdasarkan perhitungan akurasi dengan menggunakan persamaan 5.1 didapat akurasi pengenalan plat nomor sebesar 88%

$$Akurasi\ pengenalan\ plat\ nomor = \frac{22}{25} \times 100\%$$

$$Akurasi\ pengenalan\ plat\ nomor = 88\%$$

Sedangkan jumlah karakter yang dikenali berjumlah 168 buah sementara yang tidak dikenali dengan benar berjumlah 3 buah karakter. Kesalahan yang terjadi adalah sering dikenalinya huruf N sebagai H, selain itu juga dikenalinya huruf K sebagai X hal ini terjadi karena huruf yang dikenali sedikit putus-putus shingga huruf dengan bentuk yang mirip akan dikenali sebagai bentuk yang sama. Adapun seluruh gambar plat nomor yang diujikan berjumlah 171 buah. Berdasarkan perhitungan akurasi dengan menggunakan persamaan 5.1 didapat akurasi pengenalan karakter sebesar 98.2%

$$Akurasi\ pengenalan\ karakter = \frac{168}{171} \times 100\%$$

$$Akurasi\ pengenalan\ karakter = 98.2\%$$

5.6.5 Data dengan *preprocessing* dengan Gangguan Penambahan Intensitas Warna

Penambahan intensitas warna pada data uji dengan *preprocessing*dilakukan dengan 3 tingkatan yang berbeda. Yaitu penambahan intensitas warna sebesar $25, 50$, dan 75.

Setelah melakukan *testing* pada data uji dengan *preprocessing* dengan gangguan penambahan intensitas warna sebesar 25 didapatkan bahwa jumlah plat nomor yang dikenali sebanyak 22

buah sementara yang tidak dikenali berjumlah 3 buah. Adapun seluruh gambar plat nomor yang diujikan untuk data dengan penambahan intensitas warna sebesar 25 tingkatan berjumlah 25 buah. Berdasarkan perhitungan akurasi dengan menggunakan persamaan 5.1 didapat akurasi pengenalan plat nomor sebesar 88%

$$Akurasi\ pengenalan\ plat\ nomor = \frac{22}{25} \times 100\%$$

$$Akurasi\ pengenalan\ plat\ nomor = 88\%$$

Sementara itu setelah melakukan *testing* pada data uji dengan *preprocessing* dengan gangguan penambahan intensitas warna sebesar 25 jumlah karakter yang dikenali berjumlah 167 buah sementara yang tidak dikenali dengan benar berjumlah 4 buah karakter. Adapun jumlah karakter plat nomor yang diujikan berjumlah 171 buah. Berdasarkan perhitungan akurasi dengan menggunakan persamaan 5.1 didapat akurasi pengenalan karakter sebesar 97.6%

$$Akurasi\ pengenalan\ karakter = \frac{167}{171} \times 100\%$$

$$Akurasi\ pengenalan\ karakter = 97.6\%$$

Hasil *testing* pada citra dengan dengan *preprocessing* dengan gangguan penambahan intensitas warna sebesar 50 didapatkan jumlah plat nomor yang dikenali secara benar berjumlah 20 buah dan jumlah karakter yang dikenali secara benar berjumlah 165 buah. Berdasarkan perhitungan akurasi dengan menggunakan persamaan 5.1 didapat akurasi pengenalan plat nomor sebesar 80% dan akurasi pengenalan karakter sebesar 95.9%

$$Akurasi\ pengenalan\ plat\ nomor = \frac{20}{25} \times 100\% = 80\%$$

$$Akurasi\ pengenalan\ karakter = \frac{164}{171} \times 100\% = 95.9\%$$

Sementara itu hasil *testing* pada citra dengan *preprocessing* dengan gangguan penambahan intensitas warna sebesar 75 didapatkan jumlah plat nomor yang dikenali secara benar berjumlah

17 buah dan jumlah karakter yang dikenali secara benar berjumlah 161 buah. Berdasarkan perhitungan akurasi dengan menggunakan persamaan 5.1 didapat akurasi pengenalan plat nomor sebesar 68% dan akurasi pengenalan karakter sebesar 94.1%

$$Akurasi\ pengenalan\ plat\ nomor = \frac{17}{25} \times 100\% = 68\%$$

$$Akurasi\ pengenalan\ karakter = \frac{161}{171} \times 100\% = 94.1\%$$

Dari hasil *testing* yang didapatkan dapat disimpulkan untuk data dengan dengan *preprocessing* semakin besar nilai intensitas warna yang ditambahkan maka akurasi yang didapat semakin kecil, namun dibandingkan dengan model yang didapat tanpa dilakukan *preprocessing* dapat dikatakan model cukup stabil untuk mengenali plat nomor dengan gangguan penambahan intensitas warna.

5.6.6 Data dengan *Preprocessing* dengan Gangguan Pengurangan Intensitas Warna

Pengurangan intensitas warna pada data uji dengan *preprocessing*dilakukan dengan 3 tingkatan yang berbeda. Yaitu pengurangan intensitas warna sebesar $25, 50$, dan 75.

Setelah melakukan *testing* pada data uji dengan *preprocessing* dengan gangguan pengurangan intensitas warna sebesar 25 didapatkan bahwa jumlah plat nomor yang dikenali sebanyak 22 buah, sementara yang tidak dikenali berjumlah 3 buah. Adapun seluruh gambar plat nomor yang diujikan untuk data dengan pengurangan intensitas warna sebesar 25 berjumlah 25 buah. Berdasarkan perhitungan akurasi dengan menggunakan persamaan 5.1 didapat akurasi pengenalan plat nomor sebesar 88%

$$Akurasi\ pengenalan\ plat\ nomor = \frac{22}{25} \times 100\%$$

$$Akurasi\ pengenalan\ plat\ nomor = 88\%$$

Sementara itu jumlah karakter yang dikenali berjumlah 168 buah sementara yang tidak dikenali dengan benar berjumlah 4 buah karakter. Adapun jumlah karakter plat nomor yang diujikan berjumlah 171 buah. Berdasarkan perhitungan akurasi dengan menggunakan persamaan 5.1 didapat akurasi pengenalan karakter sebesar 98.2%

$$Akurasi\ pengenalan\ karakter = \frac{168}{171} \times 100\%$$

$$Akurasi\ pengenalan\ karakter = 98.2\%$$

Hasil *testing* pada citra dengan *preprocessing* dengan gangguan pengurangan intensitas warna sebesar 50 didapatkan jumlah plat nomor yang dikenali secara benar berjumlah 21 buah dan jumlah karakter yang dikenali secara benar berjumlah 167 buah. Berdasarkan perhitungan akurasi dengan menggunakan persamaan 5.1 didapat akurasi pengenalan plat nomor sebesar 84% dan akurasi pengenalan karakter sebesar 97.7%

$$Akurasi\ pengenalan\ plat\ nomor = \frac{14}{25} \times 100\% = 84\%$$

$$Akurasi\ pengenalan\ karakter = \frac{167}{171} \times 100\% = 97.7\%$$

Sementara itu hasil *testing* pada citra dengan *preprocessing* dengan gangguan pengurangan intensitas warna sebesar 75 didapatkan jumlah plat nomor yang dikenali secara benar berjumlah 21 buah dan jumlah karakter yang dikenali secara benar berjumlah 166 buah. Berdasarkan perhitungan akurasi dengan menggunakan persamaan 5.1 didapat akurasi pengenalan plat nomor sebesar 84% dan akurasi pengenalan karakter sebesar 97%

$$Akurasi\ pengenalan\ plat\ nomor = \frac{21}{25} \times 100\% = 84\%$$

$$Akurasi\ pengenalan\ karakter = \frac{166}{171} \times 100\% = 97\%$$

Dari hasil *testing* yang didapatkan dapat disimpulkan untuk data dengan dengan *preprocessing* semakin besar nilai intensitas warna yang dikurangkan, akurasi yang didapat cukup baik, dapat dikatakan bahwa model cukup stabil untuk mengenali plat nomor dengan gangguan pengurangan intensitas warna.

BAB VI
KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan yang dihasilkan berdasarkan penelitian yang telah dilaksanakan serta saran untuk pengembangan penelitian ini.

6.1 Kesimpulan

1. Pada penelitian ini, telah berhasil dibangun proses pengenalan plat nomor kendaraan bermotor dengan metode YOLO-Darknet. Adapun tahapan-tahapan yang dilakukan adalah melakukan persiapan data, *preprocessing*, Anotasi data, *training*, serta *testing*.

2. Akurasi yang diperoleh untuk model yang didapat dari *training* data tanpa *preprocessing*

   (a) Dataset tanpa gangguan menghasilkan nilai akurasi pengenalan plat nomor paling tinggi sebesar 80%, dengan akurasi pengenalan karakter sebesar 97.1%.

   (b) Dataset dengan gangguan efek cerah atau penambahan intensitas warna menghasilkan nilai akurasi pengenalan plat nomor paling tinggi sebesar 72%, dengan akurasi pengenalan karakter sebesar 96%.

   (c) Dataset dengan gangguan efek gelap atau pengurangan intensitas warna, menghasilkan nilai akurasi pengenalan plat nomor tertinggi sebesar 76% dengan akurasi pengenalan karakter sebesar 96%.

3. Akurasi yang diperoleh untuk model yang didapat dari *training* data dengan dilakukan *preprocessing*

   (a) Dataset tanpa gangguan menghasilkan nilai akurasi pengenalan plat nomor paling tinggi sebesar 88%, dengan akurasi pengenalan karakter sebesar 98.2%.

(b) Dataset dengan gangguan efek cerah atau penambahan intensitas warna menghasilkan nilai akurasi pengenalan plat nomor paling tinggi sebesar 88%, dengan akurasi pengenalan karakter sebesar 97.6%.

(c) Dataset dengan gangguan efek gelap atau pengurangan intensitas warna, menghasilkan nilai akurasi pengenalan plat nomor tertinggi sebesar 88% dengan akurasi pengenalan karakter sebesar 98.2%.

6.2 Saran

Saran yang dapat diberikan untuk penelitian serupa adalah memperbanyak jumlah data proses *training*. Sehingga program dapat mengenali objek dalam berbagai macam kondisi. Selain itu diharapkan dapat melakukan variasi pada proses *preprocessing* sehingga kualitas data yang digunakan dapat menjadi lebih baik.

DAFTAR PUSTAKA

[1] Redmon Joseph, Santosh Divvala, Ross Girshick, Ali Farhadi. (2016). ”You Only Look Once: Unified, Real-Time Object Detection”. **University of Washington, Allen Institute for AI, Facebook AI Research**.

[2] Joseph Redmon, Ali Farhadi. (2017). ”YOLO9000: Better, Faster, Stronger”. **In Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, pages 6517–6525. IEEE, 2017.**

[3] Joseph Redmon, Ali Farhadi. (2018). ”Yolov3: An incremental improvement”. **arXiv, 2018**

[4] Christya Aji Putra. (2015). ”Pengenalan Karakter Plat Nomor Kendaraan Bermotor Menggunakan Extreme Learning Machine”. **TESIS – TE42599 Institut Teknologi Sepuluh Nopember, Surabaya**.

[5] Hendry, Rung-Ching Chen. (2019) ”Automatic License Plate Recognition via sliding-window-darknet-YOLO Deep Learning” **Image and Vision Computing 87 (2019) 47-56**

[6] C. Anagnostopoulos, I. Anagnostopoulos, E. Kayafas, V. Loumos and I. Psoroulas, (2008) “License plate recognition from still images and video sequences,” **IEEE Trans. Intell. Transp. Syst.**, vol. 9, no. 3, Sep. 2008.

[7] Cheriet, M, dkk,. (2007) ”Character Recognition Systems”, **John Willey and sons inc**. publication, 2007

[8] Gonzalez, Rafael C. and Woods, Richard E. (2002) ”Digital Image Processing”, **Prentice Hall. New Jersey, 2002**.

[9] Undang-Undang Republik Indonesia Nomor 22 Tahun 2009 Tentang Lalu Lintas Dan Angkutan Jalan, 2009.

[10] Pramestya, R.H. (2018). ”Deteksi dan Klasifikasi Kerusakan Jalan Aspal Menggunakan Metode YOLO Berbasis Citra Digital”. **Tesis. Fakultas Matematika, Komputasi dan Sains Data. Institut Teknologi Sepuluh Nopember : Surabaya**.

[11] Syed Zain Masood, Guang Shu, Afshin Dehghan, Enrique G. Ortiz. (2017). ”License Plate Detection and Recognition Using Deeply Learned Convolutional Neural Networks”. **Computer Vision Lab, Sighthound Inc., Winter Park, FL**.

[12] Hui Li, Chunhua Shen. (2016). ”Reading Car License Plates Using Deep Convolutional Neural Networks and LSTMs”. **arXiv:1601.05610v1 [cs.CV]**.

[13] Rayson Laroca, Evair Severo, Luiz A. Zanlorensi, Luiz S. Oliveira, Gabriel Resende Goncalves, William Robson Schwartz, David Menotti. (2018). ”A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector”. **International Joint Conference on Neural Networks (IJCNN)**.

[14] Zitnick, C. L. dan Dollar, P. (2014),” Edge boxes: Location object proposals from edge”,**European conference on computer vision**, Springer, pp. 391-405.

[15] Everingham, M., Van Gool, L., Williams, C. K., Winn, J. dan Zisserman, A. (2010). ”The Pascal Visual Object Classes (VOC) Challenge”, **International Journal of Computer Vision 88(2)**, 303-338.

[16] Stanford University, ”An Introduction to Convolutional Neural Network,” **Vision Imaging Science and Technology Lab, Stanford University**.

[17] Muhammad Zufar, Budi Setiyono. (2016). ”Convolutional Neural Networks untuk Pengenalan Wajah Secara Real-Time”,

JURNAL SAINS DAN SENI ITS Vol. 5 No. 2 (2016) 2337-3520 (2301-928X Print)

[18] Geethapriya. S, N. Duraimurugan, S.P. Chokkalingam. (2019). ”Real-Time Object Detection with Yolo”. International **Journal of Engineering and Advanced Technology (IJEAT)**, ISSN: 2249 – 8958, Volume-8, Issue-3S.

[19] Sergey Ioffe, Christian Szegedy (2015). ”Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. arXiv:1502.03167v3 [cs.LG].

[20] Shreyas Fadnavis (2014). ”Image Interpolation Techniques in Digital Image Processing: An Overview”. **Journal of Engineering Research and Applications**, ISSN : 2248-9622, Vol. 4, Issue 10(Part -1), October 2014, pp.70-73.