TUGAS AKHIR – KM184801

DETEKSI PLAT NOMOR KENDARAAN BERGERAK BERBASIS METODE YOU ONLY LOOK ONCE (YOLO)

ARIO FAJAR PRATAMA
06111640000087

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

Departemen Matematika
Fakultas Sains dan Analitika Data
Institut Teknologi Sepuluh Nopember
Surabaya 2020

TUGAS AKHIR – KM184801

DETEKSI PLAT NOMOR KENDARAAN BERGERAK BERBASIS METODE YOU ONLY LOOK ONCE (YOLO)

ARIO FAJAR PRATAMA
06111640000087

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

Departemen Matematika
Fakultas Sains dan Analitika Data
Institut Teknologi Sepuluh Nopember
Surabaya 2020

TUGAS AKHIR – KM184801

VEHICLE LICENSE PLATE DETECTION BASED ON YOU ONLY LOOK ONCE (YOLO) METHOD

ARIO FAJAR PRATAMA
06111640000087

Supervisor
Dr. Budi Setiyono, S.Si, MT

Department of Mathematics
Faculty of Science and Data Analytics
Sepuluh Nopember Institute of Technology
Surabaya 2020

LEMBAR PENGESAHAN

DETEKSI PLAT NOMOR KENDARAAN BERGERAK BERBASIS METODE YOU ONLY LOOK ONCE (YOLO)

VEHICLE LICENSE PLATE DETECTION BASED ON YOU ONLY LOOK ONCE (YOLO) METHOD

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
Untuk memperoleh gelar Sarjana Matematika
Pada bidang studi Pemrograman dan Komputasi Visual
Program Studi S1 Departemen Matematika
Fakultas Sains, dan, Analitika Data
Institut Teknologi Sepuluh Nopember Surabaya
Oleh:
Ario Fajar Pratama
NRP. 06111640000087

Menyetujui,
Dosen Pembimbing

Dr. 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, Juni 2020

DETEKSI PLAT NOMOR KENDARAAN BERGERAK BERBASIS METODE *YOU ONLY LOOK ONCE* (YOLO)

Nama Mahasiswa	:	Ario Fajar Pratama
NRP	:	06111640000087
Departemen	:	Matematika FSAD-ITS
Pembimbing	:	Dr. Budi Setiyono, S.Si, MT

Abstrak

Semakin berkembangnya alat transportasi yang digunakan manusia, dibutuhkan sistem pengawasan lalu lintas yang canggih yang disebut dengan Intelligent Transportation System (ITS). Salah satu bagian dari ITS adalah sistem pendeteksian plat nomor kendaraan. Banyak penelitian sebelumnya berbasis deeplearning telah dilakukan dalam pendeteksian objek benda, salah satunya adalah You Only Look Once (YOLO). Oleh sebab itu, pada penelitian ini dilakukan deteksi plat nomor kendaraan bergerak berbasis metode YOLO. Alur pendeteksian plat nomor ini diantaranya input video, akuisisi, input ROI, proses YOLO, tracking, memilih confidence tertinggi dan cropping. Berdasarkan penelitian ini didapatkan Mean Average Precision (mAP%) 87.89% dan hasil deteksi lokasi plat nomor yang terlihat secara empiris memiliki rata-rata akurasi sebesar 85.81%

Kata-kunci: *YOLOV3, Deep Learning, Pengolahan Citra, Tracking*

VEHICLE LICENSE PLATE DETECTION BASED ON YOU ONLY LOOK ONCE (YOLO) METHOD

Name	:	Ario Fajar Pratama
NRP	:	06111640000087
Department	:	Mathematics FMCDS-ITS
Supervisor	:	Dr. Budi Setiyono, S.Si, MT

Abstract

The more developed means of transportation used by humans, we need a sophisticated traffic control system called Intelligent Transportation System (ITS). One part of ITS is the vehicle number plate detection system. Many previous studies based on deep learning have been conducted in object detection, one of which is You Only Look Once (YOLO). Therefore, in this study, the detection of vehicle number plates based on the YOLO method. The flow detection of this number is published video input, obtained, input ROI, YOLO process, tracking, select the highest confidence and cropping. Based on this research, it was obtained Mean Average Precision (mAP%) 87.89% and the results of detection license plates seen by empiricists had an average evaluation of 85.81%

Keywords: *YOLOV3, Deep Learning, Image Processing, Tracking*

KATA PENGANTAR

Assalamu'alaikum Wr. Wb

Puji syukur kehadirat Allah SWT karena atas berkah, rahmat dan Ridho-Nya sehingga penulis dapat menyelesaikan tugas akhir dengan judul :

"DETEKSI PLAT NOMOR KENDARAAN BERGERAK BERBASIS METODE *YOU ONLY LOOK ONCE* (YOLO)"

sebagai salah satu persyaratan akademis dalam menyelesaikan Program Sarjana Departemen Matematika, Fakultas Sains dan Analitika Data, Institut Teknologi Sepuluh Nopember Surabaya.

Tugas Akhir ini dapat diselesaikan dengan baik berkat kerja sama, bantuan, dan dukungan dari banyak pihak. Sehubungan dengan hal itu, penulis ingin mengucapkan terima kasih dan penghargaan kepada :

1. Orang tua yang telah memberikan *support*, nasehat dan motivasi sehingga penulis dapat menyelesaikan tugas akhir ini

2. Bapak Dr. Budi Setiyono, S.Si, MT selaku dosen pembimbing yang telah memberikan arahan dengan penuh kesabaran kepada penulis selama penyusunan Tugas Akhir ini.

3. Ibu Dian Winda Setyawati, S.Si, M.Si selaku dosen wali yang telah memberikan arahan dengan penuh kesabaran kepada penulis selama masa perkuliahan dan selama penyusunan Tugas Akhir ini.

4. Bapak Subchan, Ph.D selaku Kepala Departemen Matematika Institut Teknologi Sepuluh Nopember yang selalu memberikan arahan dan bimbingan selama perkuliahan hingga terselesaikannya Tugas Akhir ini.

5. Ibu Dr. Dwi Ratna Sulistyaningrum, S.Si, MT selaku Sekretaris Departemen Bidang Akademik yang selalu memberikan dukungan dan senantiasa mengingatkan penulis untuk menyelesaikan Tugas AKhir ini.

6. Bapak/Ibu dosen pengajar yang tidak bisa penulis sebutkan satu per satu, yang telah memberikan ilmu dan pengalaman yang bermanfaat kepada penulis, serta segenap Karyawan, Tendik dan keluarga besar Departemen Matematika Institut Teknologi Sepuluh Nopember atas dukungan dan bantuannya.

7. Teman - teman seperjuangan, Matematika Institut Teknologi Sepuluh Nopember Angkatan 2016 maupun Lemniscate yang telah mengisi hari - hari penulis dengan penuh keceriaan, motivasi dan pengalaman.

8. Kabinet CIPTA Kepengurusan HIMATIKA ITS 2018/2019 beserta seluruh fungsionaris yang telah memberikan cerita, pengalaman dan kekeluargaan bagi penulis

9. KM ITS dan Seluruh Civitas Akademika ITS yang telah memberi banyak ilmu dan pelajaran bagi penulis

10. Sahabat/kawan warkop yang telah bersedia menemani penulis berbagi cerita dan pengetahuan akan wawasan serta berbagai inspirasi

Penulis menyadari bahwa Tugas Akhir ini masih jauh dari kesempurnaan. Oleh karena itu, penulis mengharapkan saran

dan kritik dari pembaca. Akhir kata, semoga Tugas Akhir ini bermanfaat bagi semua pihak yang berkepentingan.
Wassalamu'alaikum Wr. Wb.

Surabaya, Juni 2020

Penulis

DAFTAR ISI

LEMBAR PENGESAHAN i

ABSTRAK iii

ABSTRACT v

KATA PENGANTAR vii

DAFTAR ISI xi

DAFTAR GAMBAR xv

BAB I PENDAHULUAN 1
- 1.1 Latar Belakang 1
- 1.2 Rumusan Masalah 4
- 1.3 Batasan Masalah 4
- 1.4 Tujuan 5
- 1.5 Manfaat 5
- 1.6 Sistematika Penulisan 5

BAB II TINJAUAN PUSTAKA 7
- 2.1 Penelitian Terdahulu 7
- 2.2 Pengolahan Citra 10
- 2.3 Jenis Citra Digital 10
  - 2.3.1 *Color Image* atau RGB (*Red, Green, Blue*) 10
  - 2.3.2 *Black and White* 11
- 2.4 Segmentasi Warna Normalisasi RGB 11
- 2.5 Plat Nomor Kendaraan 12
- 2.6 *You Only Look Once* (YOLO) 14
- 2.7 Kelebihan dan Kekurangan YOLO 15
- 2.8 *Intersection over Union* (IoU) 17

2.9 Desain Jaringan 18
2.10 *Loss function of YOLO Method* 19
2.11 *Convolutional Neural Network* (CNN) 22
2.12 *Convolutional Layer* 24
2.13 *Confusion Matrix* 25

BAB III METODE PENELITIAN 27
3.1 Studi Literatur 28
3.2 Pengumpulan Data 28
3.3 Proses Pelatihan 28
3.4 Perancangan Program 31

BAB IV PERANCANGAN DAN IMPLEMENTASI SISTEM 35
4.1 Analisis Sistem 35
4.1.1 Analisis Fungsional Sistem 35
4.1.2 Analisis non-Fungsional Sistem 36
4.2 Perancangan Data 36
4.2.1 Data Inputan 37
4.2.2 Data Proses 37
4.2.3 Data Output 38
4.3 Perancangan Sistem 39
4.3.1 Persiapan Data 39
4.4 Parameter *Training* 45
4.5 Proses Pendeteksian Plat Nomor Kendaraan . 46
4.5.1 Akuisisi Video 48
4.5.2 Pendefinisian *Region of Interest* (ROI) 49
4.5.3 Deteksi Objek Plat dengan Arsitektur Jaringan YoloV3 52
4.5.4 Output Processing (Pemfilteran dengan nilai ambang pada nilai kelas) . 97
4.5.5 YoloV3 Loss Function 99
4.5.6 Back Propagation 102
4.5.7 *Tracking* Objek Plat Nomor 116

4.5.8 Menentukan Lokalisasi Plat 120
4.6 Implementasi Sistem 122
4.6.1 Implementasi Tahap Training 122
4.6.2 Implementasi Tahap Pengujian 133

BAB V Uji Coba dan Pembahasan 139
5.1 Data Uji Coba 139
5.2 Hasil Uji Coba 141
5.2.1 Hasil Uji Coba Data Valid 141
5.2.2 Pembahasan Hasil Uji Coba Data Valid 143
5.2.3 Hasil Uji Coba Data *Testing* 146
5.2.4 Pembahasan Uji Coba Data *Testing* . . 149
5.2.5 Hasil Uji Coba dan Pembahasan Pendefinisian ROI 160

BAB VI PENUTUP 165
6.1 Kesimpulan 165
6.2 Saran 166

DAFTAR PUSTAKA 167

BIODATA PENULIS 171

DAFTAR GAMBAR

Gambar 2.1 Model Sistem YOLO [4]. 14
Gambar 2.2 Sistem Deteksi YOLO [4]. 15
Gambar 2.3 Ilustrasi Perhitungan IoU 17
Gambar 2.4 Arsitektur dasar YOLO [4] 18
Gambar 2.5 Arsitektur MLP Sederhana 22
Gambar 2.6 Proses Konvolusi pada CNN 23
Gambar 2.7 Proses Konvolusi 25

Gambar 3.1 Blok Diagram Pengerjaan Tugas Akhir 27
Gambar 3.2 YOLOV3 Processing 32
Gambar 3.3 Tahapan Perancangan Program 33

Gambar 4.1 *Layout* Perekaman Video 37
Gambar 4.2 Diagram Alir Anotasi Data 40
Gambar 4.3 Dataset *Image* 41
Gambar 4.4 Proses Tampilkan Citra dari Dataset . . 41
Gambar 4.5 *Labelling* Kotak Pembatas Secara Manual 42
Gambar 4.6 Citra dengan Koordinat Kotak Pembatas 42
Gambar 4.7 Keterangan Kotak Pembatas 43
Gambar 4.8 Keterangan Kotak Pembatas yang Baru 44
Gambar 4.9 Blok Diagram Proses Pendeteksian Plat 47
Gambar 4.10 Proses Akuisis Video 49
Gambar 4.11 Frame Setelah Didefinisikan ROI 51
Gambar 4.12 Frame ROI 52
Gambar 4.13 Arsitektur Jaringan YoloV3 53
Gambar 4.14 Proses Input Image pada YoloV3 56
Gambar 4.15 Proses input *image* menjadi *image* RGB 58

Gambar 4.16 Contoh Proses Zero Padding 60
Gambar 4.17 Proses *Convolutional 2D Layer* pada Layer Pertama 61
Gambar 4.18 Proses Konvolusi pada Posisi $(0, 0)$ . . . 64
Gambar 4.19 Proses Konvolusi pada Posisi $(0, 1)$ . . . 65
Gambar 4.20 Hasil Konvolusi pada setiap pixel 65
Gambar 4.21 Hasil *Convolutional 2D Layer* pada Layer Pertama 66
Gambar 4.22 Internal *Covariate Shift* 67
Gambar 4.23 Ilustrasi *Neural Network* pada layer Pertama 68
Gambar 4.24 Gambar yang Akan Dilakukan Batch Normalization 70
Gambar 4.25 Proses *Batch Normalization* pada *Neural Network* 80
Gambar 4.26 Perbedaan Grafik dari Fungsi ReLu dan Leaky ReLU 81
Gambar 4.27 Ilustrasi Proses LeakyReLU 82
Gambar 4.28 Proses *Shortcut Layer* pada Jaringan YoloV3 84
Gambar 4.29 Ilustrasi Proses *Upsample Layer* 86
Gambar 4.30 Proses *Route Layer* dengan parameter $(route = -4)$ 87
Gambar 4.31 Proses *Route Layer* dengan parameter $(route = (-1, 61))$ 88
Gambar 4.32 Diagram Alir dan Struktur Jaringan YoloV3 91
Gambar 4.33 Perbandingan Deteksi pada 3 Skala Jaringan YoloV3 92
Gambar 4.34 Hasil Deteksi tiap Sel Grid 94
Gambar 4.35 *Anchor Boxes* Pada Jaringan YoloV3 . 95
Gambar 4.36 Proses Deteksi menggunakan *Anchor Boxes* Pada Jaringan YoloV3 96

Gambar 4.37 Proses *Intersection over Union* 98
Gambar 4.38 Proses *Non-maksimum Suppression* . . . 99
Gambar 4.39 Proses *Loss Function* dari YoloV3 100
Gambar 4.40 Neural Network Training 103
Gambar 4.41 Tracking Objek Plat pada Deteksi Pertama 117
Gambar 4.42 Proses Rumus Jarak Euclid 118
Gambar 4.43 Proses Tracking Objek Plat Nomor . . . 119
Gambar 4.44 Proses Lokalisasi Objek Plat Nomor . . 121
Gambar 4.45 Hasil Cropping Plat Nomor 122
Gambar 4.46 Anotasi Data Menggunakan labelimg v1.7.0 123
Gambar 4.47 Halaman Utama 134

Gambar 5.1 Proses Menentukan *Trained Weights* . . 142
Gambar 5.2 Grafik mAP% pada Proses Training . . 143
Gambar 5.3 Grafik Nilai Evaluasi Model *Trained Weights* 146
Gambar 5.4 Hasil Deteksi Plat Nomor Pada Video 2 147
Gambar 5.5 Hasil Deteksi Plat Nomor Pada Video 3 148
Gambar 5.6 Hasil Deteksi Plat Nomor Pada Video 4 149
Gambar 5.7 Contoh Salah Deteksi 152
Gambar 5.8 Deteksi Plat Nomor Tanpa Karakter . . 159
Gambar 5.9 Ilustrasi Perbandingan ROI (a.) dan (b.) 161
Gambar 5.10 Hasil Perbandingan antara ROI besar dan ROI Kecil 162

BAB I
PENDAHULUAN

Pada bab ini akan dijelaskan latar belakang masalah, rumusan masalah, tujuan, dan manfaat. Subbab latar belakang masalah menjelaskan riwayat penelitian dan alasan dilakukannya penelitian, sedangkan subbab rumusan masalah berisi masalah – masalah yang akan dikaji kemudian jawaban dari masalah tersebut akan dijelaskan pada subbab tujuan. Adapun subbab batasan masalah berisi batasan – batasan yang membatasi kajian, sedangkan pada bagian manfaat berisi tentang manfaat diadakannya penelitian ini.

1.1 Latar Belakang

Seiring dengan pesatnya perkembangan ilmu pengetahuan dan teknologi saat ini menyebabkan banyak bermunculan inovasi yang semakin memudahkan manusia untuk melakukan kegiatan sehari-harinya. Salah satu yang mengalami perkembangan pesat adalah sistem pengawasan lalu lintas yang disebut dengan *Intelligent Transportation System* (ITS). ITS diperlukan untuk memantau lalu lintas jalan, merekam informasi kendaraan, melaporkan kejadian lalu lintas dan lain sebagainya. Salah satu bentuk dari ITS ini adalah sistem pengenalan plat nomor kendaraan. Sistem pengenalan plat nomor kendaraan sangatlah penting untuk dikembangkan karena plat nomor kendaraan merupakan identitas dari suatu kendaraan yang bersifat tunggal/unik (tidak ada yang sama). Dengan dikenalinya identitas plat nomor kendaraan, akan mempermudah pelacakan pada sebuah kendaraan dan dapat merekam informasi yang dilakukan oleh kendaraan tersebut.

Salah satu langkah yang penting sebelum sistem pengenalan plat nomor kendaraan adalah mendeteksi lokasi dari plat nomor tersebut. Langkah ini harus dilakukan dengan akurat karena akan berpengaruh pada proses – proses selanjutnya. Pada proses mendeteksi plat nomor kendaraan dapat dilakukan dengan menggunakan pengolahan citra digital. Pengolahan citra digital memungkinkan dapat melakukakn proses penarikan informasi atau deskripsi objek atau pengenalan objek yang terkandung dalam citra. Dengan teknik pengolahan citra digital ini, diharapkan dalam pencatatan dan pendeteksian plat nomor kendaraan tidak lagi menggunakan cara konvensional, yaitu dengan mencatat plat nomor satu per satu secara manual. Dalam proses deteksi plat nomor, data yang diperoleh akan dianalisis dan diproses sehingga menghasilkan suatu informasi yang dibutuhkan. Adapun data yang dibutuhkan untuk deteksi plat nomor kendaraan dapat berupa gambar maupun video. Pendeteksian plat nomor kendaraan dengan menggunakan video cenderung lebih baik dari gambar dikarenakan video memungkinkan pendeteksian plat nomor kendaraan secara terus menerus.

Berdasarkan dengan permasalahan di atas, maka dapat disimpulkan bahwa diperlukan suatu penelitian untuk mendeteksi plat nomor kendaraan dari sebuah video, sehingga dari hasil pengolahan citra digital tersebut dapat memberikan informasi yang berguna bagi pihak – pihak yang membutuhkan. Adapun beberapa penelitian yang telah dilakukan untuk mendeteksi plat nomor kendaraan diantaranya, 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 rekognisi plat nomor pada sebuah gambar mobil. Terinspirasi oleh keberhasilan *Deep*

Neural Networks (DNN) dalam berbagai *vision applications*, manfaat dari DNN ini dapat dipelajari untuk fitur-fitur tingkat tinggi dalam kerangka kaskade, yang mengarah pada peningkatan kinerja pada deteksi dan rekognisi[1]. Kemudian penelitian dilakukan oleh Yonten Jamtsho, Panomkhawn Riyamongkol, Rattapoom Waranusast (2019) yang berjudul "*Real-time Bhutanese license plate localization using YOLO*". Penelitian ini membahas kinerja YOLOV2 dalam mendeteksi lokasi plat nomor mobil di Negara Bhutan dari sebuah video. [2]

Selanjutnya penelitian juga dilakukan oleh Rayson Laroca, et al. (2018) tentang *"A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector"*, penelitian ini menyajikan sistem *Automatic License Plate Recognition* (ALPR) yang kuat dan efisien berdasarkan pada detektor YOLO yang canggih. *Convolutional Neural Network* dilatih dan diselesaikan untuk setiap tahap ALPR sehingga kuat dalam berbagai kondisi [3]. 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 mencapai 88% dalam *ImageNet* 2012 *Validation* [4]. Adapun beberapa penelitian yang sudah dilakukan di Indonesia diantaranya penelitian yang dilakukan oleh Ravy Hayu Pramesetya (2018) tentang "Deteksi dan Klasifikasi Kerusakan Jalan Aspal Menggunakan Metode YOLO Berbasis Citra Digital", penelitian ini menangani tentang deteksi dan klasifikasi citra kerusakan jalan dengan menggunkan metode YOLOV2 [6]. Kemudian penelitian yang dilakukan oleh Wahyu Ardiansyah (2019) tentang "Peningkatan Performansi Metode *Harris Corner* untuk Deteksi Plat Nomor Pada Video Menggunakan

Maximally Stable External Region", penelitian ini menangani permasalahan deteksi plat nomor kendaraan bergerak dari objek sebuah mobil melalui video dengan menggunakan *Maximally Stablle External Region* [7].

Berdasarakan hasil penelitian tersebut dapat disimpulkan bahwa penerapan metode YOLO khususnya YOLOV3 dalam pendeteksian lokasi plat nomor kendaraan bergerak sampai saat ini masih belum banyak digunakan. Padahal YOLOV3 terbukti merupakan metode yang lebih efisien dibandingkan dengan metode *machine learning* lainnya [5]. Maka dari itu, penulis tertarik menggunakan metode YOLOV3 untuk mendeteksi lokasi plat nomor kendaraan yang bergerak.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diberikan pada subbab sebelumnya, didapatkan beberapa rumusan masalah pada penelitian ini sebagai berikut :

1. Bagaimana merancang sistem yang dapat mendeteksi plat nomor kendaraan bergerak yang melintasi jalan raya melalui video ?

2. Bagaimana mengetahui kinerja dengan menggunakan metode YOLOV3 dalam pendeteksian lokalisasi plat nomor kendaraan bergerak ?

1.3 Batasan Masalah

Berdasarkan rumusan masalah yang telah diberikan sebelumnya, didapatkan batasan masalah tugas akhir ini dalam pendeteksian plat nomor kendaraan bergerak antara lain :

1. Data yang digunakan adalah video lalu lintas kendaraan di jalan raya pada waktu siang hari dalam kondisi yang cerah.

2. Video diambil di jalur satu arah dengan arah kendaraan mendekati kamera.

3. Plat nomor yang dideteksi adalah plat nomor dengan ukuran dan posisi pemasangan yang standard

4. Plat nomor yang terdeteksi berdasarkan lokalisasi plat nomor yang terpasang pada kendaraan.

1.4 Tujuan

Berdasarkan rumusan masalah yang telah ada, diperoleh tujuan tugas akhir ini dalam pendeteksian plat nomor kendaraan bergerak sebagai berikut :

1. Merancang sistem untuk mendeteksi plat nomor kendaraan bergerak yang melintasi jalan raya melalui video.

2. Mengetahui kinerja metode YOLOV3 dalam pendeteksian lokalisasi plat nomor kendaraan bergerak.

1.5 Manfaat

Manfaat yang akan diperoleh dari penelitian ini adalah dapat membantu pihak kepolisian dalam mendapatkan informasi berupa deteksi plat nomor kendaraan melalui sebuah video.

1.6 Sistematika Penulisan

Pada tugas akhir ini terdiri dari lima bab yaitu: Pendahuluan, Tinjauan Pustaka, Metode Penelitian, Analisis dan Pembahasan, dan Penutup. Berikut adalah deskripsi mengenai masing-masing bab pada tugas akhir ini sebagai berikut :

1. BAB I PENDAHULUAN
   Pada bab ini dijelakan tentang latar belakang, rumusan masalah, batasan masalah, tujuan, dan manfaat dari penulisan tugas akhir serta sistematika penulisan.

2. BAB II TINJAUAN PUSTAKA
   Pada bab ini berisi tentang beberapa bahan materi yang mendukung dan berkaitan tentang penulisan tugas akhir, seperti riwayat penelitian terdahulu, pengolahan citra digital, CNN, struktur jaringan YOLOV3, *confusion matrix*, dan *loss function*.

3. METODELOGI PENELITIAN
   Pada bab ini menjelaskan tahapan-tahapan yang dilakukan dalam pengerjaan penelitian atau tugas akhir mulai dari studi litelatur sampai penarikan kesimpulan dan penulisan buku tugas akhir.

4. ANALISIS DAN PEMBAHASAN
   Pada bab ini dijelaskan lebih detail tentang cara mendeteksi plat nomor kendaraan bergerak seperti akuisisi video, pendefinisian ROI, deteksi objek plat nomor menggunakan YOLOV3, *tracking*, menentukan *confidence* terbesar dan *cropping* plat nomor kendaraan.

5. PENUTUP
   Pada bab ini berisi kesimpulan dari apa yang dihasilkan pada bab analisis dan pembahasan sebelumnya, kemudian terdapat saran untuk pengembangan dan penelitian-penelitian yang akan datang.

BAB II
TINJAUAN PUSTAKA

Pada bagian ini dijelaskan dasar – dasar teori yang dapat menunjang penyelesaian masalah dalam penelitian ini.

2.1 Penelitian Terdahulu

Seiring berjalannya waktu, beberapa 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 rekognisi plat nomor pada sebuah gambar mobil. Terinspirasi oleh keberhasilan *Deep Neural Networks* (DNN) dalam berbagai *vision applications*, manfaat dari DNN ini dapat dipelajari untuk fitur-fitur tingkat tinggi dalam kerangka kaskade, yang mengarah pada peningkatan kinerja pada deteksi dan rekognisi. *Recurrent Neural Network* (RNN) memiliki panjang memori jangka pendek yang dilatih untuk mengenali fitur berurutan ekstraksi dari seluruh plat melalui CNN. Keuntungan utama dari pendekatan ini adalah segmentasi yang mudah. Dengan menjelajahi informasi konteks dan menghindari kesalahan yang disebabkan oleh segmentasi, Metode RNN berkinerja lebih baik daripada metode dasar yang menggabungkan segmentasi dan klasifikasi CNN dalam mencapai akurasi pengenalan yang canggih [1]. Kemudian penelitian dilakukan oleh Yonten Jamtsho, et al. (2019) yang berjudul "*Real-time Bhutanese license plate localization using YOLO*". Penelitian ini membahas kinerja YOLOV2 dalam mendeteksi lokasi plat nomor mobil di Negara Bhutan dari sebuah video. [2]

Selanjutnya penelitian juga dilakukan oleh Rayson Laroca, et al. (2018) tentang "*A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector*", penelitian ini menyajikan *Automatic License Plate Recognition* (ALPR) yang kuat dan efisien sistem berdasarkan pada detektor objek YOLO yang canggih. *Convolutional Neural Network* dilatih dan diselesaikan untuk setiap tahap ALPR sehingga kuat dalam berbagai kondisi (Rayson Laroca et al., 2018). 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 mencapai 88% dalam *ImageNet* 2012 *Validation* [4]. Adapun beberapa penelitian yang sudah dilakukan di Indonesia diantaranya penelitian yang dilakukan oleh Ravy Hayu Pramesetya (2018) tentang "Deteksi dan Klasifikasi Kerusakan Jalan Aspal Menggunakan Metode YOLO Berbasis Citra Digital", penelitian ini menangani tentang deteksi dan klasifikasi citra kerusakan jalan dengan menggunkan metode YOLOV2 [6]. Kemudian penelitian yang dilakukan oleh Wahyu Ardiansyah (2019) tentang "Peningkatan Performansi Metode *Harris Corner* untuk Deteksi Plat Nomor Pada Video Menggunakan *Maximally Stable External Region*", penelitian ini menangani permasalahan deteksi plat nomor kendaraan bergerak dari objek sebuah mobil melalui video dengan menggunakan *Maximally Stablle External Region* [7]. Adapun posisi penelitian ini jika dibandingkan dengan penelitian di atas dapat dilihat pada Tabel 2.1 seperti di bawah ini.

Tabel 2.1: Posisi Penelitian

Penulis	Judul	Objek	Pokok Bahasan
Hui Li, Chunhua Shen (2016)	*Reading Car License Plates Using Deep Convolutional Neural Networks and LSTMs*	Gambar Mobil	Penelitian ini menangani masalah deteksi dan rekognisi plat nomor pada sebuah gambar mobil dengan menggunakan Deep Neural Networks (DNN).
Yonten Jamtsho, et al. (2019)	*Real-time Bhutanese license plate localization using YOLO*	Video Mobil	Penelitian ini membahas kinerja YOLOV2 dalam mendeteksi lokasi plat nomor mobil di Negara Bhutan dari sebuah video
Rayson Laroca, et al. (2018)	*A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector*	Gambar Mobil	Penelitian ini menyajikan Automatic License Plate Recognition (ALPR) yang kuat dan efisien sistem berdasarkan pada detektor objek YOLO yang canggih.
Ravy Hayu Pramestya (2018)	Deteksi dan Klasifikasi Kerusakan Jalan Aspal Menggunakan Metode YOLO Berbasis Citra Digital	Gambar Aspal	Penelitian ini menangani tentang deteksi dan klasifikasi citra kerusakan jalan dengan menggunkan metode YOLO.
Wahyu Ardiansyah (2019)	Peningkatan Performansi Metode Harris Corner untuk Deteksi Plat Nomor Pada Video Menggunakan Maximally Stable External Region	Video Mobil bergerak	Penelitian ini menangani permasalahan deteksi plat nomor kendaraan bergerak dari objek sebuah mobil melalui video dengan menggunakan Maximally Stablle External Region
Ario Fajar Pratama (2020)	Deteksi Plat Nomor Kendaraan Bergerak Berbasis Metode You Only Look Once (YOLO)	Video Mobil dan Motor	Penelitian ini disajikan agar dapat lebih mengetahui hasil deteksi plat nomor pada sebuah kendaran bergerak melalui video dengan menggunakan metode You Only Look Once (YOLO).

2.2 Pengolahan Citra

Pengolahan citra digital (*Digital Image Processing*) adalah sebuah disiplin ilmu yang mempelajari tentang teknik-teknik mengolah citra. Citra yang dimaksud disini adalah gambar diam (foto) maupun gambar bergerak (yang berasal dari *webcam*). Sedangkan digital disini mempunyai maksud bahwa pengolahan citra/gambar dilakukan secara digital menggunakan komputer. Secara matematis, citra merupakan fungsi kontinu (*continue*) dengan intensitas cahaya pada bidang dua dimensi. Agar dapat diolah dengan komputer digital, maka suatu citra harus dipresentasikan secara numerik dengan nilai-nilai diskrit. Repersentasi dari fungsi kontinu menjadi nilai-nilai diskrit disebut digitalisasi citra. Sebuah citra digital dapat diwakili oleh sebuah matriks dua dimensi $f(x, y)$ yang terdiri dari M kolom dan N baris, dimana perpotongan antara kolom dan baris disebut piksel (*pixel* = *picture element*) atau elemen terkecil dari sebuah citra. Objek tertentu dapat dideteksi dengan menggunakan pengolahan citra digital ini. Salah satu metode yang digunakan adalah berdasarkan segmentasi warna. Normalisasi RGB adalah salah satu metode segmentasi warna yang memiliki kelebihan yaitu mudah, proses cepat dan efektif pada objek *traffic sign*, maupun aplikasi untuk *face detection* [9].

2.3 Jenis Citra Digital

Pada aplikasi pengolahan citra digital pada umumnya, citra digital dapat dibagi menjadi 3, *color image*, *black and white image* dan *binary image* [9].

2.3.1 *Color Image* atau RGB (*Red, Green, Blue*)

Pada color image ini masing-masing piksel memiliki warna tertentu, warna tersebut adalah merah (*Red*), hijau (*Green*) dan biru (*Blue*). Jika masing-masing warna memiliki range 0 - 255, maka totalnya adalah $255^3 = 16.581.375$ (16 K) variasi

warna berbeda pada gambar, dimana variasi warna ini cukup untuk gambar apapun. Karena jumlah bit yang diperlukan untuk setiap *pixel*, gambar tersebut juga disebut gambar-bit warna. *Color image* ini terdiri dari tiga matriks yang mewakili nilai -nilai merah, hijau dan biru untuk setiap pikselnya [9].

2.3.2 *Black and White*

Citra digital *black and white* (*grayscale*) setiap pikselnya mempunyai warna gradasi mulai dari putih sampai hitam. Rentang tersebut berarti bahwa setiap piksel dapat diwakili oleh 8 bit, atau 1 *byte*. Rentang warna pada *black and white* sangat cocok digunakan untuk pengolahan *file* gambar. Salah satu bentuk fungsinya digunakan dalam kedokteran ($X-ray$). *Black and white* sebenarnya merupakan hasil rata-rata dari *color image*, dengan demikian maka persamaannya dapat dituliskan sebagai berikut :

$$I_{BW}(x,y) = \frac{I_R(x,y) + I_G(x,y) + I_B(x,y)}{3} \qquad (2.1)$$

dimana $I_R(x,y)$ = nilai piksel *Red* titik (x,y), $I_G(x,y)$ = nilai piksel *Green* titik (x,y), $I_B(x,y)$ = nilai piksel Blue titik (x,y), sedangkan $I_{BW}(x,y)$ = nilai piksel *black and white* titik (x,y) [9].

2.4 Segmentasi Warna Normalisasi RGB

Segmentasi warna, ada bermacam - macam model warna. Model RGB (*Red, Green, Blue*) merupakan model yang banyak digunakan, salah satunya adalah monitor. Pada model ini untuk merepresentasikan gambar menggunakan 3 buah komponen warna tersebut. Selain model RGB terdapat juga model normalisasi RGB dimana model ini terdapat 3 komponen yaitu, r, g, b yang merepresentasikan presentase dari sebuah piksel pada citra digital. Nilai-nilai tersebut

mengikuti persamaan-persamaan 2.2 di bawah ini :

$$r = \frac{R}{R+G+B}, \quad g = \frac{G}{R+G+B}, \quad b = \frac{B}{R+G+B} \quad (2.2)$$

Sehingga : $r + g + b = 1$
Dengan demikian berdasarkan persamaan di atas maka cukup hanya menggunakan r dan g saja, karena nilai b bisa didapatkan dengan menggunakan $b = 1–r–g$ [9].

2.5 Plat Nomor Kendaraan

Tanda Nomor Kendaraan Bermotor (TNKB). Dalam Perkapolri nomor 5 tahun 2012, menyebutkan bahwa TNKB dibuat dari bahan yang mempunyai unsur-unsur pengaman sesuai spesifikasi teknis. Unsur-unsur pengaman TNKB yaitu berupa logo lantas dan pengaman lain yang berfungsi sebagai penjamin legalitas TNKB. Selain itu, dalam Perkapolri nomor 5 tahun 2012 juga disebutkan mengenai warna TNKB. Pengaturan mengenai TNKB, dapat dilihat ketentuannya dalam Undang-Undang No. 22 Tahun 2009 tentang Lalu Lintas dan Angkutan Jalan (UU LLAJ) beserta peraturan pelaksananya. TNKB adalah tanda registrasi dan identifikasi kendaraan bermotor yang berfungsi sebagai bukti legitimasi pengoperasian kendaraan bermotor berupa pelat atau berbahan lain dengan spefikasi tertentu yang diterbitkan Polri dan berisikan kode wilayah, nomor registrasi, serta masa berlaku dan dipasang pada kendaraan bermotor. Jika melihat pada PP Kendaraan, juga tidak ada ketentuan yang mengatur spesifikasi TNKB. Yang diatur dalam PP Kendaraan antara lain hanya :

1. Lampu penerangan tanda nomor Kendaraan Bermotor di bagian belakang Kendaraan berwarna putih.

2. Lampu penerangan tanda nomor Kendaraan Bermotor dipasang di bagian belakang dan dapat menyinari tanda

nomor Kendaraan Bermotor agar dapat dibaca pada jarak paling sedikit 50 (lima puluh) meter dari belakang.

3. Tempat pemasangan tanda nomor Kendaraan Bermotor harus memenuhi persyaratan :

   (a) Ditempatkan pada sisi bagian depan dan belakang Kendaraan Bermotor; dan

   (b) Dilengkapi lampu tanda nomor Kendaraan Bermotor pada sisi bagian belakang Kendaraan Bermotor.

Selain itu, menurut Perkapolri nomor 5 tahun 2012 juga disebutkan mengenai spesifikasi teknis TNKB, yaitu sebagai berikut :

1. Berbentuk plat aluminium dengan cetakan tulisan dua baris. Baris pertama menunjukkan : kode wilayah (huruf), nomor polisi (angka), dan kode/seri akhir wilayah (huruf). Baris kedua menunjukkan bulan dan tahun masa berlaku, masing-masing dua digit (misalnya 01.20 berarti berlaku hingga Januari 2020).

2. Bahan baku TNKB adalah aluminium dengan ketebalan 1 mm. Ukuran TNKB untuk kendaraan bermotor roda 2 dan roda 3 adalah 250—105 mm, sedangkan untuk kendaraan bermotor roda 4 atau lebih adalah 395—135 mm.

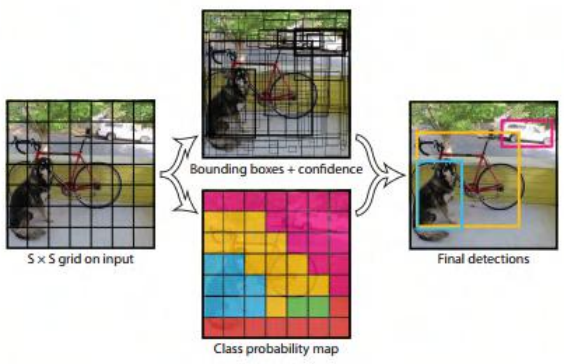
3. Terdapat garis putih di sekitar TNKB dan tidak ada batas pemisah antara nomor polisi dan masa berlaku (dari tahun 2011).

4. Pada pertengahan 2014 terjadi perubahan tampilan. Plat nomor kini sedikit diperpanjang dari ukuran semula (untuk roda empat). Selain itu, terdapat perubahan

posisi lambang Polantas dan tulisan "Korlantas Polri", yakni, lambang Polantas kini berada di sudut kiri atas dan kanan bawah, sedangkan tulisan "Korlantas Polri" berada pada sudut kiri bawah dan kanan atas [10].

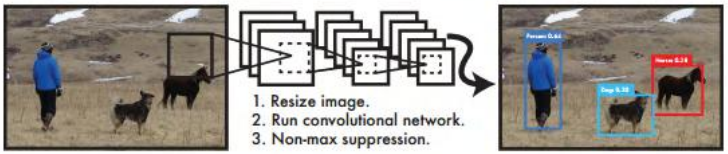
2.6 *You Only Look Once* (YOLO)

YOLO merupakan *Real Object Deteciton* 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 pendekatan 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 [3].



Gambar 2.1: Model Sistem YOLO [4].

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



Gambar 2.2: Sistem Deteksi YOLO [4].

2.7 Kelebihan dan Kekurangan YOLO

Dalam beberapa hal YOLO memiliki kelebihan diantaranya Pertama YOLO sangat cepat, karena YOLO menjadikan pendeteksian objek sebagai masalah regresi tunggal, yang memroses langsung dari piksel gambar hingga koordinat kotak pembatas dan probabilitas kelas sehingga tidak memerlukan alur yang kompleks. Dengan menggunakan YOLO, sistem hanya melihat sekali (*You Only Look Once*) pada gambar untuk memprediksi benda apa yang ada dan dimana tempatnya.

Kedua, YOLO mempertimbangkan secara global tentang citra saat membuat prediksi.Tidak seperti *sliding window* dan teknik berbasis *region proposal*, YOLO melihat keseluruhan gambar selama masa pelatihan dan pengujian secara implisit mengkodekan informasi kontekstual tentang kelas sesuai objek yang ditampilkan pada citra. *Fast R-CNN*, metode deteksi paling bagus, mempunyai kesalahan mendeteksi

latar belakang citra untuk objek karena tidak dapat melihat konteks yang lebih besar. YOLO membuat kurang dari setengah jumlah kesalahan latar belakang dibandingkan dengan *Fast R-CNN*.

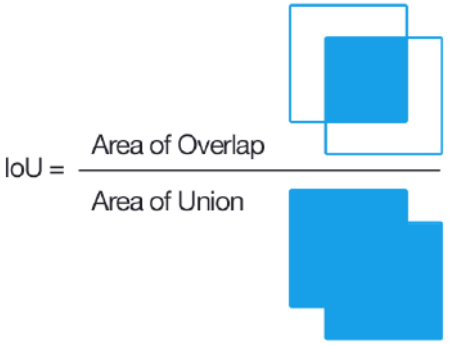
Ketiga, YOLO dapat mempelajari menggeneralisasi representasi objek. Saat dilatih tentang gambar alami dan diuji gambar seni, YOLO mengungguli metode metode pendeteksian terbaik seperti DPM dan R-CNN dengan selisih lebar. Karena YOLO sangat digeneralisasikan, kemungkinan besar akan rusak saat diterapkan pada domain baru atau masukan tak terduga [4].

Karena YOLO berbasis *deep learning* sehingga kekurangan dari YOLO adalah pengimplementasiannya yang kurang cepat jika dilakukan pada perangkat keras CPU. Sehingga perlu digunakan perangkat keras GPU dalam menjalankan YOLO ini. Adapun YOLO terbagi menjadi beberapa versi diantaranya YOLOV1, YOLOV2, dan YOLOV3. Pada YOLOV1 diperkenalkan, dengan menjadikan pendeteksian objek sebagai masalah regresi tunggal, YOLOV1 berhasil membuktikan bahwa YOLOV1 dapat menjadi metode pendeteksian objek yang efisien dibandingkan dengan metode *machine learning* lainnya. Namun, kekurangan dari YOLOV1 adalah tidak dapat melakukan pendeteksian objek yang kecil. Kemudian muncul YOLOV2 dengan memperbaiki kekurangan dari YOLOV1, YOLOV2 lebih cepat dibandingkan dengan YOLOV1 dan dapat melakukan pendeteksian pada objek yang kecil. Namun, karena YOLOV2 fokus pada kecepatan jaringannya sehingga akurasi yang dihasilkan tidak berbanding lurus dengan kecepatannya. Kemudian hadir YOLOV3 yang memfokuskan pada akurasi pendeteksian objek, berbeda dengan YOLOV2, YOLOV3 dapat menghasilkan akurasi yang lebih tinggi daripada YOLOV2 tapi, kecepetan

jaringannya lebih rendah daripada YOLOV2.

2.8 *Intersection over Union* (IoU)

Ketika mendeteksi letak objek, metode 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 pertemuan antara kotak prediksi objek dan kotak kebenaran dasar (*ground truth*) dan membaginya dengan area persatuan mereka. Perumusan IoU tersebut dapat diilustarasikan pada gambar 2.3.

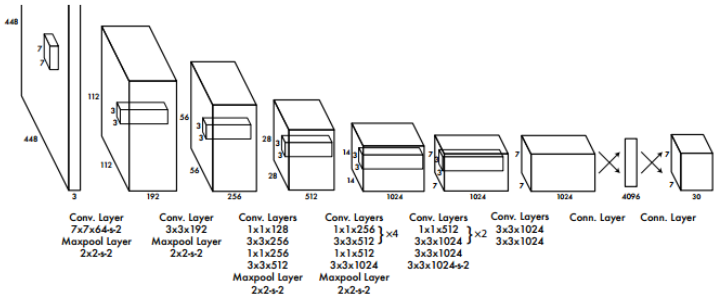


Gambar 2.3: Ilustrasi Perhitungan IoU

Saat mengevaluasi algoritma pendeteksian objek, ambang IoU sebesar 0,5 biasanya digunakan untuk menentukan apakah deteksi benar [11]. Namun, nilai IoU = 0,5 mempunyai area yang cukup longgar. Sehingga diinginkan nilai IoU yang lebih besar dari 0,5 [12].

2.9 Desain Jaringan

YOLO menyatukan komponen yang terpisah dari pendeteksian objek menjadi satu jaringan syaraf. YOLO memanfaatkan fitur dari keseluruhan gambar untuk memprediksi setiap kotak pembatas. YOLO memprediksi semua kotak pembatas pada semua kelas objek untuk sebuah gambar secara bersamaan. Ini berarti YOLO mempertimbangkan seluruh bagian citra secara global dan semua objek pada citra. YOLO membagi gambar masukan menjadi $S \times S$ petak (*grid*). Jika pusat objek ada di dalam suatu petak, sel petak tersebut bertanggung jawab untuk mendeteksi objek itu. Setiap sel petak memprediksi B kotak pembatas dan nilai keyakinan untuk kotak – kotak tersebut. Nilai keyakinan ini mencerminkan seberapa yakin kotak itu berisi objek dan juga seberapa akuratnya kotak yang diprediksi itu. Secara formal YOLO mendefinisikan kepercayaan sebagai $Pr(Object) * IoU^{truth}_{pred}$



Gambar 2.4: Arsitektur dasar YOLO [4]

Jika tidak ada objek di sel itu, nilai keyakinan harus nol. Jika tidak, nilai keyakinan akan sama dengan *intersection*

over union (IoU) antara kotak yang diprediksi dan kotak kebenaran latar belakang (*ground truth*). Setiap kotak pembatas terdiri dari 5 prediksi : x, y, w, h dan nilai keyakinan p. Koordinat (x, y) mewakili pusat kotak relatif terhadap batas sel petak. Lebar (w) dan tinggi (h) diprediksi relatif terhadap keseluruhan gambar. Akhirnya prediksi nilai keyakinan menyatakan IoU antara kotak yang diprediksi dan kotak *ground truth*. Setiap sel petak juga diprediksi probabilitas kelas kondisional C, $Pr(class_i|Object)$. Probabilitas ini dikondisikan pada sel petak yang berisi objek. YOLO memprediksi satu set probabilitas kelas per sel petak. Berapapun jumlah kotak B. pada saat uji coba kita mengalikan probabilitas kelas kondisional dan prediksi keyakinan kotak individu yang memberi nilai keyakinan khusus kelas untuk setiap kotak, dan ditunjukkan pada persamaan 2.3 di bawah ini.

$$\Pr(\text{class}_i|Object) * Pr(Object) * IoU_{pred}^{truth} = Pr(class_i) * IoU_{pred}^{truth} \quad (2.3)$$

Nilai ini menyandikan probabilitas kelas yang muncul di kotak dan seberapa baik kotak yang diprediksi sesuai dengan objek. Ilustrasi model dapat dilihat pada gambar 2.4. YOLO menerapkan model ini sebagai CNN. *Convolutiunal Layer* awal dari jaringan mengekstrak fitur dari citra, sementara *fully-connected* layer memprediksi probabilitas dan koordinat keluaran.

2.10 *Loss function of YOLO Method*

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. Berbagai macam fungsi kerugian diantaranya [13] :

1. *Loss Function Localization*
   Kerugian lokalisasi diukur dari kesalahan dalam prediksi lokasi dan ukuran kotak pembatas. Disini hanya dihitung kotak yang responsibel untuk mendeteksi objek, dapat dilihat pada persamaan 2.4.

$$\begin{aligned} &\textit{Loss Localization} = \\ &\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[(x_i - \hat{x_i})^2 + (y_i - \hat{y_i})^2 \right] \\ &+\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w_i}} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h_i}} \right)^2 \right] \end{aligned} \tag{2.4}$$

   Keterangan :
   $\lambda_{coord} = 5.0$
   (x_i, y_i) = Titik tengah kotak prediksi
   $(\hat{x_i}, \hat{y_i})$ = Titik tengah *ground truth*
   (w_i, h_i) = Ukuran kotak prediksi
   $(\hat{w_i}, \hat{h_i})$ = Ukuran *ground truth*
   1_{ij}^{obj} = Bernilai 1 jika sel i dan kotak j ada objek, dan 0 jika tidak ada objek
   s^2 = jumlah keseluruhan sel pada setiap fitur map
   B = Jumlah *bounding box* pada tiap sel

2. *Loss Function Confidence*
   Jika objek terdeksi dalam kotak pembatas, maka *Confidence loss function* ditunjukkan pada persamaan 2.5 :

$$\textit{Loss Confidence} = \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left(C_i - \hat{C}_i\right)^2 + \lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{noobj} \left(C_i - \hat{C}_i\right)^2 \quad (2.5)$$

Keterangan :
$\lambda_{noobj} = 0.5$
C_i = Nilai kelas dari kotak prediksi
$\hat{C}_i$ = Nilai kelas dari *ground truth*
1_{ij}^{noobj} = Bernilai 1 jika sel i dan kotak j tidak ada objek, dan 0 jika ada objek
s^2 = jumlah keseluruhan sel pada setiap fitur map
B = Jumlah *bounding box* pada tiap sel

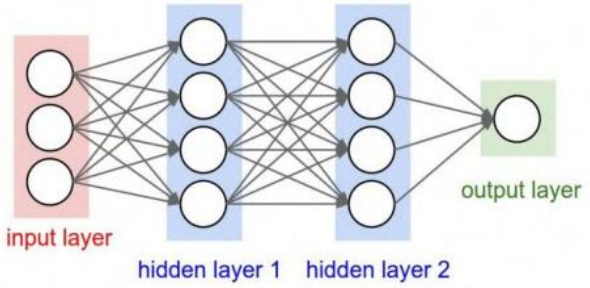
3. *Loss Function Classification*
   Jika suatu objek terdeteksi, kerugian klasifikasi di setiap sel adalah kesalahan kuadrat dari probabilitas bersyarat kelas untuk setiap kelas dan dinyatakan sebagai persamaan 2.6 berikut :

$$\textit{Loss Classification} = \sum_{i=0}^{s^2} 1_i^{obj} \sum_{c \in classes} \left(p_i(c) - \hat{p}_i(c)\right)^2 \quad (2.6)$$

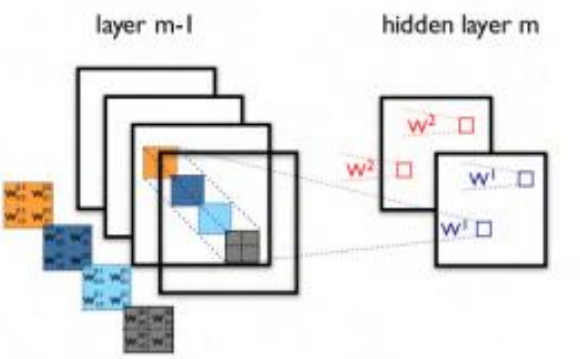
Keterangan :
$p_i(c)$ = Probabilitas kelas objek *bounding box*
$\hat{p}_i(c)$ = IoU
s^2 = jumlah keseluruhan sel pada setiap fitur map
1_i^{obj} = Bernilai 1 jika sel i ada objek, dan 0 jika tidak ada objek

2.11 *Convolutional Neural Network* (CNN)

CNN pertama kali dikembangkan dengan nama *NeoCognitron* oleh Kunihiko Fukushima, seorang peneliti dari NHK *Broadcasting Science Research Laboratories*, Kinuta, Setagaya, Tokyo, Jepang. *Convolutional Neural Network* (CNN) adalah pengembangan dari *Multilayer Perceptron* (MLP) yang didesain untuk mengolah data dua dimensi. CNN termasuk dalam jenis *Deep Neural Network* karena kedalaman jaringan yang tinggi dan banyak diaplikasikan pada data citra. Pada kasus klasifikasi citra, MLP kurang sesuai untuk digunakan karena tidak menyimpan informasi spasial dari data citra dan menganggap setiap piksel adalah fitur yang independen sehingga menghasilkan hasil yang kurang baik. Cara kerja CNN memiliki kesamaan pada MLP, namun dalam CNN setiap neuron dipresentasikan dalam bentuk dua dimensi, tidak seperti MLP yang setiap neuron hanya berukuran satu dimensi.



Gambar 2.5: Arsitektur MLP Sederhana



Gambar 2.6: Proses Konvolusi pada CNN

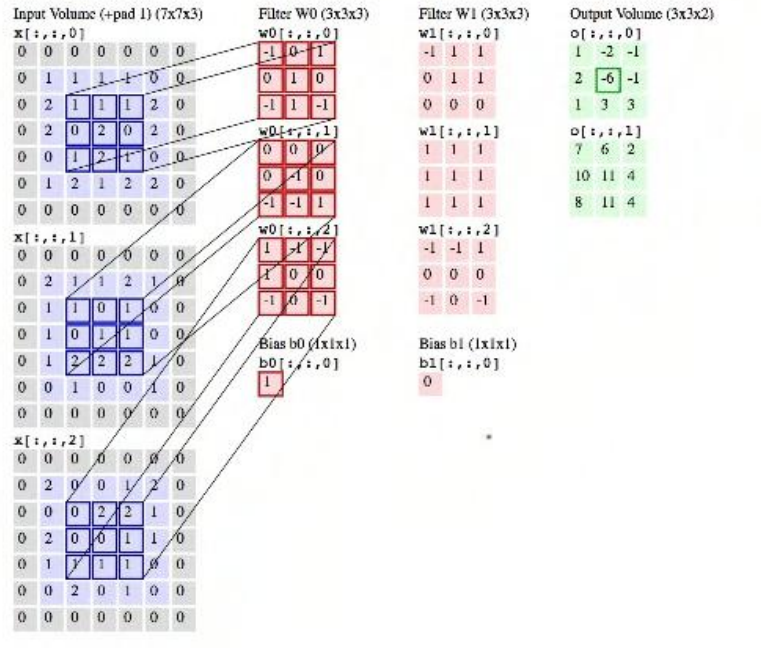
Sebuah MLP seperti pada Gambar 2.5. memiliki i *layer* (kotak merah dan biru) dengan masing-masing layer berisi J_i neuron (lingkaran putih). MLP menerima input data satu dimensi dan mempropagasikan data tersebut pada jaringan hingga menghasilkan output. Setiap hubungan antar neuron pada dua layer yang bersebelahan memiliki parameter bobot satu dimensi yang menentukan kualitas mode. Disetiap data input pada layer dilakukan operasi linear dengan nilai bobot yang ada, kemudian hasil komputasi akan ditransformasi menggunakan operasi non linear yang disebut sebagai fungsi aktivasi.Pada CNN, data yang dipropagasikan pada jaringan adalah data dua dimensi, sehingga operasi linear dan parameter bobot pada CNN berbeda. Pada CNN operasi linear menggunakan operasi konvolusi, sedangkan bobot tidak lagi satu dimensi saja, namun berbentuk empat dimensi yang merupakan kumpulan kernel konvolusi seperti pada Gambar 2.6 Dimensi bobot pada CNN adalah :

Neuron Input $\times$ Neuron Output $\times$ Tinggi $\times$ Lebar

Karena sifat proses konvolusi, maka CNN hanya dapat digunakan pada data yang memiliki struktur dua dimensi seperti citra dan suara [14].

2.12 *Convolutional Layer*

Convolution Layer melakukan operasi konvolusi pada output dari *layer* sebelumnya. *Layer* tersebut adalah proses utama yang mendasari sebuah CNN. Konvolusi adalah suatu istilah matematis yang berati mengaplikasikan sebuah fungsi pada output fungsi lain secara berulang. Dalam pengolahan citra, konvolusi berati mengaplikasikan sebuah *kernel* pada citra disemua *offset* yang memungkinkan seperti yang ditunjukkan pada Gambar.2.7. *Kernel* bergerak dari sudut kiri atas ke kanan bawah. Sehingga hasil konvolusi dari citra tersebut dapat dilihat pada gambar disebelah kanannya. Tujuan dilakukannya konvolusi pada data citra adalah untuk mengekstraksi fitur dari citra input. Konvolusi akan menghasilkan transformasi linear dari data input sesuai informasi spasial pada data. Bobot pada layer tersebut menspesifikasikan *kernel* konvolusi yang digunakan, sehingga *kernel* konvolusi dapat dilatih berdasarkan input pada CNN.



Gambar 2.7: Proses Konvolusi

2.13 *Confusion Matrix*

Confusion matrix adalah sebuah tabel yang mendeskripsikan bagaimana performa suatu model berdasarkan test data yang diberikan. Secara spesifik di dalam permasalahan *object detection*, *confusion matrix* berisikan beberapa kondisi yaitu : [11] :

1. *True Positive* (TP), kondisi dimana *bounding box* memiliki nilai IoU lebih dari *threshold* (IoU $\geq$ *threshold*)

2. *False Positive* (FP), kondisi dimana bounding box memiliki nilai IoU lebih rendah dari *threshod* (IoU $<$

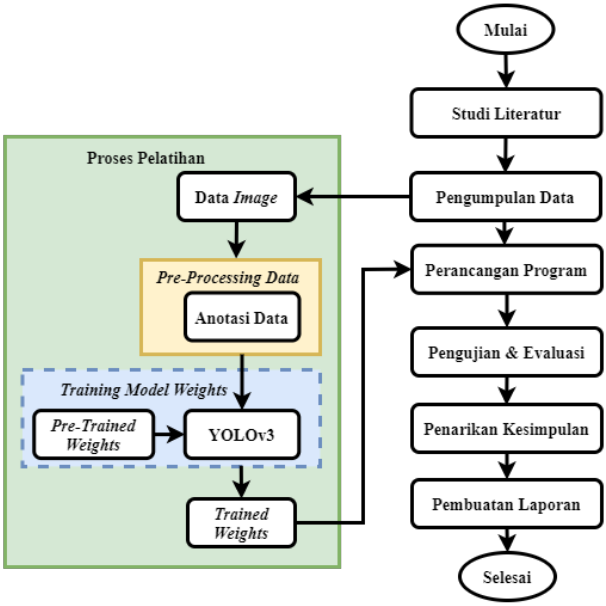
threshold) atau kondisi dimana tidak terdapatnya objek pada bounding box.

3. *False Negative* (FN), kondisi dimana objek tidak terdeteksi oleh bounding box.

Dari penjelasan diatas, dapat diketahui bahwa tidak terdapat kondisi *True Negative* (TN) dikarenakan TN digambarkan sebagai kondisi dimana tidak adanya objek terdeteksi dengan benar sebagai deteksi tanpa objek. Jika diasumsikan terdapat kondisi TN, maka sanga memungkinkan terdapat banyak deteksi tanpa objek dalam setiap data. Oleh karena itu kondisi TN dapat diabaikan, TN = 0.

BAB III
METODE PENELITIAN

Pada bab ini akan dijelaskan mengenai metodologi sistem penelitian yang akan 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: Blok Diagram Pengerjaan Tugas Akhir

3.1 Studi Literatur

Tahap studi literatur yaitu tahap melakukan pendalaman materi demi menunjang metode YOLO. Materi tersebut berupa tentang pengolahan citra digital menggunakan deep learning, cara kerja struktur CNN, dan materi – materi lain yang medukung penelitian tugas akhir ini. Materi tersebut didapatkan dari referensi yang sudah dikumpulkan dari berbagai literatur diantaranya berupa jurnal ilmiah, diskusi, buku, artikel, dan website yang membahas tentang penelitian ini.

3.2 Pengumpulan Data

Dalam melakukan penelitian, pengumpulan data merupakan faktor terpenting dalam keberhasilan penelitian. Jenis data yang diperoleh dari penelitian ini merupakan jenis data primer dan data sekunder. Data primer yang digunakan pada penelitian ini merupakan data citra video kendaraan bergerak yang terdiri dari kendaraan roda dua dan roda empat. Pengambilan data dilakukan saat kondisi langit cerah dan lensa kamera yang dipasang dapat menjangkau lebar jalan dalam jalur satu arah. Sedangkan data sekunder pada penelitian ini berupa data citra *image* kendaraan roda dua dan roda empat.

3.3 Proses Pelatihan

Deep learning adalah salah satu metode pembelajaran mesin yang memungkinkan komputer untuk mempelajari tugas-tugas yang disesuaikan dengan sifat manusia. Oleh karena itu, dalam proses pembelajarannya komputer diperlukan proses pelatihan agar dapat mengenali tugas – tugas yang diberikan. Adapun tahapan – tahapn proses pelatihan dalam penelitian ini sebagai berikut :

1. Data *Image*
   Tahap data *image* diperoleh dari kumpulan data

video yang diubah menjadi data *image* dengan cara akuisisi video dan data yang diperoleh dari *Google Open Images Dataset V6* yang dapat diunduh di *storage.googleapis.com/openimages/web/index.html*. Data *image* diperlukan karena pada proses pelatihan, data yang dibutuhan hanya berupa data *image*.

2. *Pre-Processing Data*
   Tahap *pre-processing* data dilakukan sebelum proses *training model weights*. Adapun tahap *pre-processing* data pada metode ini yaitu anotasi data. Tahap anotasi data adalah tahap memberikan keterangan pada setiap objek data citra. Keterangan tersebut berupa koordinat kotak pembatas, kelas objek, dan ukuran kotak pembatas pada objek data citra. Keterangan ini nantinya digunakan untuk proses pelatihan.

3. *Training Model Weights*
   Dalam proses *training model weights* menggunakan model dari *darknet open source*, diperlukan bobot awal dari sumber yang sama. Dalam pelatihan ini, bobot awal yang digunakan bukan data *random* tetapi, data *pre-trained weights* yang telah dilatih sebelumnya pada jaringan YOLO pada dataset COCO. Hal ini pada dasarnya menggunakan prinsip *transfer learning*. *Transfer learning* adalah suatu teknik atau metode yang memanfaatkan model yang sudah dilatih terhadap suatu dataset untuk menyelesaikan permasalahan lain yang serupa dengan cara menggunakannya sebagai *starting point*, memodifikasi dan mengupdate parameternya sehingga sesuai dengan dataset yang baru. Oleh karena itu, *Pre-trained weights* membuat proses *training model weights* berjalan lebih cepat karena menggunakan prinsip *transfer learning* pada dataset COCO dan

memanfaatkan model *darknet open source* yang sudah ada, sehingga *starting point* pengerjaan tidak mulai dari 0. Hasil dari proses ini adalah sebuah bobot baru dari hasil data *training* sebelumnya. *Trained weights* ini akan digunakan pada proses pengujian program atau *testing* dalam pendeteksian objek citra.

4. IoU Akurasi
   Pada tahap ini melakukan penghitungan kinerja metode dengan mencari nilai IoU prediksi kotak pembatas yang dijelaskan pada bagian 2.8. Akurasi menggambarkan seberapa besar model memberikan hasil klasifikasi secara benar dibanding dengan keseluruhan data. Akurasi dapat dihitung melalui persamaan 3.1 seperti berikut :

$$\text{Akurasi} = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{3.1}$$

   Presisi merupakan perbandingan jumlah data yang diklasifikasikan secara benar dan bernilai *true* dengan keseluruhan data yang diklasifikasikan *true* oleh model. Presisi dapat dihitung melalui persamaan 3.2 seperti berikut:

$$\text{Presisi} = \frac{TP}{TP + FP} \tag{3.2}$$

   Recall merupakan perbandingan jumlah data yang diklasifikasikan secara benar dan bernilai *true* dengan keseluruhan data yang diklasifikasikan *true*. *Recall* dapat dihitung melalui persamaan 3.3 seperti berikut:

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

   Keterangan :

(a) *True Positive* (TP), kondisi dimana model mengklasifikasikan suatu data sebagai *true* dan kelas sebenarnya dari data tersebut adalah *true*

(b) *True Negative* (TN), kondisi dimana model mengklasifikasikan suatu data sebagai *false* dan kelas sebenarnya dari data tersebut adalah *false*

(c) *False Positive* (FP), kondisi dimana model mengklasifikasikan suatu data sebagai *true* dan kelas sebenarnya dari data tersebut adalah *false*

(d) *False Negative* (FN), kondisi dimana model mengklasifikasikan suatu data sebagai *false* dan kelas sebenarnya dari data tersebut adalah *true*[15]

3.4 Perancangan Program

Pada tahap ini dilakukan perancangan program. Program ini menggunakan Bahasa Java. Adapun tahapan – tahapan yang akan dilakukan dalam perancangan program ini dapat dilihat pada Gambar 3.3, diantaranya :

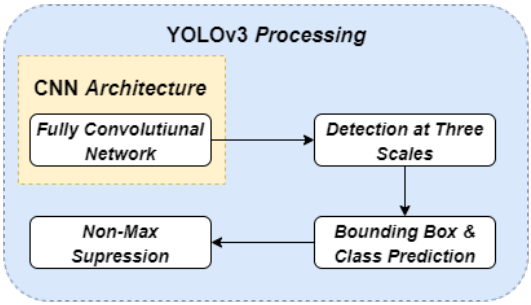
1. Input dan akuisisi video
   Dalam tahap ini video yang telah diambil pada pengumpulan data di awal dimasukkan atau diinputkan dalam program. Sedangkan pada proses akuisisi nanti, video yang telah diinputkan akan diubah menjadi *frame – frame* gambar.

2. *Load Trained Weights*
   Pada proses ini, hasil yang didapatkan *trained weights* dari proses pelatihan akan di *import* pada program dan akan digunakan dalam pendeteksian plat nomor kendaraan.

3. Atur *Region of Interest* (ROI)
   Region of Interest (ROI) adalah bagian dari video yang

akan menjadi pusat dari pengamatan. Hal ini diperlukan agar pengolahan gambar lebih efisien dan cepat karena tidak semua piksel dalam gambar diolah.

4. YOLOV3 *Processing*
   Pada tahap ini, deteksi plat nomor kendaraan dilakukan dengan menggunakan proses YOLOv3. Hasil pada proses ini berupa *bounding box* dan *class prediction* untuk objek plat nomor kendaraan.



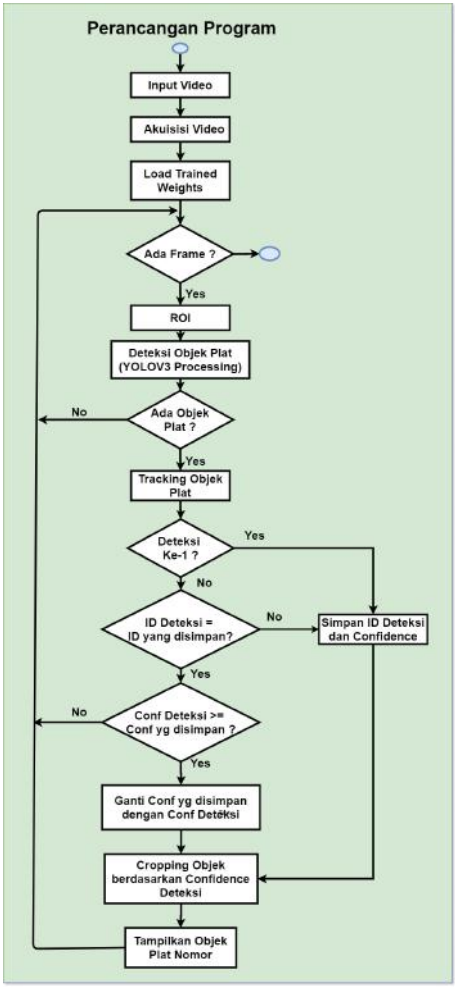
Gambar 3.2: YOLOV3 Processing

5. Tracking Objek Plat Nomor
   Pada tahap ini, setelah berhasil mendeteksi objek plat nomor, objek akan diberi nomor ID sehingga memudahkan dalam menghitung jumlah objek dan membedakan antar tiap objek yang terdeteksi

6. Memilih *Confidence* Tertinggi
   Pada tahap ini digunakan dalam mengacu lokalisasi plat nomor dari video yang masuk di area ROI.

7. *Cropping License Plate Detections*
   Pada tahap ini, setelah objek citra sudah terdeteksi hasil

tersebut akan dilakukan *cropping image* sesuai dengan ukuran kotak pembatas atau *bounding box* yang ada.



Gambar 3.3: Tahapan Perancangan Program

BAB IV
PERANCANGAN DAN IMPLEMENTASI SISTEM

Pada bab ini akan dijelaskan rancangan sistem pendeteksian plat nomor kendaraan bergerak menggunakan metode YOLOV3. Hasil dari analisis perancangan sistem ini kemudian dilanjutkan dengan implementasi dari rancangan tersebut pada sebuah perangkat lunak untuk mengetahui kinerja dari metode yang digunakan.

4.1 Analisis Sistem

Untuk merancang sistem yang dapat mendeteksi plat nomor kendaraan bergerak berbasis video digital, maka diperlukan sistem yang mengolah data masukan berupa rekaman video arus kendaraan yang melintasi jalan raya. Pada tugas akhir ini data input yang digunakan berupa video kendaraan yang melintasi jalan raya satu arah. Dengan menggunakan pengolahan citra digital dan pengolahan video pada data input tersebut, maka akan diperoleh sebuah sistem yang dapat mendeteksi plat nomor kendaraan bergerak melintasi jalan raya melalui video sesuai dengan tujuan tugas akhir ini.

4.1.1 Analisis Fungsional Sistem

Perangkat lunak yang dapat dirancang untuk mendeteksi plat nomor kendaraan bergerak memiliki kemampuan sebagai berikut :

1. Melakukan ekstraksi dari video menjadi *frame-frame* gambar.

2. Mendeteksi plat kendaraan bergerak yang melintas pada ROI (*Region of Interest*).

3. Menunjukkan hasil deteksi lokasi plat nomor kendaraan bergerak.

4. Menunjukkan waktu yang dibutuhkan plat nomor kendaraan yang terdeteksi pada area ROI

4.1.2 Analisis non-Fungsional Sistem

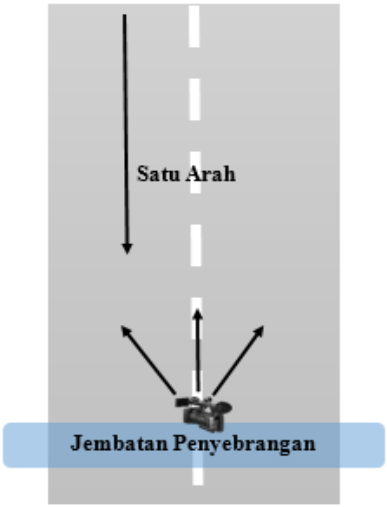
Proses yang dibangun pada penelitian ini merupakan proses deteksi dari plat nomor kendaraan bergerak berbasis video digital. Dalam video digital ini terdapat dua jenis plat kendaraan yang akan dideteksi yaitu plat kendaraan bermotor dan mobil. Proses deteksi ini mampu mengenali objek plat kendaraan bergerak dalam suatu citra. Selanjutnya hasil pengenalan objek tersebut akan ditunjukkan pada sebuah kotak pembatas dan dilakukan *cropping image* sesuai dengan ukuran kotak pembatas tersebut.

4.2 Perancangan Data

Proses pendeteksian objek berbasis citra digital dengan menggunakan *deep learning* memerlukan sejumlah besar data untuk dilatih, sehingga, mendapatkan bobot optimal dalam pendeteksian sebuah objek. Selain itu, diperlukan pula sejumlah data untuk proses pengujian untuk menguji keakuratan model dari metode yang digunakan. Dari data yang sudah terkumpul, diambil beberapa video untuk diekstraksi menjadi *frame - frame* gambar, hal ini diperlukan karena proses pelatihan hanya dapat mengolah data yang berupa data *image*. Data *image* pada proses pelatihan ini juga diperoleh dari *Google Open Images Dataset V6* yang dapat diunduh di *storage.googleapis.com/openimages/web/index.html*

4.2.1 Data Inputan

Data inputan dalam sistem ini berupa data *image* dan video digital. Data *image* digunakan pada proses pelatihan sedangkan data video digital digunakan pada proses perancangan dan pengujian program. Adapun layout dalam pengambilan video rekaman kendaraan bergerak dapat dilihat pada Gambar 4.1



Gambar 4.1: *Layout* Perekaman Video

4.2.2 Data Proses

Data proses merupakan data yang digunakan dalam proses pengolahan data input. Data proses ini diperoleh dari hasil pengolahan data input sesuai dengan tahapan algoritma dan metode yang telah disusun. Tahapan dari data proses dijelaskan pada Tabel 4.1.

Tabel 4.1: Data Proses

No	Tahapan	Input	Output
1	*Anotasi data*	Data *image*	Data label kotak pembatas yang disimpan dalam format *.txt*
2	*Training Model Weights*	*Pre-trained Weight* YOLOv3 dan data label kotak pembatas	*Trained weights*
3	*Load Trained Weights*	*Trained Weights*	*Array* yang berisi bobot jaringan YoloV3
4	Input dan Akuisisi Video	Video	*Frame image*
5	ROI (*Region of Interest*)	*frame image* dan ukuran ROI	*Frame image* dengan ROI
6	YoloV3 *Processing*	*Frame image* dan *Array Net Darknet*	Citra deteksi plat nomor kendaraan
7	Tracking Objek Plat	Plat nomor yang terdeteksi	Plat nomor dengan ID
8	Menentukan *Confidence* tertinggi	Kumpulan plat nomor dengan ID yang sama	Lokasi plat nomor dengan *confidence* tertinggi
9	*Cropping License Plate Detections*	Citra hasil lokalisasi plat nomor kendaraan	Citra plat nomor kendaraan

4.2.3 Data Output

Data output merupakan hasil pendeteksian plat nomor kendaraan bergerak. Program dalam tugas akhir ini menghasilkan hasil plat nomor yang terdeteksi pada setiap

kendaraan yang melintas area ROI (*Region of Interest*).

4.3 Perancangan Sistem

Secara umum proses pendeteksian plat nomor kendaraan bergerak menggunakan YOLOV3 dapat ditunjukkan dalam diagram alir yang telah disajikan pada Gambar 3.3. Rincian tentang perancangan sistem pendeteksian plat nomor kendaraan bergerak menggunakan YOLOV3 dijelaskan pada subbab ini.

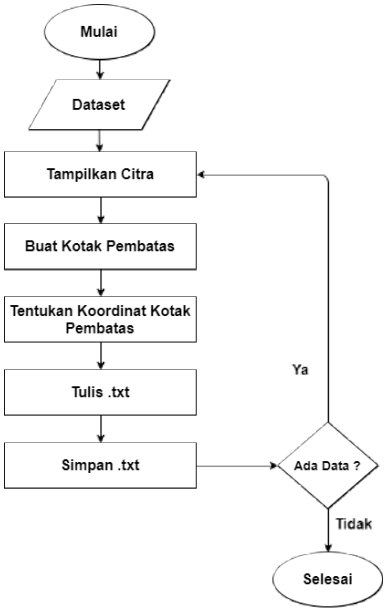
4.3.1 Persiapan Data

Hal yang perlu dilakukan sebelum membangun proses pendeteksian plat nomor kendaraan bergerak adalah dengan mempersiapkan data citra *image* dan citra video yang dibutuhkan. Pada data *image* dilakukan proses pelatihan dan data citra video dilakukan proses perancangan dan pengujian program.

Selanjutnya, diperlukan data anotasi seluruh citra untuk proses pelatihan. Diagram alur proses anotasi data disajikan pada Gambar 4.2. Proses anotasi data dimulai dengan menggambar kotak pembatas secara manual di setiap objek pada citra 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. Adapun tujuan yang menyebabkan diperlukan proses anotasi data ini adalah :

- Hasil dari anotasi data ini akan menjadi bahan untuk spesifikasi dalam menentukan *anchor boxes* dengan menggunakan *K-Means Clustering*.
- Sekumpulan anotasi data yang sudah terkumpul digunakan sebagai mesin pembelajaran dalam membuat *trained weights* pada proses pelatihan.

- Hasil kotak pembatas dari anotasi data ini berperan sebagai kotak kebenaran (*ground truth*) dalam menghitung nilai IoU.



Gambar 4.2: Diagram Alir Anotasi Data

Pada gambar 4.2 merupakan diagram alir dari proses pembuatan anotasi data atau *labelling image*, adapaun penjelasan tiap proses tersebut sebagai berikut :

1. Dataset
   Pada proses ini diperlukan sejumlah data *image* yang akan dianotasikan. Pada prinsipnya karena anotasi data digunakan pada proses pelatihan, maka dari itu semakin banyak dataset yang digunakan semkain baik pula model *trained weights* yang terbentuk.

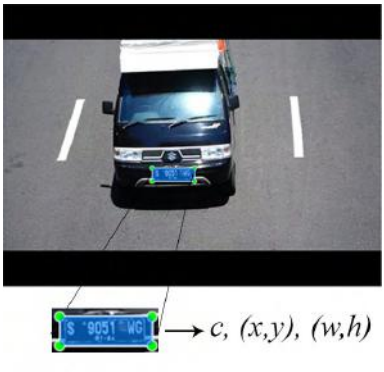
Gambar 4.3: Dataset *Image*

2. Tampilkan citra
   pada proses ini dataset *image* yang terkumpul ditampilkan satu per satu

Gambar 4.4: Proses Tampilkan Citra dari Dataset

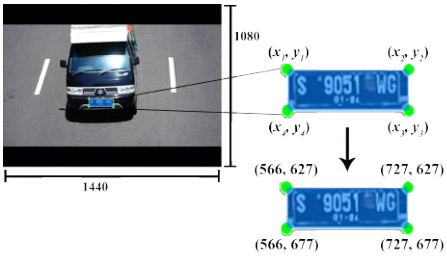
3. Buat Kotak Pembatas
   Pada proses ini dilakukan dengan menggambar kotak pembatas secara manual di setiap objek pada citra. Adapun keterangan dari kotak pembatas ini adalah nilai $c, (x, y), (w, h)$ berturut-turut adalah kelas objek,

koordinat titik pusat kotak pembatas, dan ukuran dimensi dari kotak pembatas.



Gambar 4.5: *Labelling* Kotak Pembatas Secara Manual

4. Tentukan Koordinat Kotak Pembatas
   Pada proses ini bertujuan untuk mendapatkan titik-titik koordinat dari kotak pembatas yang kemudian diolah menjadi keterangan kotak pembatas sesuai parameter yang dibutuhkan yaitu $c, (x, y), (w, h)$.



Gambar 4.6: Citra dengan Koordinat Kotak Pembatas

Pada gambar 4.6 merupakan citra koordinat ujung-

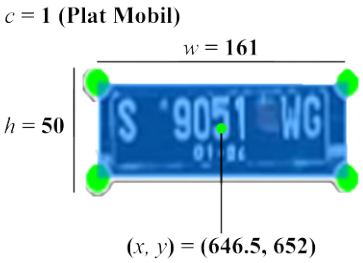
ujung kotak pembatas, dari koordinat tersebut akan dicari koordinat titik pusat dan dimensi kotak pembatas.

- Koordinat ujung kiri atas = $(566, 627)$
- Koordinat ujung kiri bawah = $(566, 677)$
- Koordinat ujung kanan atas = $(727, 627)$
- Koordinat ujung kanan bawah = $(727, 677)$

Sehingga dari data di atas didapatkan :

- Koordinat Titik Pusat Kotak Pembatas

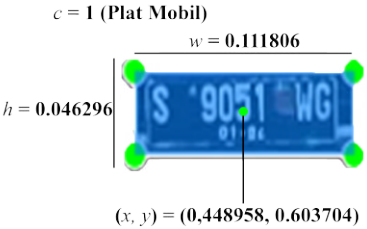
  $(x, y) = (566 + \frac{727 - 566}{2}, 627 + \frac{677 - 627}{2})$
  $(x, y) = (646.5, 652)$
- Dimensi Kotak Pembatas
  $(w, h) = (727 - 566, 677 - 627)$
  $(w, h) = (161, 50)$
- Kelas Objek
  $c = 1 - - >$ Plat Mobil



Gambar 4.7: Keterangan Kotak Pembatas

Hal yang perlu diperhatikan karena dataset *image* yang dikumpulkan memiliki dimensi yang berbeda-beda, sedangkan pada jaringan YoloV3 input *image* 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 tersebut, hal ini bertujuan agar menjaga keterangan kotak pembatas dalam dimensi citra yang berbeda sesuai dengan proporsinya. Sehingga keterangan kotak pembatas yang baru adalah :

- Kelas Objek $(c) = 1$ (Plat Mobil)
- $(x, y) = (\frac{646.5}{1440}, \frac{652}{1080}) = (0, 448958, 0.603704)$
- $(w, h) = (\frac{161}{1440}, \frac{50}{1080}) = (0.111806, 0.046296)$



Gambar 4.8: Keterangan Kotak Pembatas yang Baru

5. Tulis dan Simpan .txt
   Setalah keterangan kotak pembatas didapatkan, proses selanjutnya adalah menyimpan keterangan tersebut dalam file *.txt* dan menyimpan pada direktori dataset yang sama.

4.4 Parameter *Training*

Parameter yang digunakan ketika proses *training* dapat dilihat pada tabel 4.2 seperti berikut.

Tabel 4.2: Parameter *Training* YOLOV3

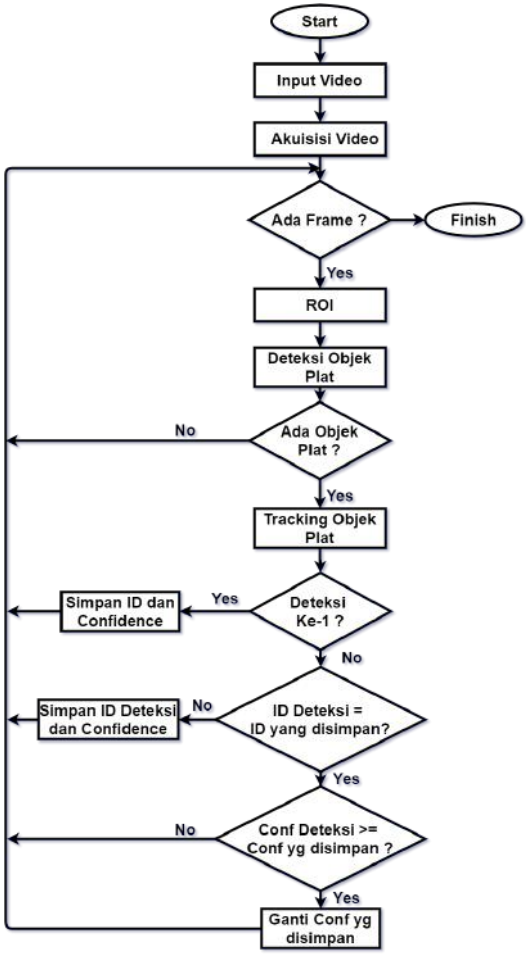
Parameter	Nilai	Parameter	Nilai	Parameter	Nilai
batch	64	momentum	0.9	angle	0
subdivisions	16	decay	0.0005	saturation	1.5
width	416	learning_rate	0.001	exposure	1.5
height	416	burn_in	1000	hue	0.1
channels	3	max_batches	50000		
		policy	steps		
		steps	40000,45000		
		scales	0.1,0.1		

Proses training diatur dengan *batch* sebesar 64 dan *subdivisions* 16. Pengaturan *batch* dan *subdivisions* diperlukan untuk mengatur banyak sampel data yang akan diproses untuk setiap satu iterasi. Selanjutnya seluruh citra input akan dikonversi ukurannya menjadi $416 \times 416 \times 3$. Untuk mencegah lonjakan perubahan pada saat pembaharuan bobot, parameter *momentum* diatur sebesar 0.9 dan *decay* diatur sebesar 0.0005 untuk menghindari *overfitting*. Dilanjutkan dengan nilai *learning rate* awal sebesar 0.001. Pada saat awal proses *training*, ada kemungkinan data yang diberikan saat proses *training* sangat berbeda antara satu data dengan yang lain. Oleh karena itu, untuk mencegah *overfitting* di awal. *learning rate* harus diatur lebih kecil. Maka parameter *burn in* diatur 1000 dan didapatkan *learning rate* saat iterasi di bawah 1000 ($i < 1000$) sebesar $\textit{learning rate}_{new} = \textit{learning rate}_{old} \times \left(\frac{i}{\textit{burn in}}\right)^2$. Saat proses *training* sudah lama berlangsung, *learning rate* perlu diatur menjadi lebih kecil untuk mencapai bobot yang optimum. Oleh karena itu ketika proses training sudah mencapai iterasi ke 40000 dan 45000, *learning rate* akan

diperbaharui menjadi *learning rate*$_{new}$ = *learning rate*$_{old}$ × *scales*. Untuk mendapatkan hasil pengenalan yang optimal, pada saat proses *training*, citra dari dataset akan dipilih secara acak untuk dirubah warnanya dengan mengatur *exposure*, *saturation*, dan *hue*.

4.5 Proses Pendeteksian Plat Nomor Kendaraan

Sebelum membahas tiap proses pendeteksian plat nomor kendaraan, pada subbab ini akan dijelaskan secara garis besar alur dalam pendeteksian plat nomor dengan inputan video. Adapun proses garis besar tersebut ditunjukkan pada gambar blok diagram berikut.



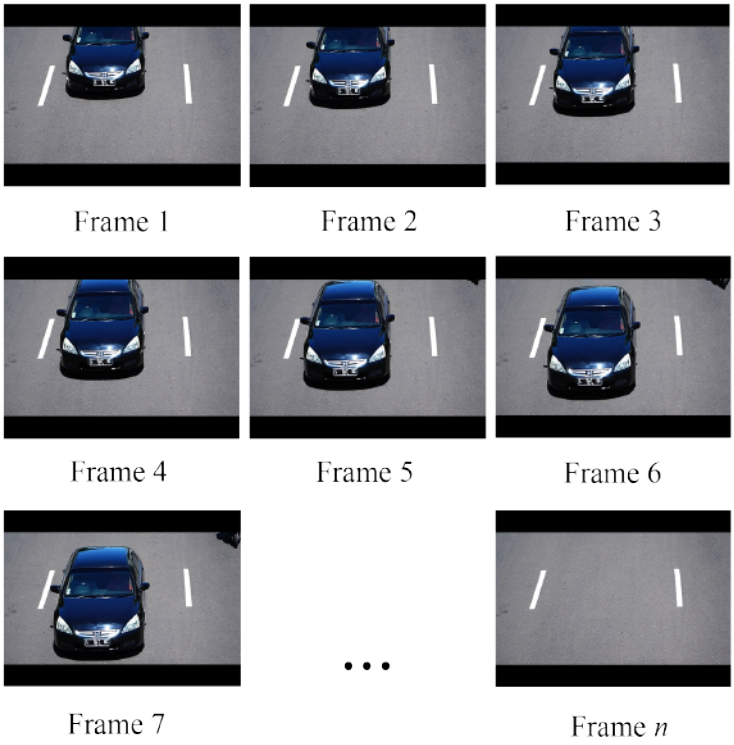
Gambar 4.9: Blok Diagram Proses Pendeteksian Plat

Pada proses di atas dapat dilihat bahwa proses pendeteksian plat dilakukan tiap *frame* dan diambil pixel-pixel input sesuai dengan ukuran ROI yang digunakan. Selanjutnya pixel yang diambil tersebut dilakukan

pendeteksian menggunakan Yolov3 dan diberi ID pada objek plat dalam proses *tracking*. Yolov3 mendeteksi dan menghasilkan sebuah prediksi *bounding box* objek plat yang memiliki parameter koordinat titik tengah *box*, ukuran *box*, prediksi kelas objek plat dan nilai *confidence* dari objek plat. parameter-parameter ini akan disimpan dan dibandingkan pada deteksi selanjutnya dalam menentukan ID objek plat dan lokalisasi dari plat nomor kendaraan. Proses di atas terlihat bahwa dalam menentukan lokalisasi plat nomor kendaraan digunakan nilai prediksi *confidence* yang paling tinggi pada setiap objek plat nomor yang masuk ke ROI sampai keluar pada ROI.

4.5.1 Akuisisi Video

Pada proses ini data masukkan berupa rekaman video digital secara *offline* dan dilakukan proses ekstraksi menjadi *frame-frame* gambar. Ektraksi *frame* ini diperlukan karena video hanya bisa diolah ketika dalam bentuk *frame - frame* gambar. untuk lebih jelasnya perhatikan Gambar 4.10.



Gambar 4.10: Proses Akuisis Video

4.5.2 Pendefinisian *Region of Interest* (ROI)

Selanjutnya *frame* yang telah diekstraksi akan dilakukan pendefinisian ROI. ROI berguna mengurangi beban komputasi karena tidak semua piksil dalam *frame* diolah. Langkah pertama dalam penentuan ROI yaitu dengan membuat bentuk segi empat pada bidang citra *frame* video. Dari bentuk segi empat tersebut akan didapatkan posisi titik TL (*top left*) yaitu (x_1, y_1) dan BR (*bottom right*) yaitu

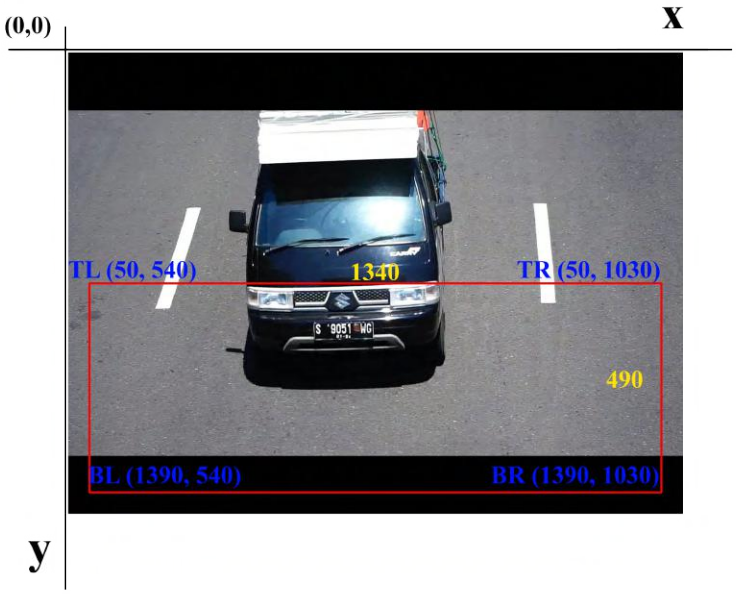
(x_2, y_2). Sehingga dapat ditentukan posisi kedua titik lainnya yaitu TR (*top right*) pada (x_2, y_1) dan BL (*bottom left*) pada (x_1, y_2) seperti pada Gambar 4.11.

Garis dari titik TL dan TR akan digunakan sebagai awal dari pendeteksian objek yang selanjutnya akan dipasangkan dan dilacak pada setiap *frame*. Sedangkan garis dari titik BL hingga BR akan digunakan sebagai garis akhir dari pendeteksian. Sheingga pendeteksian kendaraan bergerak hanya dilakukan pada area ROI.

Misalkan diambil (x_1, y_1) sebagai titik TL adalah $(50, 540)$ dengan panjang ROI adalah 1340 dan lebar 490, sehingga didapatkan titik BR yaitu

$$\begin{aligned}(x_2, y_2) &= (x_1 + panjang, y_1 + lebar) \\ &= (50 + 1340, 540 + 490) \\ &= (1390, 1030)\end{aligned}$$

Sehingga didapat juga kedua titk lainnya yaitu TR pada $(x_2, y_1) = (1390, 540)$ dan BL pada $(x_1, y_2) = (50, 1030)$.



Gambar 4.11: Frame Setelah Didefinisikan ROI

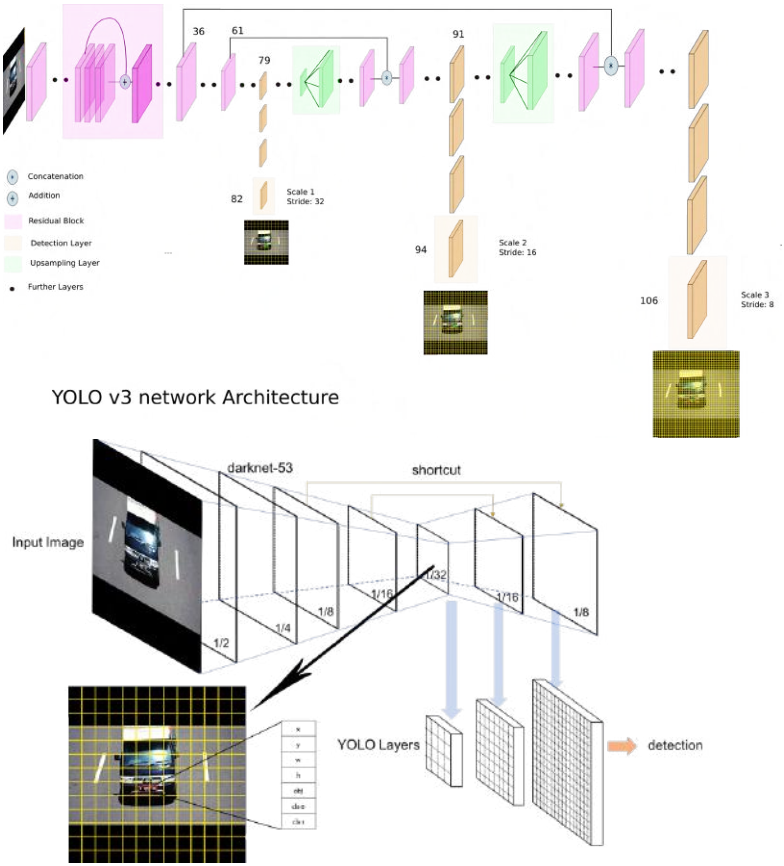
Gambar 4.11 adalah *frame* gambar yang telah didefinisikan ROI. terlihat dalam gambar tersebut terdapat kotak merah yang merupakan ROI. Kemudian kotak merah tersebut akan diambil sebagai *frame* baru seperti pada Gambar 4.12. *Frame* tersebut yang kemudian diolah untuk dilakukan pendeteksian objek plat nomor kendaraan.

Gambar 4.12: Frame ROI

Pada prinsipnya semakin besar ROI yang digunakan maka semakin besar pula beban komputsainya sehingga waktu yang digunakan dalam pendeteksian akan semakin lama tapi keakuratannya semakin bagus. Namun jika semakin kecil ROI yang digunakan maka semakin kecil pula beban komputasinya sehingga waktu yang digunakan dalam pendeteksian akan semakin cepat tapi keakuratannya semakin jelek.

4.5.3 Deteksi Objek Plat dengan Arsitektur Jaringan YoloV3

Desain arsitektur YoloV3 pada prinsipnya menggunakan Darknet53 sebagai jaringan dasar. Sesuai dengan namanya, jaringan ini memiliki 53 lapisan yang secara berturut-turut teridiri dari proses konvolusi dengan filters 3×3 dan 1×1 dan proses residu. Selanjutnya pada jaringan Darknet53 ini ditambahkan beberapa lapisan konvolusional yang mengakibatkan terbentuknya arsitektur 106 lapisan *fully convolution*, jaringan ini lah yang disebut jaringan YoloV3. Adapun desain arsitektur jaringan YoloV3 yang digunakan pada penelitian tugas akhir ini ditunjukkan pada Gambar 4.13 dan Tabel 4.3.

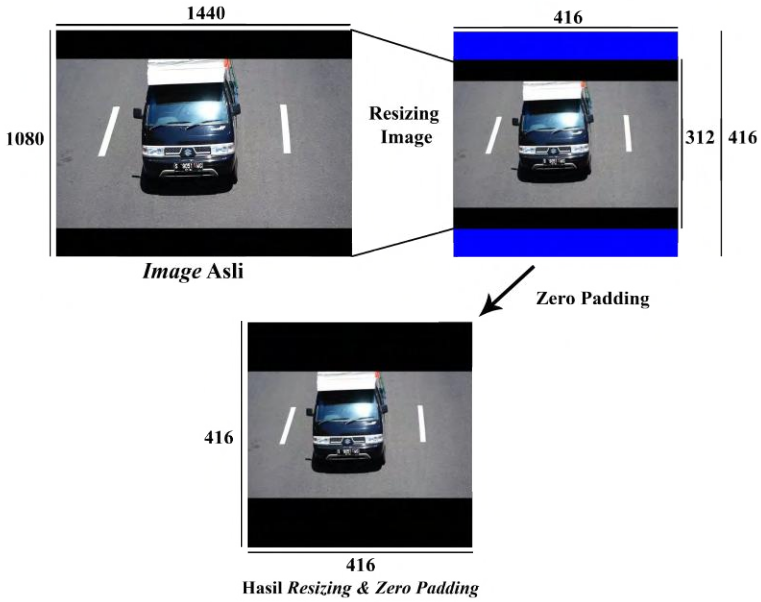


Gambar 4.13: Arsitektur Jaringan YoloV3

Tabel 4.3: Arsitektur Jaringan YOLOV3

	Layer	Total Filter	Ukuran Filter	Stride	Fungsi Aktivasi	Input	Output
0	Convolutional	32	3x3	1	leaky	416x416x3	416x416x32
1	Convolutional	64	3x3	2	leaky	416x416x32	208x208x64
2-4	Residual Unit					208x208x64	208x208x64
5	Convolutional	128	3x3	2	leaky	208x208x64	104x104x128
6-8	Residual Unit					104x104x128	104x104x128
9-11	Residual Unit					104x104x128	104x104x128
12	Convolutional	256	3x3	2	leaky	104x104x128	52x52x256
13-15	Residual Unit					52x52x256	52x52x256
16-18	Residual Unit					52x52x256	52x52x256
19-21	Residual Unit					52x52x256	52x52x256
22-24	Residual Unit					52x52x256	52x52x256
25-27	Residual Unit					52x52x256	52x52x256
28-30	Residual Unit					52x52x256	52x52x256
31-33	Residual Unit					52x52x256	52x52x256
34-36	Residual Unit					52x52x256	52x52x256
37	Convolutional	512	3x3	2	leaky	52x52x256	26x26x512
38-40	Residual Unit					26x26x512	26x26x512
41-43	Residual Unit					26x26x512	26x26x512
44-46	Residual Unit					26x26x512	26x26x512
47-49	Residual Unit					26x26x512	26x26x512
50-52	Residual Unit					26x26x512	26x26x512
53-55	Residual Unit					26x26x512	26x26x512
56-58	Residual Unit					26x26x512	26x26x512
59-61	Residual Unit					26x26x512	26x26x512
62	Convolutional	1024	3x3	2	leaky	26x26x512	13x13x1024
63-65	Residual Unit					13x13x1024	13x13x1024
66-68	Residual Unit					13x13x1024	13x13x1024
69-71	Residual Unit					13x13x1024	13x13x1024
72-74	Residual Unit					13x13x1024	13x13x1024
75	Convolutional	512	1x1	1	leaky	13x13x1024	13x13x512
76	Convolutional	1024	3x3	1	leaky	13x13x512	13x13x1024
77	Convolutional	512	1x1	1	leaky	13x13x1024	13x13x512
78	Convolutional	1024	3x3	1	leaky	13x13x512	13x13x1024
79	Convolutional	512	1x1	1	leaky	13x13x1024	13x13x512
80	Convolutional	1024	3x3	1	leaky	13x13x512	13x13x1024
81	Convolutional	18	1x1	1	linear	13x13x1024	13x13x18
82	YOLO						
83	Route - 79					13x13x512	13x13x512
84	Convolutional	256	1x1	1	leaky	13x13x512	13x13x256
85	Upsample					13x13x256	26x26x256
86	Route - 85 61					26x26x768	26x26x768
87	Convolutional	256	1x1	1	leaky	26x26x768	26x26x256
88	Convolutional	512	3x3	1	leaky	26x26x256	26x26x512
89	Convolutional	256	1x1	1	leaky	26x26x512	26x26x256
90	Convolutional	512	3x3	1	leaky	26x26x256	26x26x512
91	Convolutional	256	1x1	1	leaky	26x26x512	26x26x256
92	Convolutional	512	3x3	1	leaky	26x26x256	26x26x512
93	Convolutional	18	1x1	1	linear	26x26x256	26x26x18
94	YOLO						
95	Route -91					26x26x256	26x26x256
96	Convolutional	128	1x1	1	leaky	26x26x256	26x26x128
97	Upsample					26x26x128	52x52x128
98	Route - 97 36					52x52x384	52x52x384
99	Convolutional	128	1x1	1	leaky	52x52x384	52x52x128
100	Convolutional	256	3x3	1	leaky	52x52x128	52x52x256
101	Convolutional	128	1x1	1	leaky	52x52x256	52x52x128
102	Convolutional	256	3x3	1	leaky	52x52x128	52x52x256
103	Convolutional	128	1x1	1	leaky	52x52x256	52x52x128
104	Convolutional	256	3x3	1	leaky	52x52x128	52x52x256
105	Convolutional	18	1x1	1	linear	52x52x256	52x52x18
106	YOLO						

Dalam arsitektur jaringan YoloV3 hal pertama yang perlu diperhatikan adalah proses input *image*, asumsikan input *image* merupakan *frame* hasil ROI. Pada jaringan YoloV3 ini setiap *image* yang masuk dilakukan *resizing image* berukuran 416 × 416. Akurasi pendeteksian semakin baik apabila *image* sudah berukuran 416 × 416 tapi, jika *image* memiliki dimensi yang berbeda maka *image* tersebut dilakukan *resizing image* berukuran 416 × 416 dengan tetap menjaga proporsi *image* tersebut. Jika *image* memiliki dimensi (1440 × 1080) maka hasil *resizing image* adalah (416 × 312). Karena hasil *resizing image* 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. Untuk lebih jelasnya proses input *image* dapat dilihat pada Gambar 4.14.



Gambar 4.14: Proses Input Image pada YoloV3

Jaringan YoloV3 mempunyai jumlah layer sebanyak 106 *layer* yang terbagi menjadi beberapa proses diantaranya proses *Convolutional Layer*, *Shortcut Layer* atau *Residual*, *Upsample*, *Route*, dan *Detections at Three Scales*.

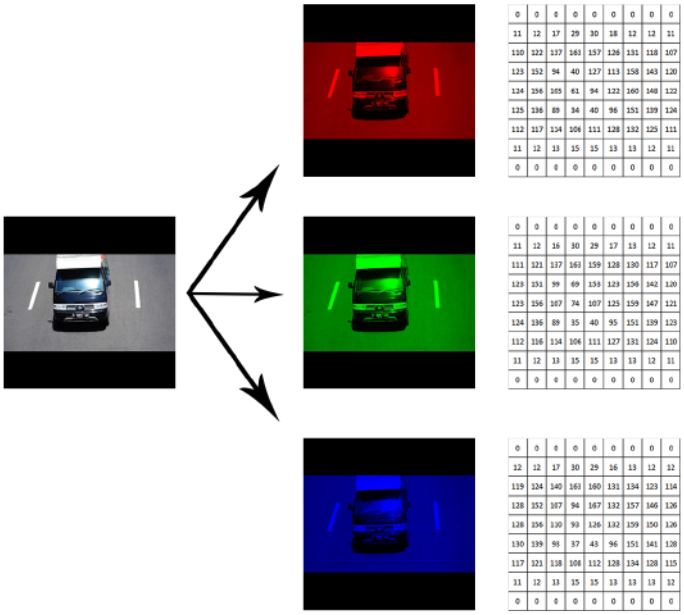
1. **Convolutional Layer**
   Pada proses ini, jaringan akan melalui proses *Convolutional 2D Layer*, *Batch Normalization Layer* dan *Leaky ReLU Layer*.

   (a) Convolutional 2D Layer
   Pada proses *Convolutional2D Layer* ini inputan *image* yang sudah berukuran (416 × 416) akan

dilakukan proses konvolusi pada tiap *channel*, dengan menggerakan sebuah kernel konvolusi (filter) berukuran tertentu ke sebuah gambar, sehingga didapatkan informasi representatif baru dari hasil perkalian bagian *image* tersebut dengan filter yang digunakan. Adapun proses pada layar pertama jaringan YoloV3 seperti di bawah ini.

Input	416 x 416 x 3
Proses	*Filters* = 32
	Stride = 1
	Size Filters = 3 x 3
Output	416 x 416 x 32

Sebelum dilakukan konvolusi, inputan *image* akan dipecah menjadi 3 bagian *channels* yaitu *image Red, Green* dan *Blue* yang selanjutnya tiap *channels* ini dilakukan proses konvolusi.



Gambar 4.15: Proses input *image* menjadi *image* RGB

Adapaun hal-hal yang perlu diperhatikan ketika proses konvolusi pada jaringan YoloV3 ini adalah :

- Ukuran ketebalan dari sebuah filter selalu mengikuti ketebalan/volume dari gambar input yang digunakan.
- Tinggi dan lebar filter (F) pada umumnya berukuran ganjil. Secara intuisi, filter berukuran ganjil memberikan representasi yang lebih baik karena dapat mencakup bagian kiri dan kanan yang "seimbang".

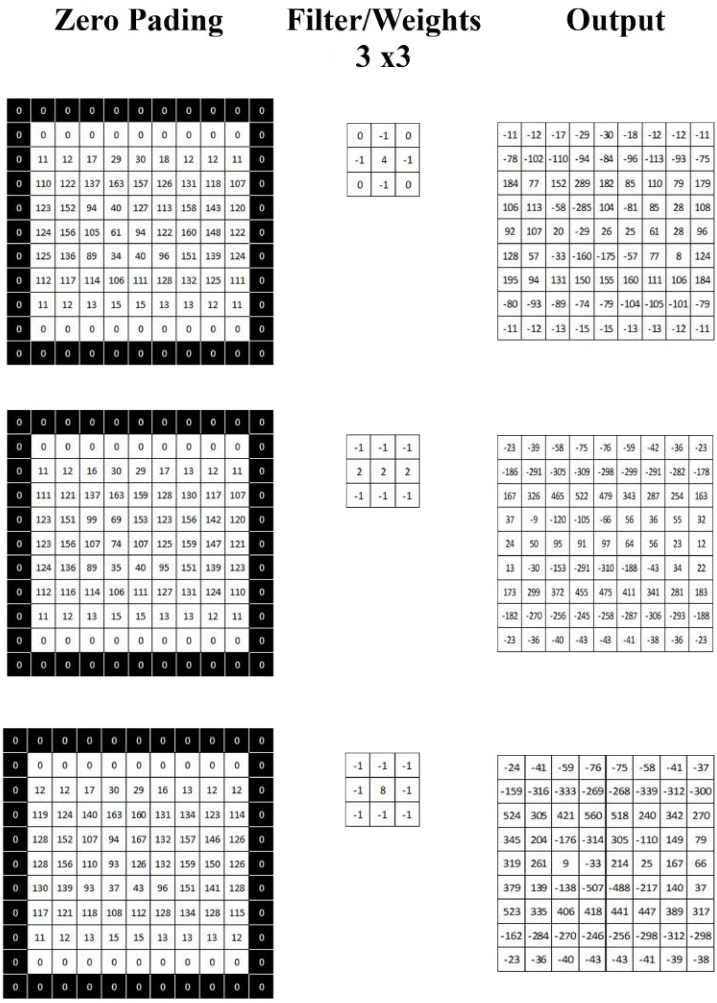
- Dalam sebuah *convolutional 2D layer*, filter-filter yang digunakan berukuran sama, untuk kemudahan proses komputasi.
- Jumlah filter (K) yang digunakan dalam sebuah *convolutional layer* adalah kelipatan 2(*powers of 2*). Beberapa *library* memiliki *subroutine* khusus untuk komputasi kernel/dimensi berkelipatan 2 yang dapat meningkatkan efisiensi.
- Besaran *zero padding* (P) umumnya menyesuaikan agar ukuran spasial dari output yang dihasilkan tetap sama dengan ukuran spasial input. ($P = \frac{F-1}{2}$).

0	0	0	0	0	0	0	0	0
11	12	17	29	30	18	12	12	11
110	122	137	163	157	126	131	118	107
123	152	94	40	127	113	158	143	120
124	156	105	61	94	122	160	148	122
125	136	89	34	40	96	151	139	124
112	117	114	106	111	128	132	125	111
11	12	13	15	15	13	13	12	11
0	0	0	0	0	0	0	0	0

Zero Padding

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	11	12	17	29	30	18	12	12	11	0
0	110	122	137	163	157	126	131	118	107	0
0	123	152	94	40	127	113	158	143	120	0
0	124	156	105	61	94	122	160	148	122	0
0	125	136	89	34	40	96	151	139	124	0
0	112	117	114	106	111	128	132	125	111	0
0	11	12	13	15	15	13	13	12	11	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

Gambar 4.16: Contoh Proses Zero Padding



Gambar 4.17: Proses *Convolutional 2D Layer* pada Layer Pertama

Pada proses Convolutional 2D Layer filter/weights didapatkan secara acak dari model yang sudah dilatih pada suatu dataset. Secara keseluruhan, bila input sebuah *convolutional layer* adalah gambar dengan ukuran $W1 \times H1 \times D1$ dengan $W1 = H1$, output dari *layer* tersebut adalah sebuah *image* baru dengan ukuran $W2 \times H2 \times D2$ dengan $W2 = H2$, dimana :

$$W2 = \frac{(W1 - F + 2P)}{S} + 1 = H2$$
$$D2 = K$$

dimana :

- K adalah jumlah filter yang digunakan.
- F adalah ukuran spasial dari filter (lebar/tinggi).
- S *stride*, atau besar pergeseran filter dalam konvolusi.
- P adalah *padding*, jumlah penambahan nol pada *image*

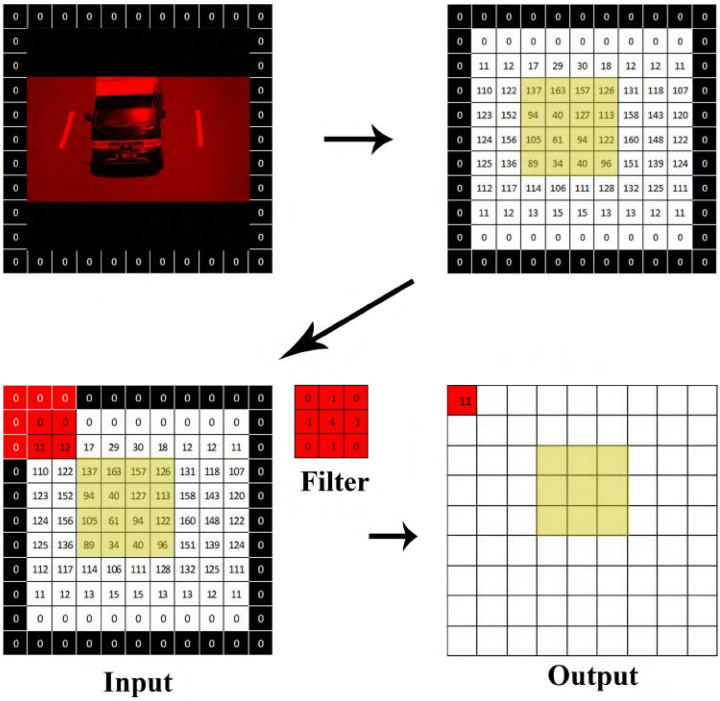
Sehingga pada layer pertama jaringan YoloV3 Didapatkan :

- **Input**
  $W1 \times H1 \times D1 = 416 \times 416 \times 3$
- **Proses**
  $F = 3$,
  $S = 1$,
  $P = \frac{F-1}{2} = \frac{3-1}{2} = 1$, dan
  $K = 32$
- **Output**
  $W2 = H2 = \frac{(416 - 3 + 2 \times 1)}{1} + 1 = 416$

$D2 = 32$
$W2 \times H2 \times D2 = 416 \times 416 \times 32$

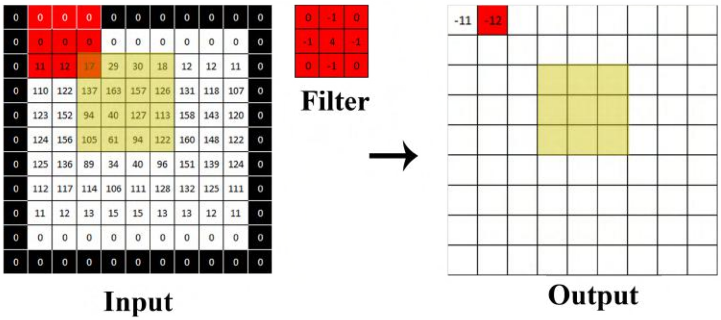
Operasi *convolutional 2D layer* dilakukan dengan menggeser kernel konvolusi *pixel* per *pixel*. Hasil konvolusi disimpan di dalam matriks yang baru. Operasi *Convolutional 2D Layer* pada inputan *image "RED"* diilustarsikan sebagai berikut :

i. Tempatkan kernel pada sudut kiri atas, kemudian hitung nilai *pixel* pada posisi $(0, 0)$ dari filter/kernel/weights. Hasil konvolusi pada posisi $(0, 0) = -11$. Nilai ini dihitung dengan cara berikut :
$(0 \times 0) + (-1 \times 0) + (0 \times 0) + (-1 \times 0) + (4 \times 0) + (-1 \times 0) + (0 \times 0) + (-1 \times 11) + (0 \times 12) = -11$.
Proses ini ditunjukkan pada Gambar 4.18.



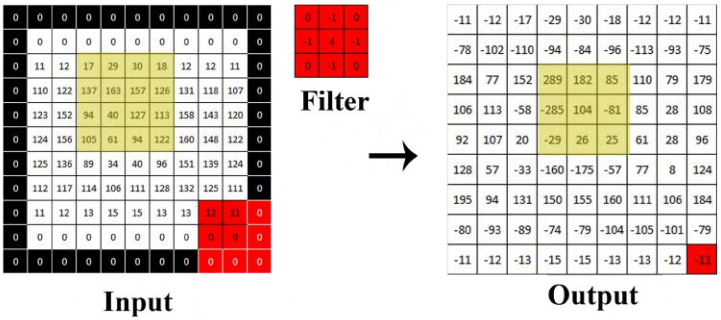
Gambar 4.18: Proses Konvolusi pada Posisi $(0, 0)$

ii. Geser kernel/filter sebesar nilai *stride* (*stride* = 1) pixel ke kanan, kemudian hitung nilai pixel pada posisi $(0, 1)$ dari kernel/filter. Hasil konvolusi pada posisi $(0, 1)$ $= -12$. Nilai ini dihitung dengan cara berikut: $(0 \times 0) + (-1 \times 0) + (0 \times 0) + (-1 \times 0) + (4 \times 0) + (-1 \times 0) + (0 \times 11) + (-1 \times 12) + (0 \times 17) = -12$. Proses ini ditunjukkan pada Gambar 4.19.

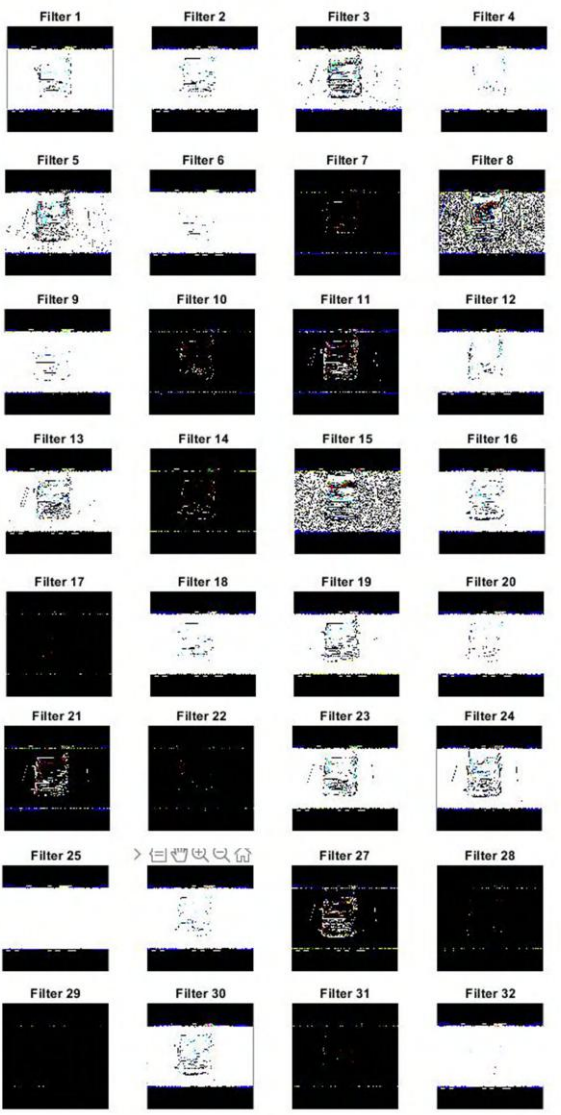


Gambar 4.19: Proses Konvolusi pada Posisi $(0, 1)$

iii. Dengan cara yang sama seperti di atas, maka *pixel-pixel* hasil proses konvolusi menjadi :

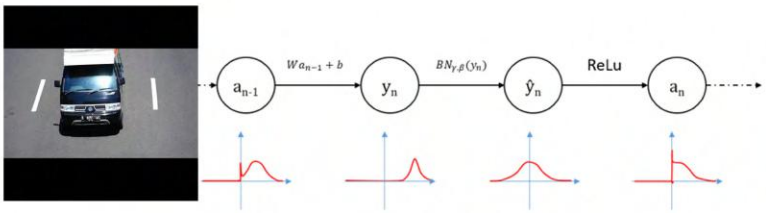


Gambar 4.20: Hasil Konvolusi pada setiap pixel

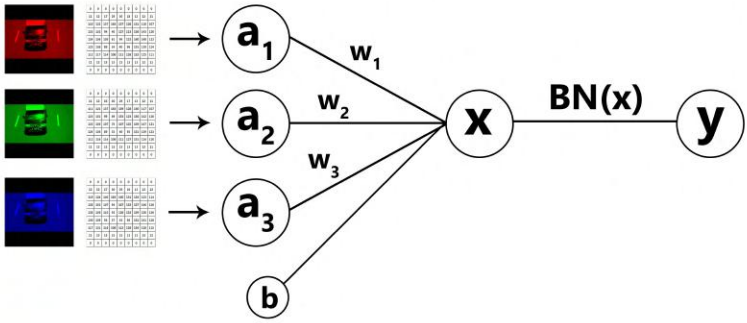


Gambar 4.21: Hasil *Convolutional 2D Layer* pada Layer Pertama

(b) Batch Normalization Layer
Proses *Batch Normalization* berfungsi dalam mempercepat pelatihan *Deep Neural Network* dengan mengurangi internal *covariate shift*, 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 saraf.



Gambar 4.22: Internal *Covariate Shift*



Gambar 4.23: Ilustrasi *Neural Network* pada layer Pertama

Pada Gambar 4.23 terdapat suatu channel yaitu bias yang berfungsi untuk mengontrol *variance* data. Adapun nilai aktivasi x_i pada layer pertama jaringan YoloV3 ini ditunjukkan pada persamaan 4.1 seperti berikut :

$$x = \sum_{i=1}^{n} (a_i * w_i + b) \tag{4.1}$$

Keterangan :
$a_i * w_i$ = Operasi Conv2D.
Adapun algoritma transformasi normalisasi *batch* pada aktivasi x dapat ditunjukkan seperti di bawah ini [16] :

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

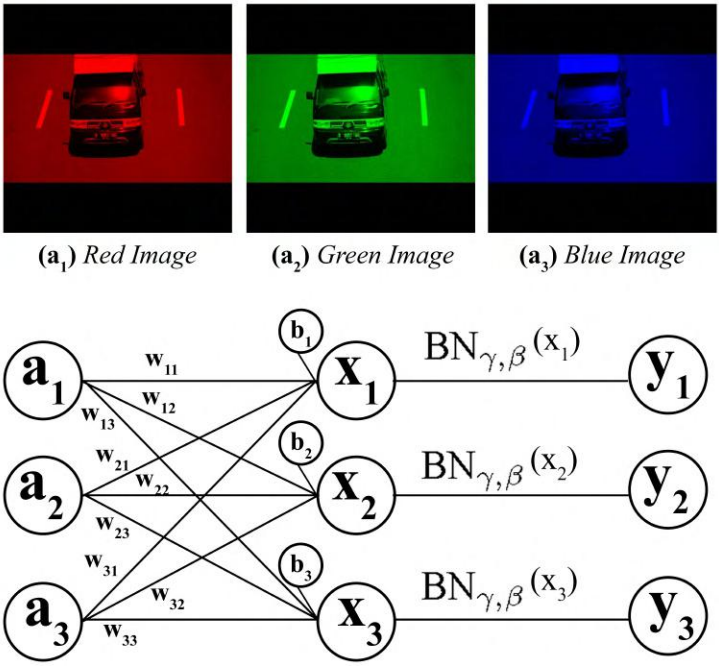
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

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

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

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

Sebagai ilustrasi proses *Batch Normalizaton* perhatikan Gambar 4.24



Gambar 4.24: Gambar yang Akan Dilakukan Batch Normalization

Pada Gambar 4.24 *image* direpresentaskan secara berturut - turut menjadi matriks sebagai berikut :

$$a_1 = \begin{pmatrix} 6 & 8 & 5 & 8 & 6 \\ 0 & 9 & 0 & 2 & 3 \\ 6 & 7 & 1 & 5 & 0 \\ 6 & 5 & 8 & 6 & 4 \\ 7 & 9 & 4 & 0 & 1 \end{pmatrix}, a_2 = \begin{pmatrix} 1 & 6 & 5 & 8 & 4 \\ 2 & 2 & 4 & 2 & 2 \\ 1 & 5 & 1 & 2 & 9 \\ 1 & 6 & 4 & 5 & 0 \\ 0 & 4 & 8 & 6 & 1 \end{pmatrix},$$

$$a_3 = \begin{pmatrix} 0 & 4 & 8 & 9 & 8 \\ 8 & 0 & 1 & 2 & 1 \\ 5 & 1 & 9 & 4 & 1 \\ 8 & 6 & 5 & 4 & 3 \\ 3 & 3 & 7 & 7 & 0 \end{pmatrix}$$

Sedangkan *Weights* dan *Bias* yang digunakan adalah :

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

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

$$w_{31} = w_{22} = w_{13} = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix},$$

$b_1 = b_2 = b_3 = 1$

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* dengan besaran *zero padding* dirumuskan $(P = \dfrac{F-1}{2})$ dengan $F =$ ukuran *weights*(w).

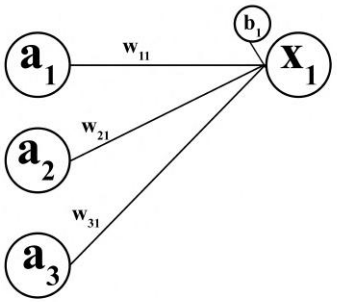
$F = 3, P = \dfrac{3-1}{2} = 1$

$$a_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 8 & 5 & 8 & 6 & 0 \\ 0 & 0 & 9 & 0 & 2 & 3 & 0 \\ 0 & 6 & 7 & 1 & 5 & 0 & 0 \\ 0 & 6 & 5 & 8 & 6 & 4 & 0 \\ 0 & 7 & 9 & 4 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$a_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 6 & 5 & 8 & 4 & 0 \\ 0 & 2 & 2 & 4 & 2 & 2 & 0 \\ 0 & 1 & 5 & 1 & 2 & 9 & 0 \\ 0 & 1 & 6 & 4 & 5 & 0 & 0 \\ 0 & 0 & 4 & 8 & 6 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$a_3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 8 & 9 & 8 & 0 \\ 0 & 8 & 0 & 1 & 2 & 1 & 0 \\ 0 & 5 & 1 & 9 & 4 & 1 & 0 \\ 0 & 8 & 6 & 5 & 4 & 3 & 0 \\ 0 & 3 & 3 & 7 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- Menghitung Aktivasi x_1



dengan mengunakan persamaan 4.1 didapatkan :

$$x_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 8 & 5 & 8 & 6 & 0 \\ 0 & 0 & 9 & 0 & 2 & 3 & 0 \\ 0 & 6 & 7 & 1 & 5 & 0 & 0 \\ 0 & 6 & 5 & 8 & 6 & 4 & 0 \\ 0 & 7 & 9 & 4 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} *$$

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 6 & 5 & 8 & 4 & 0 \\ 0 & 2 & 2 & 4 & 2 & 2 & 0 \\ 0 & 1 & 5 & 1 & 2 & 9 & 0 \\ 0 & 1 & 6 & 4 & 5 & 0 & 0 \\ 0 & 0 & 4 & 8 & 6 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} *$$

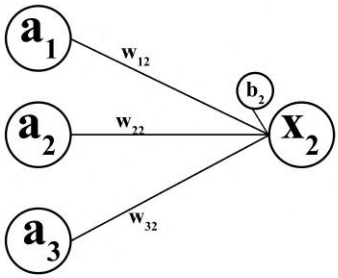
$$\begin{pmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 8 & 9 & 8 & 0 \\ 0 & 8 & 0 & 1 & 2 & 1 & 0 \\ 0 & 5 & 1 & 9 & 4 & 1 & 0 \\ 0 & 8 & 6 & 5 & 4 & 3 & 0 \\ 0 & 3 & 3 & 7 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} *$$

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} + \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$x_1 = \begin{pmatrix} 16 & 12 & 4 & 19 & 13 \\ -21 & 21 & -17 & -8 & 4 \\ 11 & 7 & -16 & 11 & -12 \\ 6 & -10 & 16 & 7 & 9 \\ 13 & 20 & -1 & -11 & 0 \end{pmatrix} +$$

$$\begin{pmatrix} 10 & 16 & 30 & 26 & 20 \\ -5 & -3 & -11 & -13 & -15 \\ 1 & -5 & -7 & 7 & 13 \\ 4 & 3 & 4 & -9 & -8 \\ 1 & 13 & 21 & 21 & 9 \end{pmatrix} +$$
$$\begin{pmatrix} -12 & 15 & 48 & 52 & 52 \\ 54 & -36 & -29 & -25 & -16 \\ 17 & -34 & 49 & 6 & -6 \\ 46 & 7 & -1 & -4 & 8 \\ 7 & -5 & 31 & 37 & -14 \end{pmatrix} +$$
$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$
$$x_1 = \begin{pmatrix} 15 & 44 & 83 & 98 & 86 \\ 29 & -17 & -56 & -45 & -26 \\ 30 & -31 & 27 & 25 & -4 \\ 57 & 1 & 20 & -5 & 10 \\ 22 & 29 & 52 & 48 & -4 \end{pmatrix}$$

- Menghitung Aktivasi x_2



Dengan cara yang sama dengan aktivasi x_1 dan mengunakan persamaan 4.1 didapatkan :

$$x_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 8 & 5 & 8 & 6 & 0 \\ 0 & 0 & 9 & 0 & 2 & 3 & 0 \\ 0 & 6 & 7 & 1 & 5 & 0 & 0 \\ 0 & 6 & 5 & 8 & 6 & 4 & 0 \\ 0 & 7 & 9 & 4 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} *$$

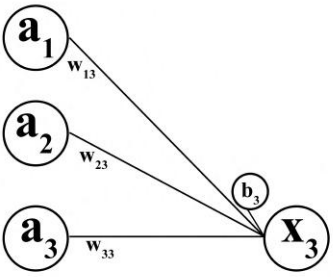
$$\begin{pmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 6 & 5 & 8 & 4 & 0 \\ 0 & 2 & 2 & 4 & 2 & 2 & 0 \\ 0 & 1 & 5 & 1 & 2 & 9 & 0 \\ 0 & 1 & 6 & 4 & 5 & 0 & 0 \\ 0 & 0 & 4 & 8 & 6 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} *$$

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 8 & 9 & 8 & 0 \\ 0 & 8 & 0 & 1 & 2 & 1 & 0 \\ 0 & 5 & 1 & 9 & 4 & 1 & 0 \\ 0 & 8 & 6 & 5 & 4 & 3 & 0 \\ 0 & 3 & 3 & 7 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} *$$

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} + \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$x_2 = \begin{pmatrix} 6 & 72 & 68 & 99 & 66 \\ 20 & -37 & -25 & -40 & -24 \\ 2 & 4 & 0 & -21 & 53 \\ 4 & 36 & 2 & 32 & -1 \\ 12 & 31 & 60 & 40 & -20 \end{pmatrix}$$

- Menghitung Aktivasi x_3



Dengan cara yang sama dengan aktivasi x_1 dan mengunakan persamaan 4.1 didapatkan :

$$x_3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 8 & 5 & 8 & 6 & 0 \\ 0 & 0 & 9 & 0 & 2 & 3 & 0 \\ 0 & 6 & 7 & 1 & 5 & 0 & 0 \\ 0 & 6 & 5 & 8 & 6 & 4 & 0 \\ 0 & 7 & 9 & 4 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} *$$

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 6 & 5 & 8 & 4 & 0 \\ 0 & 2 & 2 & 4 & 2 & 2 & 0 \\ 0 & 1 & 5 & 1 & 2 & 9 & 0 \\ 0 & 1 & 6 & 4 & 5 & 0 & 0 \\ 0 & 0 & 4 & 8 & 6 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} *$$

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 8 & 9 & 8 & 0 \\ 0 & 8 & 0 & 1 & 2 & 1 & 0 \\ 0 & 5 & 1 & 9 & 4 & 1 & 0 \\ 0 & 8 & 6 & 5 & 4 & 3 & 0 \\ 0 & 3 & 3 & 7 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} *$$

$$\begin{pmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{pmatrix} + \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$x_3 = \begin{pmatrix} 28 & 76 & 55 & 116 & 73 \\ -25 & 22 & -67 & -50 & -19 \\ 8 & 34 & -34 & 20 & 13 \\ 28 & 13 & 23 & 30 & 8 \\ 30 & 52 & 42 & 4 & 4 \end{pmatrix}$$

Karena x_1, x_2, dan x_3 telah diketahui selanjutnya akan dicari nilai dari μ_B dan σ_B^2.

$\mu_B = \frac{1}{3} \sum_{i=1}^{3} x_i$

$$\mu_B = \frac{1}{3} \times (\begin{pmatrix} 15 & 44 & 83 & 98 & 86 \\ 29 & -17 & -56 & -45 & -26 \\ 30 & -31 & 27 & 25 & -4 \\ 57 & 1 & 20 & -5 & 10 \\ 22 & 29 & 52 & 48 & -4 \end{pmatrix} +$$

$$\begin{pmatrix} 6 & 72 & 68 & 99 & 66 \\ 20 & -37 & -25 & -40 & -24 \\ 2 & 4 & 0 & -21 & 53 \\ 4 & 36 & 2 & 32 & -1 \\ 12 & 31 & 60 & 40 & -20 \end{pmatrix} +$$

$$\begin{pmatrix} 28 & 76 & 55 & 116 & 73 \\ -25 & 22 & -67 & -50 & -19 \\ 8 & 34 & -34 & 20 & 13 \\ 28 & 13 & 23 & 30 & 8 \\ 30 & 52 & 42 & 4 & 4 \end{pmatrix})$$

$$\mu_B = \frac{1}{3} \times \begin{pmatrix} 49 & 192 & 206 & 313 & 225 \\ 24 & -32 & -148 & -135 & -69 \\ 40 & 7 & -7 & 24 & 62 \\ 89 & 50 & 45 & 57 & 17 \\ 64 & 112 & 154 & 92 & -20 \end{pmatrix}$$

$$\mu_B = \begin{pmatrix} 16.33 & 64 & 68.67 & 104.33 & 75 \\ 8 & -10.67 & -49.33 & -45 & -23 \\ 13.33 & 2.33 & -2.33 & 8 & 20.67 \\ 29.67 & 16.67 & 15 & 19 & 5.667 \\ 21.33 & 37.33 & 51.33 & 30.67 & -6.67 \end{pmatrix}$$

$$\sigma_B^2 = \frac{1}{3} \sum_{i=1}^{3} (x_i - \mu_B)^2$$

$$\sigma_B^2 = \begin{pmatrix} 81.56 & 202.67 & 130.89 & 68.22 & 68.67 \\ 558 & 600.22 & 316.22 & 16.67 & 8.67 \\ 144.89 & 705.56 & 622.89 & 424.67 & 570.89 \\ 469.56 & 210.89 & 86 & 288.67 & 22.89 \\ 54.22 & 108.22 & 54.22 & 366.22 & 99.56 \end{pmatrix}$$

Kemudian akan dicari nilai dari $\hat{x_1}, \hat{x_2}$ dan $\hat{x_3}$ seperti di bawah ini :

Menghitung Nilai $\hat{x_1}$

$$\hat{x_1} = \frac{x_1 - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

ϵ berfungsi untuk stabilitas matematika.

$$\hat{x_1} = \begin{pmatrix} -0.1476 & -1.4049 & 1.2528 & -0.7668 & 1.3275 \\ 0.8890 & -0.2585 & -0.3749 & 0 & -1.0190 \\ 1.3846 & -1.2549 & 1.1753 & 0.8249 & -1.0324 \\ 1.2614 & -1.0788 & 0.5392 & -1.4126 & 0.9058 \\ 0.0905 & -0.8011 & 0.0905 & 0.9058 & 0.2673 \end{pmatrix}$$

Menghitung Nilai $\hat{x_2}$

$$\hat{x_2} = \frac{x_2 - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

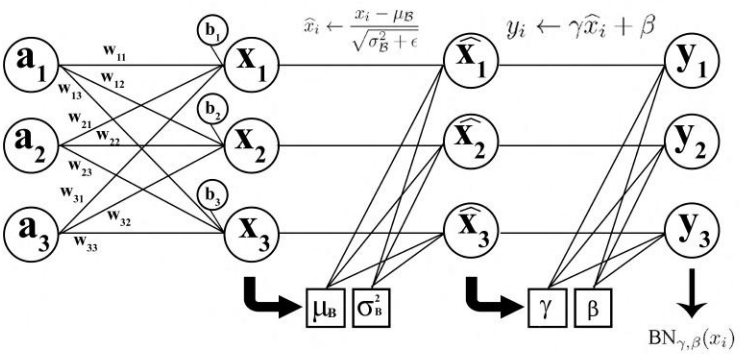
$$\hat{x_2} = \begin{pmatrix} -1.1442 & 0.5620 & -0.0583 & -0.6457 & -1.0861 \\ 0.5080 & -1.0749 & 1.3684 & 1.2247 & -0.3397 \\ -0.9415 & 0.0627 & 0.0935 & -1.4073 & 1.3532 \\ -1.1845 & 1.3313 & -1.4018 & 0.7651 & -1.3935 \\ -1.2675 & -0.6088 & 1.1770 & 0.4877 & -1.3363 \end{pmatrix}$$

Menghitung Nilai $\hat{x_3}$

$$\hat{x_3} = \frac{x_3 - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

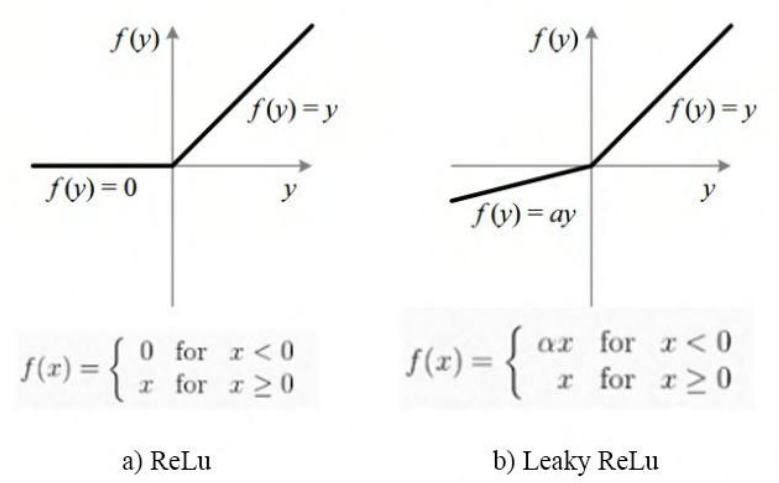
$$\hat{x_3} = \begin{pmatrix} 1.2919 & 0.8429 & -1.1946 & 1.4125 & -0.2414 \\ -1.3970 & 1.3334 & -0.9935 & -1.2247 & 1.3587 \\ -0.4431 & 1.1922 & -1.2688 & 0.5823 & -0.3209 \\ -0.0769 & -0.2525 & 0.8627 & 0.6474 & 0.4877 \\ 1.1770 & 1.4099 & -1.2675 & -1.3935 & 1.0690 \end{pmatrix}$$

Selanjutnya akan dicari nilai dari y_1, y_2 dan y_3 dengan menggunakan parameter γ dan β yang diperoleh dari pembelajaran dari model *transfer learning* Darknet YoloV3.weights yang telah dilatih pada dataset COCO. Parameter ini dilatih dan digunakan untuk mengatur nilai aktivasi agar tidak terjadi *internal covariate shift*. Jika diingkan nilai aktivasi yang asli, maka parameter dapat diatur menjadi $\gamma = \sqrt{\sigma_B^2 + \epsilon}$ dan $\beta = \mu_B$. Sehingga nilai $y_i = x_i$. Proses *Batch Normalization* pada jaringan *Neural Network* dapat dilihat pada Gambar 4.25.



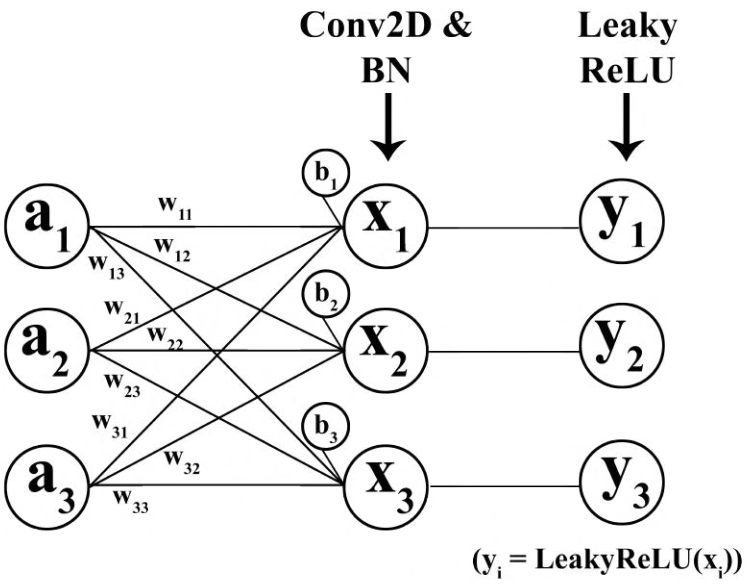
Gambar 4.25: Proses *Batch Normalization* pada *Neural Network*

(c) Leaky ReLU Layer
Pada jaringan YoloV3 fungsi aktivasi yang digunakan adalah fungsi *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.



Gambar 4.26: Perbedaan Grafik dari Fungsi ReLu dan Leaky ReLU

Leaky ReLU membantu untuk menambah *range* dari fungsi ReLU. Pada jaringan YoloV3 nilai dari α adalah 0.01. Proses aktivasi ReLU dapat diilustrasikan seperti pada Gambar 4.27



Gambar 4.27: Ilustrasi Proses LeakyReLU

- Mencari nilai y_1 dari nilai x_1 yang didapatkan pada proses *Batch Normalization*

$$x_1 = \begin{pmatrix} 15 & 44 & 83 & 98 & 86 \\ 29 & -17 & -56 & -45 & -26 \\ 30 & -31 & 27 & 25 & -4 \\ 57 & 1 & 20 & -5 & 10 \\ 22 & 29 & 52 & 48 & -4 \end{pmatrix}$$

$$y_1 = LeakyReLU(x_1)$$

$$y_1 = \begin{pmatrix} 15 & 44 & 83 & 98 & 86 \\ 29 & -0.17 & -0.56 & -0.45 & -0.26 \\ 30 & -0.31 & 27 & 25 & -0.04 \\ 57 & 1 & 20 & -0.05 & 10 \\ 22 & 29 & 52 & 48 & -0.04 \end{pmatrix}$$

- Mencari nilai y_2 dari nilai x_2 yang didapatkan pada proses *Batch Normalization*

$$x_2 = \begin{pmatrix} 6 & 72 & 68 & 99 & 66 \\ 20 & -37 & -25 & -40 & -24 \\ 2 & 4 & 0 & -21 & 53 \\ 4 & 36 & 2 & 32 & -1 \\ 12 & 31 & 60 & 40 & -20 \end{pmatrix}$$

$$y_2 = LeakyReLU(x_2)$$

$$y_2 = \begin{pmatrix} 6 & 72 & 68 & 99 & 66 \\ 20 & -0.37 & -0.25 & -0.4 & -0.24 \\ 2 & 4 & 0 & -0.21 & 53 \\ 4 & 36 & 2 & 32 & -0.01 \\ 12 & 31 & 60 & 40 & -0.2 \end{pmatrix}$$

- Mencari nilai y_3 dari nilai x_3 yang didapatkan pada proses *Batch Normalization*

$$x_3 = \begin{pmatrix} 28 & 76 & 55 & 116 & 73 \\ -25 & 22 & -67 & -50 & -19 \\ 8 & 34 & -34 & 20 & 13 \\ 28 & 13 & 23 & 30 & 8 \\ 30 & 52 & 42 & 4 & 4 \end{pmatrix}$$

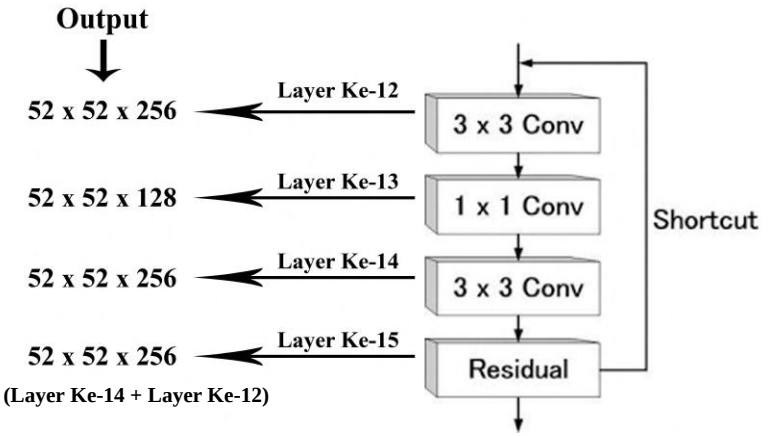
$$y_3 = LeakyReLU(x_3)$$

$$y_3 = \begin{pmatrix} 28 & 76 & 55 & 116 & 73 \\ -0.25 & 22 & -0.67 & -0.5 & -0.19 \\ 8 & 34 & -0.34 & 20 & 13 \\ 28 & 13 & 23 & 30 & 8 \\ 30 & 52 & 42 & 4 & 4 \end{pmatrix}$$

Hasil dari *Convolutional Layer* pada prinsipnya akan menjadi inputan pada proses *layer* setelahnya baik proses *convolutional layer*, *shortcut layer*, *route* maupun *upsample*.

2. **Shortcut Layer atau Residual**
   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*. Ilustrasi dari *shortcut layer* dapat dilihat pada Gambar 4.28.



Gambar 4.28: Proses *Shortcut Layer* pada Jaringan YoloV3

Misalkan pada *layer* ke-12 dapat dinyatakan sebagai peta fitur map berukuran $a^{[12]} = 3 \times 3 \times 2$ dan *layer* ke-14 dinyatakan sebagai $a^{[14]} = 3 \times 3 \times 2$, maka hasil *shortcut layer* pada layer ke-15 adalah $a^{[15]} = a^{[14]} + a^{[12]}$ dengan $a^{[12]}$ dan $a^{[14]}$ secara berturut - turut :

$$a^{[12]} = \begin{cases} \begin{pmatrix} 3 & 1 & 2 \\ 2 & 5 & 5 \\ 1 & 4 & 0 \end{pmatrix} \\ \begin{pmatrix} 4 & 1 & 0 \\ 4 & 3 & 1 \\ 3 & 1 & 5 \end{pmatrix} \end{cases} \qquad a^{[14]} = \begin{cases} \begin{pmatrix} 1 & 2 & 0 \\ 1 & 0 & 0 \\ 0 & 5 & 1 \end{pmatrix} \\ \begin{pmatrix} 2 & 1 & 1 \\ 0 & 1 & 0 \\ 5 & 0 & 3 \end{pmatrix} \end{cases}$$

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

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

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

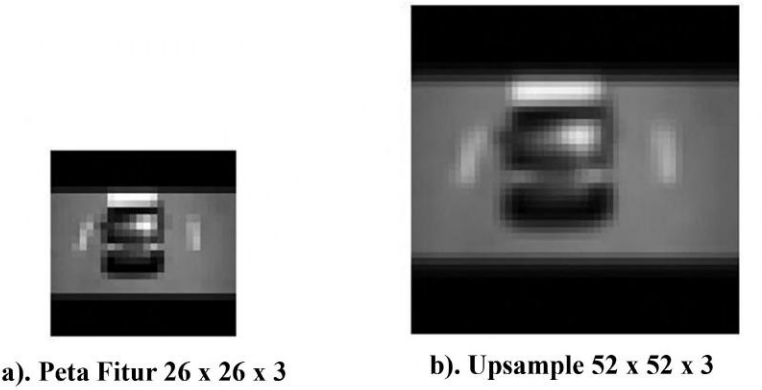
3. **Upsample**

   Pada jaringan YoloV3 proses *upsample* menggunakan input hasil peta fitur dari *layer* sebelumnya, teknik yang digunakan pada proses ini menggunakan *bilinear upsampling* dengan nilai faktor yaitu $2\times$. Jika hasil peta fitur *layer* sebelumnya berukuran $26 \times 26 \times 256$, maka pada proses *upsample* ukurannya menjadi $52 \times 52 \times 256$ (faktor $= 2$). Misalkan hasil peta fitur dari *layer* sebelumnya dinyatakan :

   $a^l = \begin{pmatrix} 2 & 4 & 3 \\ 3 & 2 & 4 \\ 1 & 3 & 4 \end{pmatrix}$

   $upsample(a^l) =$

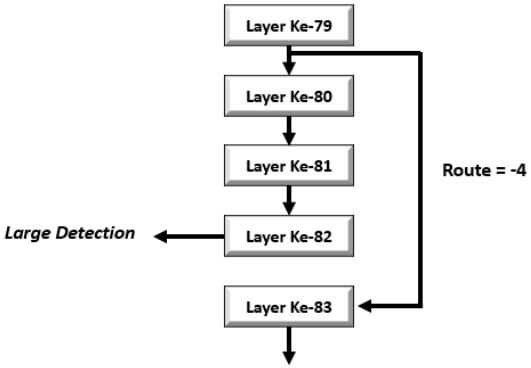
$$\begin{pmatrix} 2 & 2.5 & 3.5 & 3.75 & 3.25 & 3 \\ 2.25 & 2.5625 & 3.1875 & 3.4375 & 3.3125 & 3.25 \\ 2.75 & 2.6875 & 2.5625 & 2.8125 & 3.4375 & 3.75 \\ 2.5 & 2.4375 & 2.3125 & 2.6875 & 3.5625 & 4 \\ 1.5 & 1.8125 & 2.4375 & 3.0625 & 3.6875 & 4 \\ 1 & 1.5 & 2.5 & 3.25 & 3.75 & 4 \end{pmatrix}$$



Gambar 4.29: Ilustrasi Proses *Upsample Layer*

4. **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 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.



Gambar 4.30: Proses *Route Layer* dengan parameter $(route = -4)$

Misalkan peta fitur dari *layer* ke-79 dinyatakan sebagai :

$$a^{[79]} = \begin{pmatrix} 2 & 4 & 3 \\ 3 & 2 & 4 \\ 1 & 3 & 4 \end{pmatrix}$$

$$a^{[83]} = Route(-4)$$

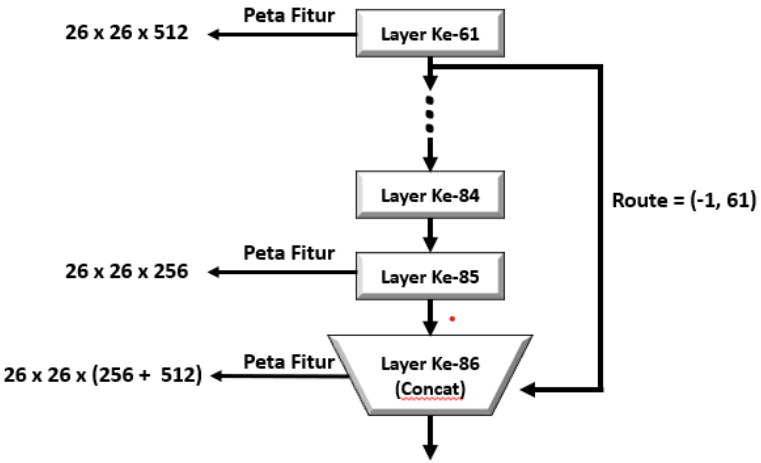
$$a^{[83]} = a^{[83-4]}$$

$$a^{[83]} = a^{[79]}$$

$$a^{[83]} = \begin{pmatrix} 2 & 4 & 3 \\ 3 & 2 & 4 \\ 1 & 3 & 4 \end{pmatrix}$$

- 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 ($route = (-1, 36)$ dan $route = (-1, 61)$), 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 fitur tersebut.



Gambar 4.31: Proses *Route Layer* dengan parameter $(route = (-1, 61))$

Misalkan peta fitur dari *layer* ke-61 dan ke-85 dinyatakan sebagai :

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

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

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

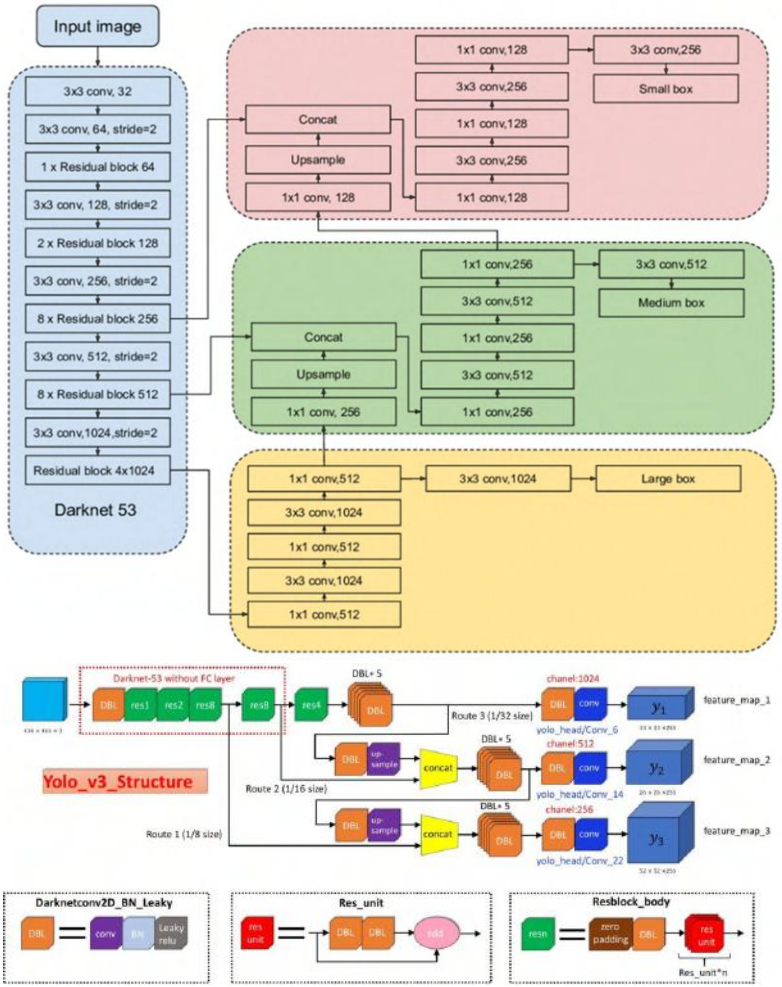
$$a^{[86]} = \begin{cases} \begin{pmatrix} 5 & 2 & 8 \\ 1 & 8 & 2 \\ 1 & 2 & 9 \end{pmatrix} \\ \begin{pmatrix} 3 & 6 & 8 \\ 1 & 4 & 5 \\ 2 & 3 & 5 \end{pmatrix} \\ \begin{pmatrix} 9 & 7 & 0 \\ 2 & 3 & 0 \\ 7 & 5 & 5 \end{pmatrix} \end{cases}$$

5. **Detections at Three Scales**
   Berdasarkan desain arsitektur YoloV3 pada Gambar 4.9 diketahui bahwa jaringan Yolov3 melakukan pendeteksian dengan memprediksi pada 3 skala yang berbeda, diantaranya :

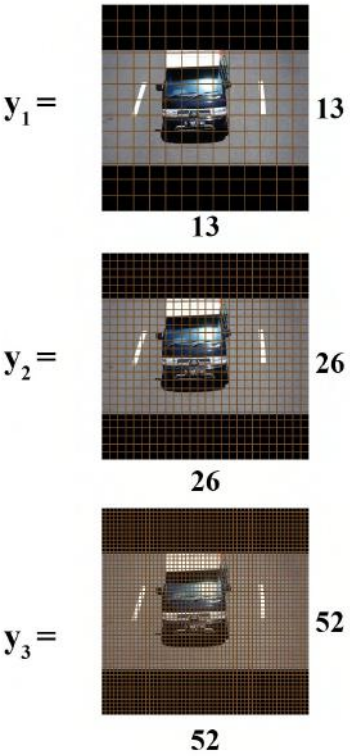
   - Skala pertama terjadi pada lapisan ke-82 dengan *stride* = 32. Karena ukuran dimensi *image* pada YoloV3 ini adalah 416 × 416, maka ukuran skala pada lapisan ini adalah $(\frac{416}{32} \times \frac{416}{32}) = (13 \times 13)$. Pada lapisan ini sering disebut juga sebagai lapisan *large detection.*
   - Skala kedua terjadi pada lapisan ke-94 dengan *stride* = 16. Karena ukuran dimensi *image* pada YoloV3 ini adalah 416 × 416, maka ukuran skala

pada lapisan ini adalah $(\frac{416}{16} \times \frac{416}{16}) = (26 \times 26)$. Pada lapisan ini sering disebut juga sebagai lapisan *medium detection*.

- Skala ketiga terjadi pada lapisan ke-106 dengan *stride* $= 8$. Karena ukuran dimensi *image* pada YoloV3 ini adalah 416×416, maka ukuran skala pada lapisan ini adalah $(\frac{416}{8} \times \frac{416}{8}) = (52 \times 52)$. Pada lapisan ini sering disebut juga sebagai lapisan *small detection*.



Gambar 4.32: Diagram Alir dan Struktur Jaringan YoloV3



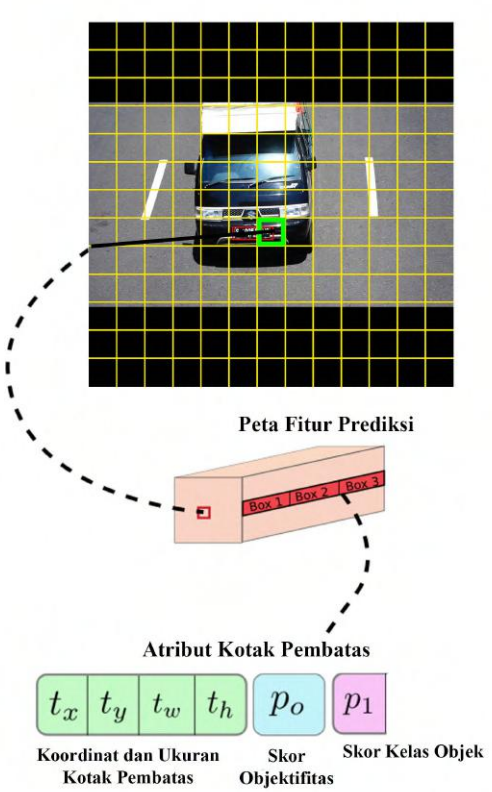
Gambar 4.33: Perbandingan Deteksi pada 3 Skala Jaringan YoloV3

Hal pertama yang perlu diketahui dalam mengintepretasikan output dari YoloV3 ini.

- input adalah kuwmpulan gambar sebanyak m dengan ukuran $(416, 416, 3)$
- Output adalah daftar kotak pembatas dengan kelas objek yang dikenali. Setiap kotak pembatas diwakili oleh 6 parameter $(p_c, b_x, b_y, b_h, b_w, c)$.

Parameter c merupakan kelas objek "Plat Nomor".

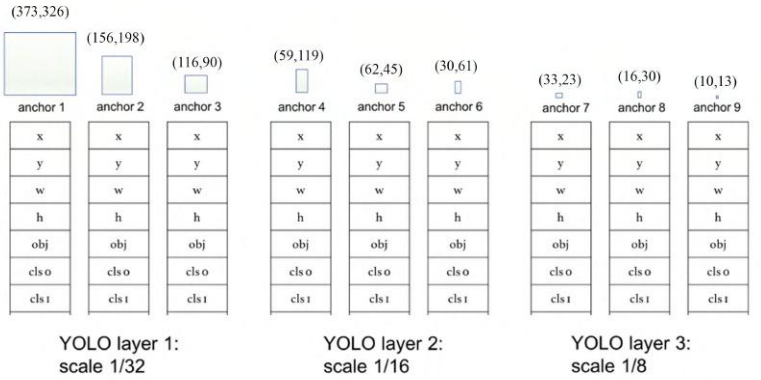
Dengan cara yang sama seperti pada semua fitur objek detektor yang telah dipelajari oleh *convolutional layer*, Yolo melakukan prediksi dengan menggunakan lapisan konvolusional (konvolusi $= 1 \times 1$). Sehingga hal pertama yang harus diperhatikan adalah output dari Yolo adalah peta fitur. Karena Yolo telah menggunakan konvolusi 1×1, maka ukuran peta prediksi persis dengan ukuran peta fitur sebelumnya. YoloV3 membagi gambar input ke dalam kisi dimensi yang sama dengan peta fitur akhir seperti Pada Gambar 4.33.



Gambar 4.34: Hasil Deteksi tiap Sel Grid

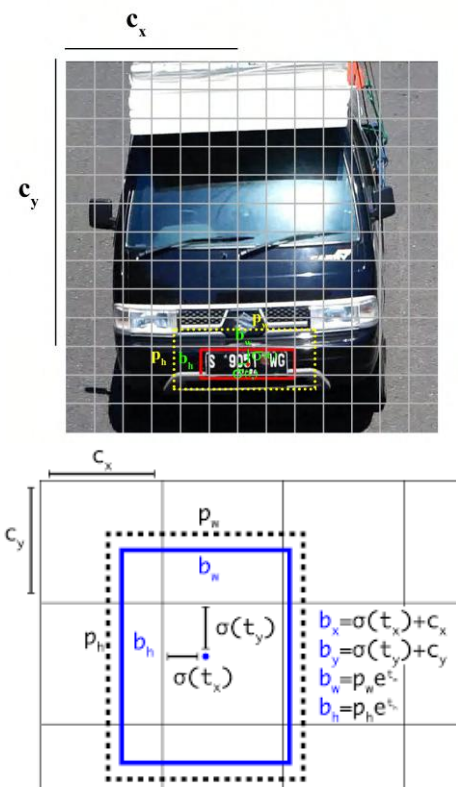
Pada Gambar 4.34 di atas sel yang berwarna hijau memiliki tiga kotak pembatas. Dimana sel tersebut akan bertugas dalam memprediksi kelas objek plat nomor. Pada YoloV3 setiap skala akan melakukan prediksi kelas objek menggunakan *anchor boxes* yang diperoleh dari *clustering* anotasi data. Dalam YoloV3, 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 YoloV3 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 4.35: *Anchor Boxes* Pada Jaringan YoloV3

Seperti yang disajikan pada Gambar. 4.34 pusat objek jatuh dalam sel tertentu dari grid 13 x 13. Sel itu adalah sel yang ditunjuk yang harus mendeteksi keberadaan objek. Selama pelatihan, 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* yang hadir di sel itu dalam melakukan deteksi objek.



Gambar 4.36: Proses Deteksi menggunakan *Anchor Boxes* Pada Jaringan YoloV3

Di sini b_x, b_y, b_h, b_w adalah koordinat pusat x, y, lebar dan tinggi prediksi Yolov3. t_x, t_y, t_w, t_h adalah hasil keluaran jaringan. c_x dan c_y adalah koordinat kiri atas dari kisi. p_w dan p_h adalah dimensi *anchor box*.

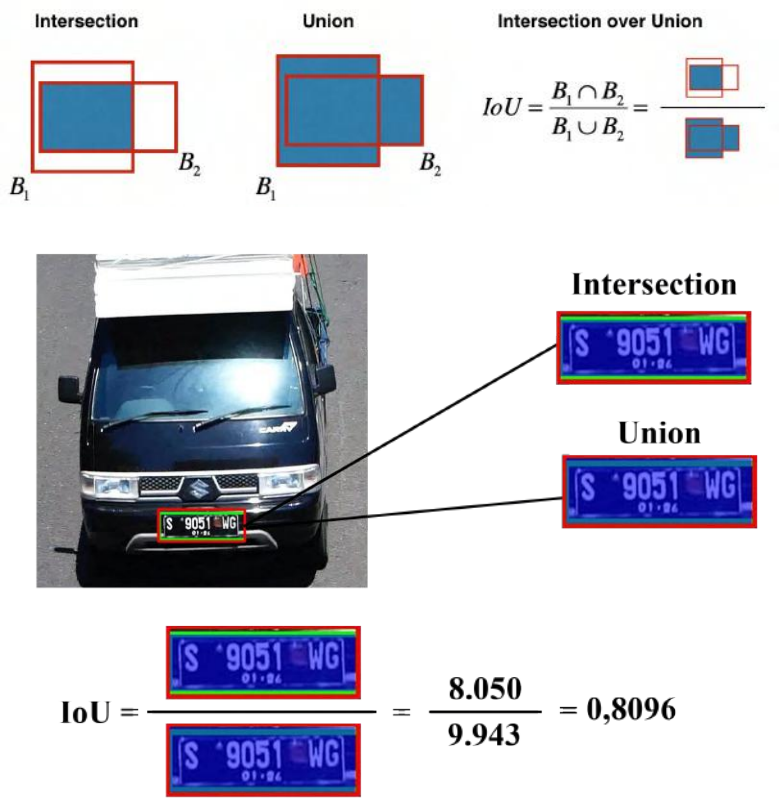
4.5.4 Output Processing (Pemfilteran dengan nilai ambang pada nilai kelas)

Pada gambar dengan ukuran 416 × 416, YOLOV3 memprediksi $((52 \times 52) + (26 \times 26) + 13 \times 13)) \times 3 = 10647$ kotak pembatas. Dalam menggunakan gambar sebelumnya, hanya ada satu objek yaitu plat mobil. Jadi, YoloV3 akan mengurangi deteksi dari 10647 menjadi 1.

Pertama, YoloV3 memfilter kotak berdasarkan skor objeknya. Kotak yang memiliki skor di bawah ambang batas (di bawah $0,5$) diabaikan. Selanjutnya, *Non-maksimum Suppression* (NMS) bermaksud untuk menyelesaikan masalah beberapa deteksi gambar 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 langkah-langkah berikut :

- Singkirkan kotak pembatas dengan skor rendah (artinya, kotak tidak terlalu *confidence* dalam mendeteksi kelas)
- Pilih hanya satu kotak pembatas ketika beberapa kotak saling tumpang tindih dan mendeteksi objek yang sama (*Non-max suppression*).

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. Setelah pemfilteran dengan menetapkan nilai skor kelas, YoloV3 masih memiliki banyak kotak pembatas yang tumpang tindih. Filter kedua untuk memilih kotak yang tepat disebut *NMS*. *NMS* menggunakan fungsi yang sangat penting yang disebut "*Intersection over Union*", atau IoU.



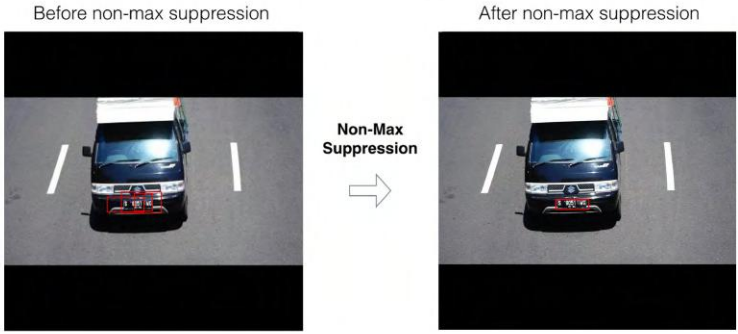
Gambar 4.37: Proses *Intersection over Union*

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

- Pilih kotak yang memiliki skor tertinggi.
- Hitung tumpang tindihnya dengan semua kotak pembatas yang lain, dan hapus kotak yang memiliki tumpang tindih lebih kecil dari *iou threshold*.

- Kembali ke langkah 1 dan ulangi sampai tidak ada lagi kotak dengan skor lebih rendah dari kotak yang dipilih saat ini.

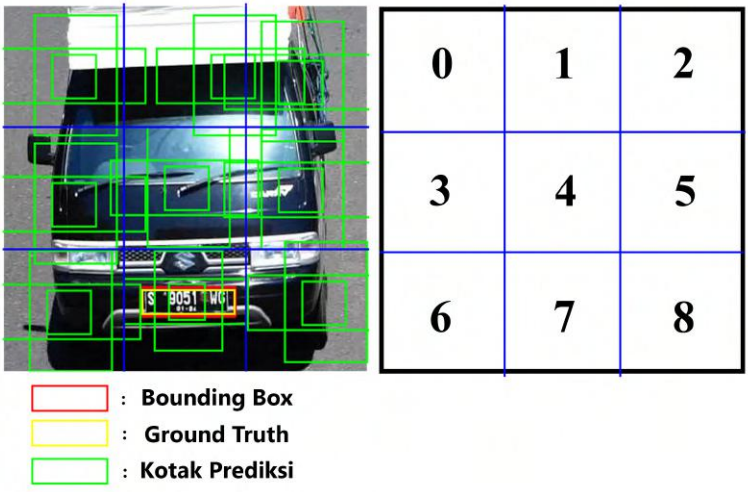
Langkah-langkah ini akan menghapus semua kotak yang memiliki tumpang tindih besar dengan kotak yang dipilih. Hanya kotak "terbaik" yang akan tersisa :



Gambar 4.38: Proses *Non-maksimum Suppression*

4.5.5 YoloV3 Loss Function

Pada tahap ini akan dicari nilai *loss function* dari model jaringan Yolov3 yang telah dibuat. Pada Gambar 4.39 direpresentasikan proses perhitungan *loss function* dari YoloV3 dengan dibagi menjadi 9 grid sel dan 3 *anchor boxes* :



Gambar 4.39: Proses *Loss Function* dari YoloV3

Pada Gambar 4.31 diperlihatkan bahwa hanya 1 grid sel yang memenuhi objek dari plat nomor mobil yaitu sel 7. Misalkan pada sel 7 memiliki parameter sebagai berikut :

- *Ground truth* $= (0, 0.449653, 0.604167, 0.111806, 0.047222)$ yang secara berturut-turut terdiri dari kelas objek, koordinat *center box* dan ukuran *box*.
- *Bounding box* $= (0.9, 0.447861, 0.605194, 0.108891, 0.0541309, 0.862)$ yang secara berturut-turut terdiri dari kelas objek, koordinat *center box*, ukuran *box* dan probabilitas kelas objek.
- Kotak prediksi pada sel $i = 7$:

  1. Kotak prediksi 1 $(B_1) =$ $(0.8, 0.447861, 0.605194, 0.072115, 0.146634, 0.9)$

yang secara berturut-turut terdiri dari nilai *confidence*, koordinat *center box*, ukuran *box* dan kelas objek.

2. Kotak prediksi 2 (B_2) = $(0.8, 0.447861, 0.605194, 0.149038, 0.108173, 0.9)$ yang secara berturut-turut terdiri dari nilai *confidence*, koordinat *center box*, ukuran *box* dan kelas objek.
3. Kotak prediksi 3 (B_3) = $(0.8, 0.447861, 0.605194, 0.141826, 0.286057, 0.9)$ yang secara berturut-turut terdiri dari nilai *confidence*, koordinat *center box*, ukuran *box* dan kelas objek.

Sehingga dari parameter yang diketahui dapat dihitung *loss function* dari yolov3 sebagai berikut :

1. ***Loss Function Localization***
   Karena nilai dari $1_{ij}^{obj} = 1$ ketika berada di sel $i = 7$ dan $1_{ij}^{obj} = 0$ ketika berada di sel selain $i = 7$ maka dengan menggunakan persamaan 2.4 didapatkan :

$$\lambda_{coord} \sum_{j=0}^{2} 1_{7j}^{obj} \left[(x_7 - \hat{x_7})^2 + (y_7 - \hat{y_7})^2 \right] +$$
$$\lambda_{coord} \sum_{j=0}^{2} 1_{7j}^{obj} \left[\left(\sqrt{w_7} - \sqrt{\hat{w_7}}\right)^2 + \left(\sqrt{h_7} - \sqrt{\hat{h_7}}\right)^2 \right] =$$

$5 \times 3 \times \left[(0.447861 - 0.449653)^2 + (0.605194 - 0.604167)^2 \right] +$
$5 \times 3 \times$
$\left[\left(\sqrt{0.108891} - \sqrt{0.111806}\right)^2 + \left(\sqrt{0.0541309} - \sqrt{0.047222}\right)^2 \right] =$
$0.000063989895 + 1.422370064303238 = 1.422434054198238$

2. ***Loss Function Confidence***
   Karena nilai dari $1_{ij}^{obj} = 1$ ketika berada di sel $i = 7$

serta kotak pembatas dan *ground truth* juga berada di sel $i = 7$, jika diasumsikan nilai $C_i = 0$ pada $i \neq 7$ maka dengan menggunakan persamaan 2.5 didapatkan :

$$\sum_{j=0}^{2} 1_{7j}^{obj} \left(C_7 - \hat{C}_7 \right)^2 = 3 \times (0.8 - 1)^2 = 0.04$$

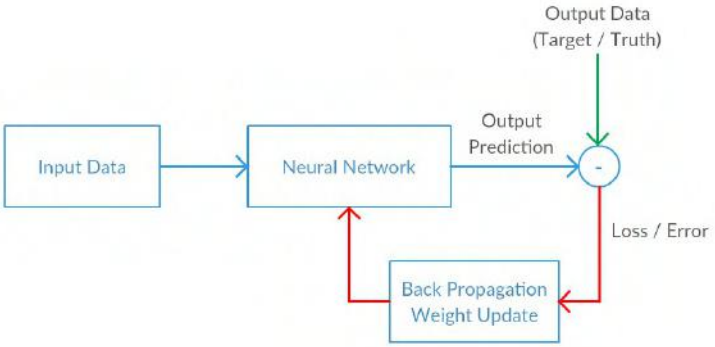
3. ***Loss Function Classification***
   Karena nilai dari $1_i^{obj} = 1$ ketika berada di sel $i = 7$ maka dengan menggunakan persamaan 2.6 didapatkan :

   $p_i(c) = 0.9 \times 0.862 = 0.7758$
   $\sum_{c=0}^{1} \left(p_i(c) - \hat{p}_i(c) \right)^2 = (0.7758 - 0.8096)^2 =$
   0.00114244

Sehingga Total *Loss Function* dari YoloV3 adalah $1.422434054198238 + 0.004 + 0.00114244 = 1.42757$

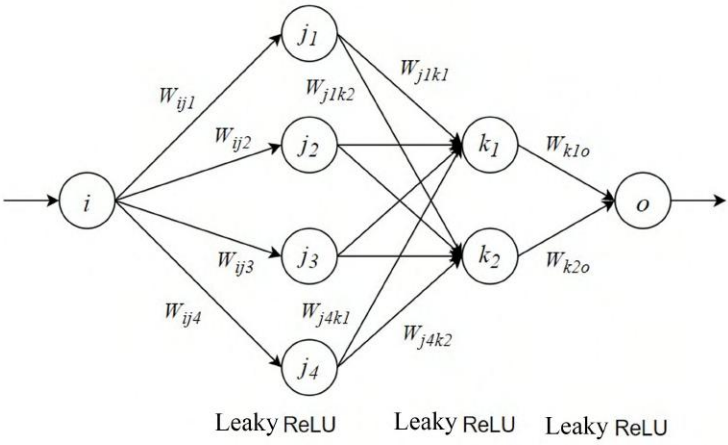
4.5.6 Back Propagation

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 untuk menyesuaikan kembali tiap *weights* dan *bias* berdasarkan *error* yang didapat pada saat *forward pass*.



Gambar 4.40: Neural Network Training

Misalkan jaringan YoloV3 dinyatakan sebagai jaringan seperti ini



$$input = [2.0]\,; output = [10.0]$$

$$W_{ij} = \begin{bmatrix} w_{ij_1} & w_{ij_2} & w_{ij_3} & w_{ij_4} \end{bmatrix} = \begin{bmatrix} 0.25 & 0.5 & 0.75 & 1.0 \end{bmatrix}$$

$$W_{jk} = \begin{bmatrix} w_{j_1k_1} & w_{j_1k_2} \\ w_{j_2k_1} & w_{j_2k_2} \\ w_{j_3k_1} & w_{j_3k_2} \\ w_{j_4k_1} & w_{j_4k_2} \end{bmatrix} = \begin{bmatrix} 1.0 & 0 \\ 0.75 & 0.25 \\ 0.5 & 0.5 \\ 0.25 & 0.75 \end{bmatrix}$$

$$W_{ko} = \begin{bmatrix} w_{k_1o} \\ w_{k_2o} \end{bmatrix} = \begin{bmatrix} 1.0 \\ 0.5 \end{bmatrix}$$

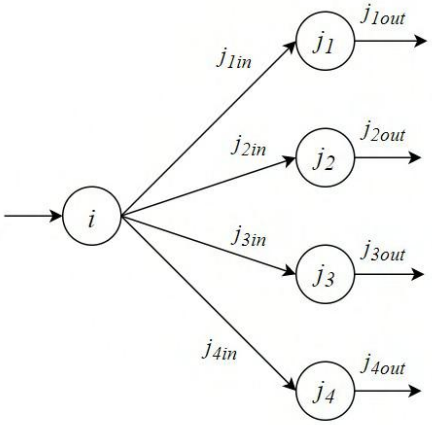
$$b_{ij} = \begin{bmatrix} b_{ij_1} & b_{ij_2} & b_{ij_3} & b_{ij_4} \end{bmatrix} = \begin{bmatrix} 1.0 & 1.0 & 1.0 & 1.0 \end{bmatrix}$$

$$b_{jk} = \begin{bmatrix} b_{jk_1} & b_{jk_2} \end{bmatrix} = \begin{bmatrix} 1.0 & 1.0 \end{bmatrix}$$

$$b_o = [1.0]$$

Neural network diatas terdiri dari 2 hidden layer. Hidden layer pertama dan kedua menggunakan *ReLU activation function*. Bias pada diagram diatas sebenarnya ada tetapi tidak digambarkan. Terdapat 4 weight dan 4 bias diantara input layer dan hidden layer pertama, 8 weight dan 2 bias diantara hidden layer pertama dan kedua, 2 weight dan 1 bias diantara hidden layer kedua dan output layer. Sehingga total ada 21 parameter yang harus diupdate.

Forward Pass (Input $->$ Hidden Layer 1)



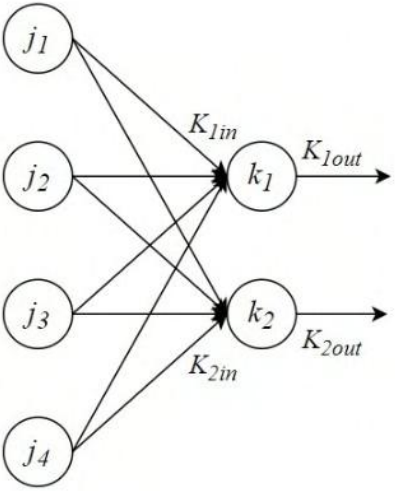
Pada proses ini akan dilakukan *forward pass* data input menuju hidden layer 1. Hal yang dilakukan adalah melakukan perkalian (*dot product*) dan penjumlahan matriks antara input, weights dan bias.

$$[j_{1in} \quad j_{2in} \quad j_{3in} \quad j_{4in}] = [input] \times [w_{ij_1} \quad w_{ij_2} \quad w_{ij_3} \quad w_{ij_4}] + [b_{ij_1} \quad b_{ij_2} \quad b_{ij_3} \quad b_{ij_4}]$$
$$[j_{1in} \quad j_{2in} \quad j_{3in} \quad j_{4in}] = [2.0] \times [0.25 \quad 0.5 \quad 0.75 \quad 1.0] + [1.0 \quad 1.0 \quad 1.0 \quad 1.0]$$
$$[j_{1in} \quad j_{2in} \quad j_{3in} \quad j_{4in}] = [2.0] \times [0.25 \quad 0.5 \quad 0.75 \quad 1.0] + [1.0 \quad 1.0 \quad 1.0 \quad 1.0]$$
$$[j_{1in} \quad j_{2in} \quad j_{3in} \quad j_{4in}] = [0.5 \quad 1 \quad 1.5 \quad 2] + [1.0 \quad 1.0 \quad 1.0 \quad 1.0]$$
$$[j_{1in} \quad j_{2in} \quad j_{3in} \quad j_{4in}] = [1.5 \quad 2.0 \quad 2.5 \quad 3.0]$$

Nilai diatas adalah input dari tiap node pada hidden layer 1. Semua nilai tersebut akan dikeluarkan setelah melalui *activation function*. Pada hidden layer 1 *activation function* yang digunakan adalah Leaky ReLU $=> f(x) = max(0.01x, x)$. Sehingga output dari hidden layer 1 adalah sebagai berikut :

$$LeakyReLU([\,j_{1in} \quad j_{2in} \quad j_{3in} \quad j_{4in}\,]) = [\,1.5 \quad 2.0 \quad 2.5 \quad 3.0\,]$$
$$[\,j_{1out} \quad j_{2out} \quad j_{3out} \quad j_{4out}\,] = [\,1.5 \quad 2.0 \quad 2.5 \quad 3.0\,]$$

Forward Pass (Hidden Layer 1 − > Hidden Layer 2)



Sama seperti *forward pass* pada layer sebelumnya, output dari tiap neuron pada ReLU layer akan mengalir ke semua neuron pada Leaky ReLU layer.

$$[\,k_{1in} \quad k_{2in}\,] = [\,j_{1out} \quad j_{2out} \quad j_{3out} \quad j_{4out}\,] \times \begin{bmatrix} w_{j_1k_1} & w_{j_1k_2} \\ w_{j_2k_1} & w_{j_2k_2} \\ w_{j_3k_1} & w_{j_3k_2} \\ w_{j_4k_1} & w_{j_4k_2} \end{bmatrix} + [\,b_{jk_1} \quad b_{jk_2}\,]$$

$$[\,k_{1in} \quad k_{2in}\,] = [\,1.5 \quad 2 \quad 2.5 \quad 3.0\,] \times \begin{bmatrix} 1.0 & 0 \\ 0.75 & 0.25 \\ 0.5 & 0.5 \\ 0.25 & 0.75 \end{bmatrix} + [\,1.0 \quad 1.0\,]$$

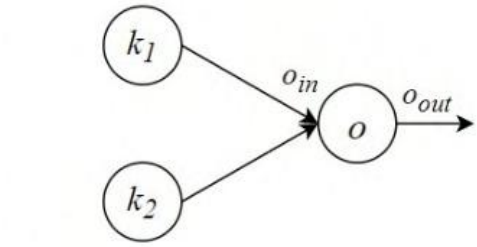
$$[\,k_{1in} \quad k_{2in}\,] = [\,5.0 \quad 4.0\,] + [\,1.0 \quad 1.0\,]$$

$$[\,k_{1in} \quad k_{2in}\,] = [\,6.0 \quad 5.0\,]$$

Setelah activation function :

$$LeakyReLU\left([k_{1in} \quad k_{2in}]\right) = [6.0 \quad 5.0]$$
$$[k_{1out} \quad k_{2out}] = [6.0 \quad 5.0]$$

Forward Pass (Hidden Layer 2 $->$ Output)



Sama seperti *forward pass* pada layer sebelumnya, output dari tiap neuron pada Lekay ReLU layer akan mengalir ke neuron pada Leaky ReLU layer (Output).

$$[o_{in}] = [k_{1out} \quad k_{2out}] \times \begin{bmatrix} w_{k1o} \\ w_{k2o} \end{bmatrix} + [b_o]$$
$$[o_{in}] = [6.0 \quad 5.0] \times \begin{bmatrix} 1.0 \\ 0.5 \end{bmatrix} + [1]$$
$$[o_{in}] = [9.5]$$

Setelah activation function :

$$LeakyReLU\left([o_{in}]\right) = [max(0.095, 9.5)]$$
$$[o_{out}] = [9.5]$$

Setelah mendapatkan nilai prediksi output, Selanjutnya akan dicari *loss* dengan menggunakan *squared error* (L2 Loss).

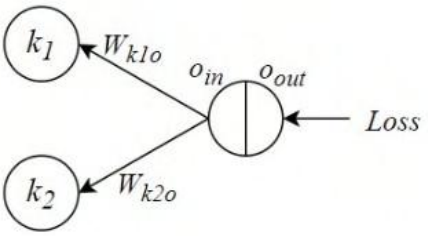
$$Loss = \frac{1}{2}(Prediction - Target)^2$$
$$Loss = \frac{1}{2}(o_{out} - output)^2$$
$$Loss = \frac{1}{2}(9.5 - 10.0)^2$$
$$Loss = \frac{1}{2}(-0.5)^2$$
$$Loss = \frac{1}{2}(0.25) = 0.125$$

Activation Function Derivatives $y = max(0.01x, x)$ atau

$$y = \begin{cases} x & x \geq 0 \\ 0.01x & x < 0 \end{cases}$$

$$\frac{\partial y}{\partial x} = \begin{cases} 1 & x \geq 0 \\ 0.01 & x < 0 \end{cases}$$

Backward Pass (Output $->$ Hidden Layer 2)



Hampir sama seperti pada forward pass, pada backward pass, loss akan mengalir menuju semua node pada hidden *layer* untuk dicari gradient nya. terhadap parameter yang akan diupdate. Misalkan diinginkan mengupdate parameter W_{k1o}, maka bisa digunakan prinsip *chain rule* atau aturan rantai seperti dibawah ini.

$$\frac{\partial Loss}{\partial w_{k_1 o}} = \frac{\partial Loss}{\partial o_{out}} \times \frac{\partial o_{out}}{\partial o_{in}} \times \frac{\partial o_{in}}{\partial w_{k_1 o}}$$

Pertama akan dicari berapa besar perubahan Loss berdasarkan output. Sehingga harus mencari turunan parsial (*partial derivative*) dari *loss function* terhadap output, hal ini juga bisa disebut sebagai *gradient loss function* terhadap output. Pada persamaan dibawah, *loss* akan dikalikan dengan $\frac{1}{2}$, tujuan fungsi diturunkan adalah agar fungsi *loss* akan menjadi 1 kali *Loss* (menetralisir turunan fungsi kuadrat).

$$Loss = \frac{1}{2}(output - o_{out})^2$$

$$\frac{\partial Loss}{\partial o_{out}} = \frac{\partial \left(\frac{1}{2}(output - o_{out})^2 \right)}{\partial o_{out}}$$

$$\frac{\partial Loss}{\partial o_{out}} = -1 \times 2 \times \frac{1}{2}(output - o_{out})$$

$$\frac{\partial Loss}{\partial o_{out}} = output - o_{out}$$

$$\frac{\partial Loss}{\partial o_{out}} = 9.5 - 10 = -0.5$$

Selanjutnya akan dicari *gradient* dari O_{out} terhadap O_{in}. Karena *activation function* yang digunakan adalah Leaky ReLU, maka :

$$o_{out} = max(0.01 o_{in}, o_{in})$$

$$o_{out} = max(0.095, 9.5)$$

$$\frac{\partial o_{out}}{\partial o_{in}} = \begin{cases} 1 & o_{in} \geq 0 \\ 0.01 & o_{in} < 0 \end{cases} \quad \frac{\partial o_{out}}{\partial o_{in}} = 1$$

Setelah itu akan dicari *gradient* dari O_{in} terhadap W_{k1o}, W_{k2o} dan bias (b_o). Perhatikan persamaan dibawah ini :

$$o_{in} = w_{k_1o}k_{1out} + w_{k_2o}k_{2out} + b_o$$

$$\frac{\partial o_{in}}{\partial w_{k_1o}} = \frac{\partial \left(w_{k_1o}k_{1out} + w_{k_2o}k_{2out} + b_o\right)}{\partial w_{k_1o}}$$

$$\begin{bmatrix} \frac{\partial o_{in}}{\partial w_{k_1o}} \\ \frac{\partial o_{in}}{\partial w_{k_2o}} \end{bmatrix} = \begin{bmatrix} k_{1out} \\ k_{2out} \end{bmatrix} = \begin{bmatrix} 6.0 \\ 5.0 \end{bmatrix}$$

$$\frac{\partial o_{in}}{\partial b_o} = [1]$$

Terakhir akan diterapkan *chain rule* untuk mencari *gradient loss* terhadap *weights* dan *bias*.

$$\begin{bmatrix} \frac{\partial Loss}{\partial w_{k_1o}} \\ \frac{\partial Loss}{\partial w_{k_2o}} \end{bmatrix} = \begin{bmatrix} \frac{\partial Loss}{\partial o_{out}} \times \frac{\partial o_{out}}{\partial o_{in}} \times \frac{\partial o_{in}}{\partial w_{k_1o}} \\ \frac{\partial Loss}{\partial o_{out}} \times \frac{\partial o_{out}}{\partial o_{in}} \times \frac{\partial o_{in}}{\partial w_{k_2o}} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial Loss}{\partial w_{k_1o}} \\ \frac{\partial Loss}{\partial w_{k_2o}} \end{bmatrix} = \begin{bmatrix} -0.5 \times 1 \times 6.0 \\ -0.5 \times 1 \times 5.0 \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial Loss}{\partial w_{k_1o}} \\ \frac{\partial Loss}{\partial w_{k_2o}} \end{bmatrix} = \begin{bmatrix} -3.0 \\ -2.5 \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial Loss}{\partial b_o} \end{bmatrix} = \begin{bmatrix} \frac{\partial Loss}{\partial o_{out}} \times \frac{\partial o_{out}}{\partial o_{in}} \times \frac{\partial o_{in}}{\partial b_o} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial Loss}{\partial b_o} \end{bmatrix} = [-0.5 \times 1 \times 1]$$

$$\begin{bmatrix} \frac{\partial Loss}{\partial b_o} \end{bmatrix} = -0.5$$

Stochastic Gradient Descent (SGD) Update

SGD adalah algoritma yang digunakan untuk meng*update* parameter dalam hal ini *weights* dan *bias*. Pada dasarnya langkah yang dilakukan hanya mengurangi *initial weights* dengan "sebagian" dari nilai gradient yang sudah didapat. Sebagian disini diwakili oleh *hyper-parameter* bernama *learning rate (alpha)*. Pada jaringan yolov3 nilai $alpha = 0.001$.

$$w'_{k_1o} = w_{k_1o} - \alpha \left(\frac{\partial Loss}{\partial w_{k_1o}} \right) = 1 - 0.001(-3.0) = 1.003$$

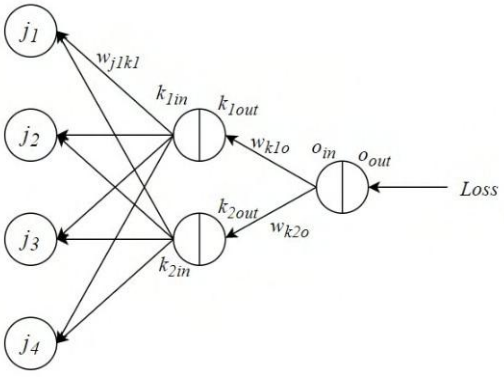
$$w'_{k_2o} = w_{k_2o} - \alpha \left(\frac{\partial Loss}{\partial w_{k_2o}} \right) = 0.5 - 0.001(-2.5) = 0.5025$$

$$b'_o = b_o - \alpha \left(\frac{\partial Loss}{\partial b_o} \right) = 1 - 0.001(-0.5) = 1.0005$$

$$W_{ko} = \begin{bmatrix} w_{k_1o} \\ w_{k_2o} \end{bmatrix} = \begin{bmatrix} 1.003 \\ 0.5025 \end{bmatrix}$$

$$b_o = [1.0005]$$

Backward Pass (Hidden Layer 2 − > Hidden Layer 1)



Pertama kita akan mencari *gradient loss* terhadap K_{1out}.

$$\frac{\partial Loss}{\partial k_{1out}} = \frac{\partial Loss}{\partial o_{out}} \times \frac{\partial o_{out}}{\partial o_{in}} \times \frac{\partial o_{in}}{\partial w_{k_1 o}} \times \frac{\partial w_{k_1 o}}{\partial k_{1out}}$$

$$\frac{\partial Loss}{\partial k_{1out}} = -0.5 \times 1 \times 6 \times w_{k_1 o(Lama)}$$

$$\frac{\partial Loss}{\partial k_{1out}} = -0.5 \times 1 \times 6 \times 1$$

$$\begin{bmatrix} \frac{\partial Loss}{\partial k_{1out}} & \frac{\partial Loss}{\partial k_{2out}} \end{bmatrix} = [-3 \quad -1.25]$$

Karena fungsi aktivasi yang digunakan adalah Leaky ReLU maka hasil turunananya adalah :

$$\begin{bmatrix} \frac{\partial k_{1out}}{\partial k_{1in}} \\ \frac{\partial k_{2out}}{\partial k_{2in}} \end{bmatrix} = \begin{bmatrix} 0.001 \\ 0.001 \end{bmatrix}$$

Selanjutnya kita akan cari *gradient* K_{1in} terhadap $W_{j_1 k_1}$.

$$k_{1in} = w_{j_1 k_1} j_{1out} + w_{j_2 k_1} j_{2out} + w_{j_3 k_1} j_{3out} + w_{j_4 k_1} j_{4out}$$

$$\frac{\partial k_{1in}}{\partial w_{j_1 k_1}} = \frac{\partial \left(w_{j_1 k_1} j_{1out} + w_{j_2 k_1} j_{2out} + w_{j_3 k_1} j_{3out} + w_{j_4 k_1} j_{4out} \right)}{\partial w_{j_1 k_1}}$$

$$\begin{bmatrix} \frac{\partial k_{1in}}{\partial w_{j_1 k_1}} & \frac{\partial k_{1in}}{\partial w_{j_2 k_1}} & \frac{\partial k_{1in}}{\partial w_{j_3 k_1}} & \frac{\partial k_{1in}}{\partial w_{j_4 k_1}} \\ \frac{\partial k_{2in}}{\partial w_{j_1 k_2}} & \frac{\partial k_{2in}}{\partial w_{j_2 k_2}} & \frac{\partial k_{2in}}{\partial w_{j_3 k_2}} & \frac{\partial k_{2in}}{\partial w_{j_4 k_2}} \end{bmatrix} = \begin{bmatrix} j_{1out} & j_{2out} & j_{3out} & j_{4out} \\ j_{1out} & j_{2out} & j_{3out} & j_{4out} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial k_{1in}}{\partial w_{j_1 k_1}} & \frac{\partial k_{1in}}{\partial w_{j_2 k_1}} & \frac{\partial k_{1in}}{\partial w_{j_3 k_1}} & \frac{\partial k_{1in}}{\partial w_{j_4 k_1}} \\ \frac{\partial k_{2in}}{\partial w_{j_1 k_2}} & \frac{\partial k_{2in}}{\partial w_{j_2 k_2}} & \frac{\partial k_{2in}}{\partial w_{j_3 k_2}} & \frac{\partial k_{2in}}{\partial w_{j_4 k_2}} \end{bmatrix} =$$

$$\begin{bmatrix} 1.5 & 2.0 & 2.5 & 3.0 \\ 1.5 & 2.0 & 2.5 & 3.0 \end{bmatrix}$$

Kemudian menghitung *gradient loss* terhadap $W_{j_1k_1}$ dengan menerapkan *chain rule* yang tadi.

$$\frac{\partial Loss}{\partial w_{j_1k_1}} = \frac{\partial Loss}{\partial k_{1out}} \times \frac{\partial k_{1out}}{\partial k_{1in}} \times \frac{\partial k_{1in}}{\partial w_{j_1k_1}}$$

$$\frac{\partial Loss}{\partial w_{j_1k_1}} = -3 \times 0.001 \times 1.5$$

$$\begin{bmatrix} \frac{\partial Loss}{\partial w_{j_1k_1}} & \frac{\partial Loss}{\partial w_{j_1k_2}} \\ \frac{\partial Loss}{\partial w_{j_2k_1}} & \frac{\partial Loss}{\partial w_{j_2k_2}} \\ \frac{\partial Loss}{\partial w_{j_3k_1}} & \frac{\partial Loss}{\partial w_{j_3k_2}} \\ \frac{\partial Loss}{\partial w_{j_4k_1}} & \frac{\partial Loss}{\partial w_{j_4k_2}} \end{bmatrix} = \begin{bmatrix} -0.0045 & -0.001875 \\ -0.006 & -0.0025 \\ -0.0075 & -0.003125 \\ -0.009 & -0.00375 \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial Loss}{\partial b_{jk_1}} & \frac{\partial Loss}{\partial b_{jk_2}} \end{bmatrix} = [-0.003 \quad -0.00125]$$

SGD Update (Hidden Layer 2 $->$ Hidden Layer 1)

$$\begin{bmatrix} w'_{j_1k_1} & w'_{j_1k_2} \\ w_{j_2k_1} & w_{j_2k_2} \\ w_{j_3k_1} & w_{j_3k_2} \\ w_{j_4k_1} & w_{j_4k_2} \end{bmatrix} =$$

$$\begin{bmatrix} w_{j_1k_1} - \alpha\left(\frac{\partial Loss}{\partial w_{j_1k_1}}\right) & w_{j_1k_2} - \alpha\left(\frac{\partial Loss}{\partial w_{j_1k_2}}\right) \\ w_{j_2k_1} - \alpha\left(\frac{\partial Loss}{\partial w_{j_2k_1}}\right) & w_{j_2k_2} - \alpha\left(\frac{\partial Loss}{\partial w_{j_2k_2}}\right) \\ w_{j_3k_1} - \alpha\left(\frac{\partial Loss}{\partial w_{j_3k_1}}\right) & w_{j_3k_2} - \alpha\left(\frac{\partial Loss}{\partial w_{j_3k_2}}\right) \\ w_{j_4k_1} - \alpha\left(\frac{\partial Loss}{\partial w_{j_4k_1}}\right) & w_{j_4k_2} - \alpha\left(\frac{\partial Loss}{\partial w_{j_4k_2}}\right) \end{bmatrix}$$

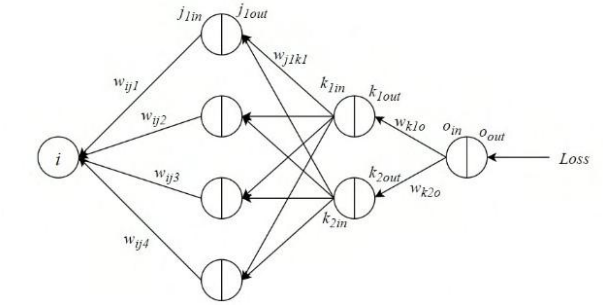
$$\begin{bmatrix} w'_{j_1k_1} & w'_{j_1k_2} \\ w'_{j_2k_1} & w'_{j_2k_2} \\ w'_{j_3k_1} & w'_{j_3k_2} \\ w'_{j_4k_1} & w'_{j_4k_2} \end{bmatrix} = \begin{bmatrix} 1 & 1.875 \\ 0.75 & 0.25 \\ 0.5 & 0.5 \\ 0.25 & 0.75 \end{bmatrix}$$

$$\begin{bmatrix} b'_{jk_1} & b'_{jk_2} \end{bmatrix} =$$

$$\begin{bmatrix} b_{jk_1} - \alpha\left(\frac{\partial Loss}{\partial b_{jk_1}}\right) & b_{jk_2} - \alpha\left(\frac{\partial Loss}{\partial b_{jk_2}}\right) \end{bmatrix}$$

$$\begin{bmatrix} b'_{jk_1} & b'_{jk_2} \end{bmatrix} = [1.000003 \quad 1.00000125]$$

Backward Pass (Hidden Layer 1 − > Input Layer)



$$\frac{\partial Loss}{\partial k_{out}} = \frac{\partial Loss}{\partial k_{1out}} + \frac{\partial Loss}{\partial k_{2out}} = -4.25$$

$$\frac{\partial k_{out}}{\partial k_{in}} = \frac{\partial k_{1out}}{\partial k_{in}} + \frac{\partial k_{2out}}{\partial k_{in}} = 0.002$$

$$\frac{\partial k_{in}}{\partial w_{j_1 k}} = \frac{\partial k_{1in}}{\partial w_{j_1 k_1}} + \frac{\partial k_{2in}}{\partial w_{j_1 k_2}} = 3$$
$$\frac{\partial w_{j_1 k}}{\partial w_{j1out}} = \frac{\partial w_{j_1 k_1}}{\partial w_{j1out}} + \frac{\partial w_{j_1 k_2}}{\partial w_{j1out}} = 1$$
$$\frac{\partial Loss}{\partial j_{1out}} = \frac{\partial Loss}{\partial k_{out}} \times \frac{\partial k_{out}}{\partial k_{in}} \times \frac{\partial k_{in}}{\partial w_{j_1 k}} \times \frac{\partial w_{j_1 k}}{\partial w_{j_{1out}}}$$
$$\frac{\partial Loss}{\partial j_{1out}} = -4.25 \times 0.002 \times 3 \times 1$$
$$\frac{\partial Loss}{\partial j_{1out}} = -0.0255$$

Lanjut dengan gradient J_{1out} terhadap J_{1in}.

$$j_{1out} = max(0.001 j_{1in}, j1in)$$
$$j_{1out} = max(0.0015, 0.001)$$
$$\frac{\partial j_{1out}}{\partial j_{1in}} = \begin{cases} 1 & j_{1in} \geq 0 \\ 0.001 & j_{1in} < 0 \end{cases} \frac{\partial j_{1out}}{\partial j_{1in}} = 1$$

Selanjutnya akan dicari gradient J_{1in} terhadap W_{ij1}.

$$j_{1in} = w_{ij_1} i + b_{ij_1}$$
$$\frac{\partial j_{1in}}{\partial w_{ij_1}} = \frac{\partial \left(w_{ij_1} i + b_{ij_1} \right)}{\partial w_{ij_1}}$$
$$\frac{\partial j_{1in}}{\partial w_{ij_1}} = i$$
$$\frac{\partial j_{1in}}{\partial w_{ij_1}} = 2.0$$

Pada akhirnya dapat dihitung *gradient loss* terhadap W_{ij_1} dengan menerapkan *chain rule*.

$$\frac{\partial Loss}{\partial w_{ij_1}} = \frac{\partial Loss}{\partial j_{1out}} \times \frac{\partial j_{1out}}{\partial j_{1in}} \times \frac{\partial j_{1in}}{\partial w_{ij_1}}$$
$$\frac{\partial Loss}{\partial w_{ij_1}} = -0.0255 \times 1 \times 2 = -0.051$$

Perhitungan yang baru saja dilakukan tadi akan diterapkan untuk semua parameter. Maka didapat semua *gradient* yang dibutuhkan untuk melakukan *update*.

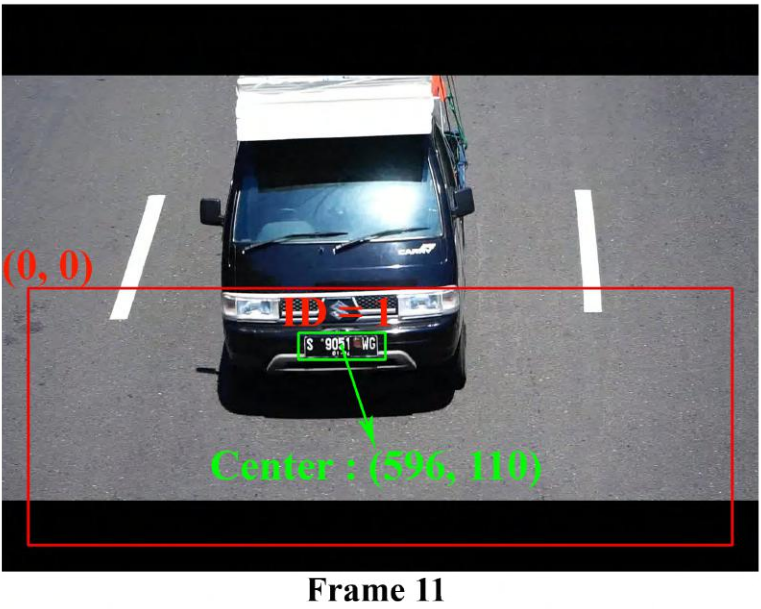
$$\begin{bmatrix} \frac{\partial Loss}{\partial w_{ij_1}} & \frac{\partial Loss}{\partial w_{ij_2}} & \frac{\partial Loss}{\partial w_{ij_3}} & \frac{\partial Loss}{\partial w_{ij_4}} \end{bmatrix} =$$
$$[-0.051 \quad -0.05528 \quad -0.0691 \quad -0.08292]$$
$$\begin{bmatrix} \frac{\partial Loss}{\partial b_{ij_1}} & \frac{\partial Loss}{\partial b_{ij_2}} & \frac{\partial Loss}{\partial b_{ij_3}} & \frac{\partial Loss}{\partial b_{ij_4}} \end{bmatrix} =$$
$$[-0.02073 \quad -0.02764 \quad -0.03455 \quad -0.04146]$$

SGD Update (Hidden Layer 1 $->$ Input Layer)

$$\begin{bmatrix} w'_{ij_1} & w'_{ij_2} & w'_{ij_3} & w'_{ij_4} \end{bmatrix} =$$
$$[0,250051 \quad 0.51382 \quad 0.76728 \quad 1.02073]$$
$$\begin{bmatrix} b'_{ij_1} & b'_{ij_2} & b'_{ij_3} & b'_{ij_4} \end{bmatrix} =$$
$$[1.02073 \quad 1.02764 \quad 1.03455 \quad 1.02073]$$

4.5.7 *Tracking* Objek Plat Nomor

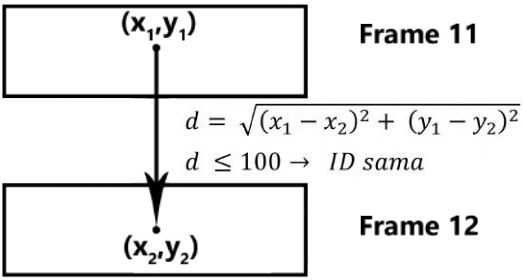
Pada proses ini setelah didapatkan prediksi *bounding box* pada objek plat nomor, selanjutnya dilakukan proses *tracking* objek. Tujuan dilakukan *tracking* ini agar setiap objek plat nomor yang diprediksi dapat memiliki ID sehingga dapat diketahui jumlah objek plat yang dideteksi pada sebuah video. Ilustrasi proses *tracking* objek plat nomor kendaraan ditunjukkan seperti di bawah ini.



Gambar 4.41: Tracking Objek Plat pada Deteksi Pertama

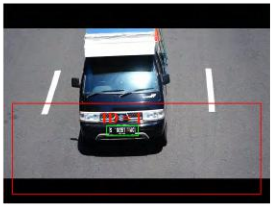
Pada Gambar 4.41 terlihat bahwa deteksi objek pertama kali terjadi pada *frame* ke-11, sehingga ketika dilakukan *tracking* maka objek plat akan diberi $ID = 1$. Selanjutnya akan disimpan nilai koordinat *center* dari *bounding box* pada *frame* ke-11. Koordinat *center* ini akan diperiksa pada prediksi *bounding box* di *frame* ke-12 untuk mengetahui apakah titik *center* juga berada di *bounding box* di *frame* ke-12. Jika titik tersebut berada di dalam *bounding box* di *frame* ke-12 maka ID objek plat nomor pada *frame* ke-12 adalah $ID = 1$. Jika titik tersebut tidak berada di *bounding box* di *frame* ke-12 maka ID objek plat nomor pada *frame* ke-12 adalah $ID = 2$. Namun, dalam hal ini untuk menentukan

objek memiliki ID yang sama digunakan rumus jarak euclid yang dapat dilihat pada Gambar 4.42.

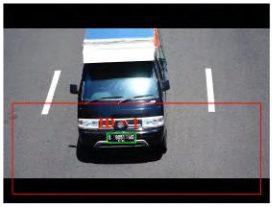


Gambar 4.42: Proses Rumus Jarak Euclid

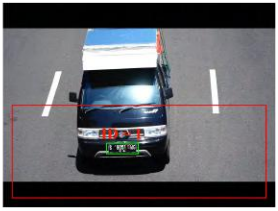
Pada Gambar 4.43 hasil deteksi koordinat titik pusat akan diperiksa pada *bounding box frame* selanjutnya, apakah titik tersebut berada di dalam *bounding box* atau tidak. Pada Gambar 4.43 diperlihatkan untuk warna yang sama yaitu warna kuning dan biru, akan dilakukan pemeriksaan titik pusat pada *bounding box* di *frame* selanjutnya.



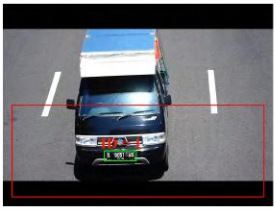
Frame 11



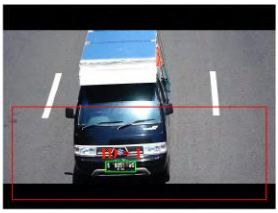
Frame 12



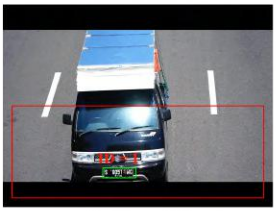
Frame 13



Frame 14



Frame 15



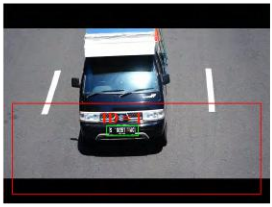
Frame 16

Frame Ke-	ID	Koordinat *Bounding Box*		Ukuran *Bounding Box*	
		Titik Pusat	Titik Kiri Atas	Panjang	Lebar
11	1	{596, 110}	{514, 85}	165	51
12	1	{592, 136}	{508, 109}	169	54
13	1	{587, 161}	{509, 135}	157	52
14	1	{581, 187}	{497, 159}	168	56
15	1	{576, 213}	{491, 186}	170	54
16	1	{571, 234}	{491, 212}	160	55

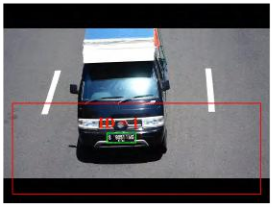
Gambar 4.43: Proses Tracking Objek Plat Nomor

4.5.8 Menentukan Lokalisasi Plat

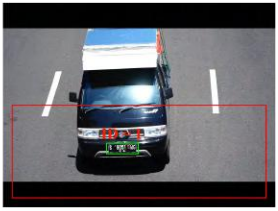
Setelah berhasil mendeteksi plat nomor dan *tracking* objeknya, proses selanjutnya adalah menentukan lokalisasi dari plat nomor yang terdeteksi dari beberapa *frame*. Dalam menentukan lokasi plat nomor dari beberapa *frame*, proses ini akan memilih nilai *confidence* tertinggi dari *frame* - *frame* yang berhasil terdeteksi dari objek masuk sampai keluar ROI. Perlu diingat bahwa nilai *confidence* didapatkan dari proses deteksi menggunakan YoloV3. Adapun proses dalam menentukan lokalisasi plat nomor dapat dilihat pada Gambar 4.44.



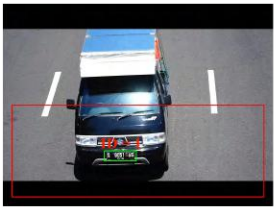
Frame 11



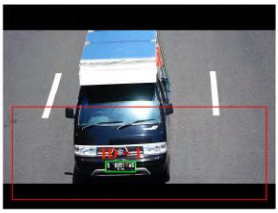
Frame 12



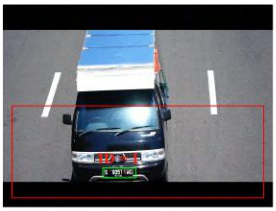
Frame 13



Frame 14



Frame 15



Frame 16

Frame Ke-	ID	Parameter *Bounding Box*			Confidence
		Titik Pusat	Panjang	Lebar	
11	1	{596, 110}	165	51	0.9779
12	1	{592, 136}	169	54	0.9839
13	1	{587, 161}	157	52	0.9614
14	1	{581, 187}	168	56	0.9925
15	1	{576, 213}	170	54	0.9956
16	1	{571, 234}	160	55	0.9885

Gambar 4.44: Proses Lokalisasi Objek Plat Nomor

Pada Gambar 4.44 merupakan proses melakukan penentuan lokalisasi plat nomor kendaraan. Pada gambar

tersebut dapat disimpulkan bahwa dalam menentukan lokasi plat nomor kendaraan bergerak didasarkan pada nilai *confidence* tertinggi dari *frame-frame* yang berhasil terdeteksi pada ROI. Proses ini dilakukan diawali ketika objek pertama kali terdeteksi di ROI sampai objek keluar dari ROI. Hasil tersebut dapat dilihat pada baris berwarna hijau dengan dinyatakan nilai *confidence* tertinggi didapatkan pada *frame* ke-15 dengan nilai *confidence* = 0.9956. Selanjutnya parameter *bounding box* yang ada di *frame* ke-15 akan digunakan sebagai lokalisasi objek plat nomor dengan ID = 1 dan dilakukan proses *cropping* berdasarkan parameter tersebut. Hasil *cropping* berdasarkan parameter *bounding box* di *frame* ke-15 dapat dilihat pada Gambar 4.45.

Gambar 4.45: Hasil Cropping Plat Nomor

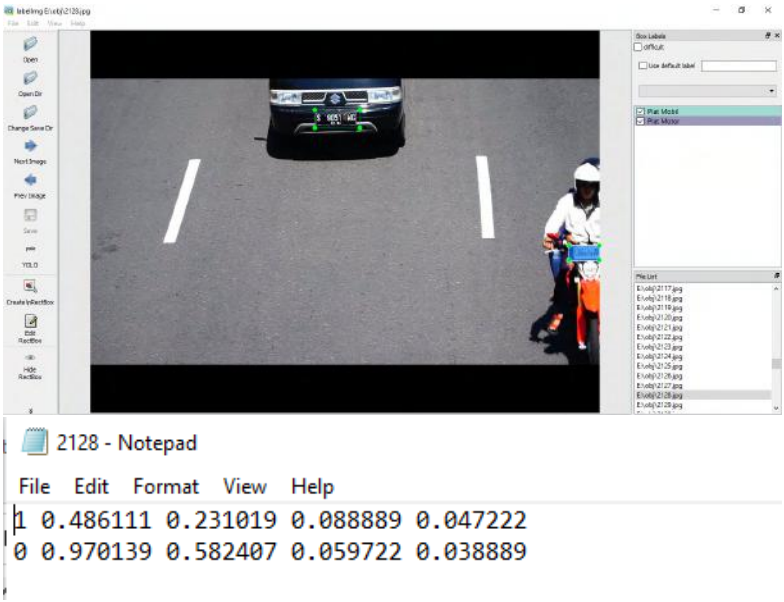
4.6 Implementasi Sistem

Sistem diimplementasikan dengan menggunakan pemrograman Java dan Python menggunakana *library* openCV 4.0.0.

4.6.1 Implementasi Tahap Training

Implementasi tahap training mengacu pada desain arsitektur YoloV3. Tahap training bertujuan untuk memperoleh *trained weights*. Pada tahap ini sistem diimplementasikan dengan menggunakan pemrograman Python.

1. **Anotasi Data**
   Pembuatan data anotasi dilakukan secara manual menggunakan aplikasi labelimg v1.7.0. Data input berupa data citra kendaraan, kemudian diberi *box* dan label jenis plat nomor kendaraan. Hasil pelabelan tersebut, kemudian disimpan sebagai data anotasi. Isi data tersebut akan disimpan pada *file .txt* dengan nilai $c, (x, y), (w, h)$ berturut-turut adalah kelas objek, koordinat titik pusat kotak pembatas, dan ukuran dimensi dari kotak pembatas



Gambar 4.46: Anotasi Data Menggunakan labelimg v1.7.0

2. **Membuat Model Jaringan Layer YoloV3**
   Proses ini dilakukan bertujuan untuk membuat model

jaringan YoloV3 seperti *convolutional layers, shortcut layers, upsample, activation* dan *route*. Berikut source code dalam membuat model jaringan YoloV3 :

```
def DarknetConv(x, filters, size,
    strides=1, batch_norm=True):
   if strides == 1:
      padding = 'same'
   else:
      x = ZeroPadding2D(((1, 0), (1,
          0)))(x)
      padding = 'valid'
   x = Conv2D(filters=filters,
       kernel_size=size,
      strides=strides, padding=padding,
      use_bias=not batch_norm,
          kernel_regularizer=l2(0.0005))(x)
   if batch_norm:
      x = BatchNormalization()(x)
      x = LeakyReLU(alpha=0.1)(x)
   return x

def DarknetResidual(x, filters):
   previous = x
   x = DarknetConv(x, filters // 2, 1)
   x = DarknetConv(x, filters, 3)
   x = Add()([previous , x])
   return x

def DarknetBlock(x, filters, blocks):
   x = DarknetConv(x, filters, 3,
       strides=2)
   for _ in repeat(None, blocks):
      x = DarknetResidual(x, filters)
   return x

def Darknet(name=None):
```

```
    x = inputs = Input([None, None, 3])
    x = DarknetConv(x, 32, 3)
    x = DarknetBlock(x, 64, 1)
    x = DarknetBlock(x, 128, 2)
    x = x_36 = DarknetBlock(x, 256, 8)
    x = x_61 = DarknetBlock(x, 512, 8)
    x = DarknetBlock(x, 1024, 4)
    return tf.keras.Model(inputs, (x_36,
        x_61, x), name=name)

def YoloConv(filters, name=None):
    def yolo_conv(x_in):
        if isinstance(x_in, tuple):
            inputs =
                Input(x_in[0].shape[1:]),
                Input(x_in[1].shape[1:])
            x, x_skip = inputs

            x = DarknetConv(x, filters, 1)
            x = UpSampling2D(2)(x)
            x = Concatenate()([x, x_skip])
        else:
            x = inputs = Input(x_in.shape[1:])

            x = DarknetConv(x, filters, 1)
            x = DarknetConv(x, filters * 2, 3)
            x = DarknetConv(x, filters, 1)
            x = DarknetConv(x, filters * 2, 3)
            x = DarknetConv(x, filters, 1)
        return Model(inputs, x,
            name=name)(x_in)
    return yolo_conv

def YoloOutput(filters, anchors,
    classes, name=None):
    def yolo_output(x_in):
        x = inputs = Input(x_in.shape[1:])
        x = DarknetConv(x, filters * 2, 3)
```

```
            x = DarknetConv(x, anchors *
                (classes + 5), 1,
                batch_norm=False)
            x = Lambda(lambda x: tf.reshape(x,
                (-1, tf.shape(x)[1],
                tf.shape(x)[2],
            anchors, classes + 5)))(x)
            return tf.keras.Model(inputs, x,
                name=name)(x_in)
        return yolo_output

        def yolo_boxes(pred, anchors, classes):
            grid_size = tf.shape(pred)[1]
            box_xy, box_wh, score, class_probs =
                tf.split(pred, (2, 2, 1, classes),
                axis=-1)

            box_xy = tf.sigmoid(box_xy)
            score = tf.sigmoid(score)
            class_probs = tf.sigmoid(class_probs)
            pred_box = tf.concat((box_xy,
                box_wh), axis=-1)

            grid =
                tf.meshgrid(tf.range(grid_size),
                tf.range(grid_size))
            grid = tf.expand_dims(tf.stack(grid,
                axis=-1), axis=2)

            box_xy = (box_xy + tf.cast(grid,
                tf.float32)) / tf.cast(grid_size,
                tf.float32)
            box_wh = tf.exp(box_wh) * anchors

            box_x1y1 = box_xy - box_wh / 2
            box_x2y2 = box_xy + box_wh / 2
            bbox = tf.concat([box_x1y1,
                box_x2y2], axis=-1)
```

```
            return bbox, score, class_probs,
                pred_box
```

3. **Batch Normalization**
   Proses selanjutnya adalah mengimplementasikan *Batch Normalization* yang dinyatakan dengan *source code* seperti di bawah ini :

```
class BatchNormalization
    (tf.keras.layers.BatchNormalization):
   def call(self, x, training=False):
      if training is None: training =
          tf.constant(False)
      training = tf.logical_and(training,
          self.trainable)
      return super().call(x, training)
```

4. **Membuat *Anchor Boxes***
   Proses ini dilakukan dengan *clustering bounding box* dari hasil anotasi data. Teknik yang digunakan adalah *K-Means Clustering*, adapun *source code* yang diberikan adalah :

```
def kmeans(boxes, k, dist=np.median):
   rows = boxes.shape[0]

   distances = np.empty((rows, k))
   last_clusters = np.zeros((rows,))

   np.random.seed()

   clusters =
       boxes[np.random.choice(rows, k,
       replace=False)]
```

```
        while True:
            for row in range(rows):
                distances[row] = 1 -
                    iou(boxes[row], clusters)

            nearest_clusters =
                np.argmin(distances, axis=1)

            if (last_clusters ==
                nearest_clusters).all():
                break

            for cluster in range(k):
                clusters[cluster] =
                    dist(boxes[nearest_clusters
                    == cluster], axis=0)

            last_clusters = nearest_clusters

        return clusters
```

5. ***Intersection Over Union* (IoU)**
   Proses selanjutnya adalah membuat *function* IoU yang bertujuan untuk melihat kinerja akurasi prediksi terhadap *ground truth*. Berikut *source code* dari IoU :

```
    def intersectionOverUnion(box1, box2):
        intersect_w =
            interval_overlap([box1.xmin,
            box1.xmax], [box2.xmin, box2.xmax])
        intersect_h =
            interval_overlap([box1.ymin,
            box1.ymax], [box2.ymin, box2.ymax])
        intersect_area = intersect_w *
            intersect_h

        w1, h1 = box1.xmax-box1.xmin,
```

```
            box1.ymax-box1.ymin
        w2, h2 = box2.xmax-box2.xmin,
            box2.ymax-box2.ymin

        union_area = w1*h1 + w2*h2 -
            intersect_area
        return float(intersect_area) /
            union_area
```

6. ***Non-Max Supression*** **(NMS)**
   Proses selanjutnya adalah membuat *function Non-Max Supression* yang bertujuan menspesifikan kotak pembatas terhadap objek yang dideteksi. Berikut *source code* pada proses NMS :

```
    def nonMaximumSuppression(outputs,
        anchors, masks, classes):
        boxes, conf, out_type = [], [], []

        for output in outputs:
            boxes.append(tf.reshape(output[0],
                (tf.shape(output[0])[0], -1,
                tf.shape(output[0])[-1])))
            conf.append(tf.reshape(output[1],
                (tf.shape(output[1])[0], -1,
                tf.shape(output[1])[-1])))
            out_type.append(tf.reshape(output[2],
                (tf.shape(output[2])[0], -1,
                tf.shape(output[2])[-1])))

        bbox = tf.concat(boxes, axis=1)
        confidence = tf.concat(conf, axis=1)
        class_probs = tf.concat(out_type,
            axis=1)

        scores = confidence * class_probs
```

```
        boxes, scores, classes,
            valid_detections =
            tf.image.combined_non_max_suppression
            (
           boxes=tf.reshape(bbox,
               (tf.shape(bbox)[0], -1, 1, 4)),
           scores=tf.reshape(
           scores, (tf.shape(scores)[0], -1,
               tf.shape(scores)[-1])),
           max_output_size_per_class=100,
           max_total_size=100,
           iou_threshold=yolo_iou_threshold,
           score_threshold=yolo_score_threshold
        )

        return boxes, scores, classes,
            valid_detections
```

7. ***Trained Weights***
   Pada proses ini menggunakan prinsip *transfer learning* dimana dalam membuat bobot baru untuk objek "Plat Mobil" dan "Plat Motor" menggunakan bobot dari *yolov3.weights* dengan jaringan dasar *darknet53.conv.74*. Proses ini menggunakan *Google Collaboratory*

```
!./darknet detector train data/obj.data
    cfg/yolov3_custom3.cfg
    darknet53.conv.74 -dont_show
```

8. **YoloV3 *Loss Function***
   Pada proses ini membuat sebuah fungsi untuk menghitung *loss*. Berikut *source code* dalam menghitung *Yolo Loss* :

```
        def YoloLoss(anchors, classes=2,
            ignore_thresh=0.5):
           def yolo_loss(y_true, y_pred):
              # 1. transform all pred outputs
              # y_pred: (batch_size, grid, grid,
                  anchors, (x, y, w, h, obj,
                  ...cls))
              pred_box, pred_obj, pred_class,
                  pred_xywh = yolo_boxes(
              y_pred, anchors, classes)
              pred_xy = pred_xywh[..., 0:2]
              pred_wh = pred_xywh[..., 2:4]

              # 2. transform all true outputs
              # y_true: (batch_size, grid, grid,
                  anchors, (x1, y1, x2, y2, obj,
                  cls))
              true_box, true_obj, true_class_idx =
                  tf.split(
              y_true, (4, 1, 1), axis=-1)
              true_xy = (true_box[..., 0:2] +
                  true_box[..., 2:4]) / 2
              true_wh = true_box[..., 2:4] -
                  true_box[..., 0:2]

              # give higher weights to small boxes
              box_loss_scale = 2 - true_wh[..., 0]
                  * true_wh[..., 1]

              # 3. inverting the pred box equations
              grid_size = tf.shape(y_true)[1]
              grid =
                  tf.meshgrid(tf.range(grid_size),
                  tf.range(grid_size))
              grid = tf.expand_dims(tf.stack(grid,
                  axis=-1), axis=2)
              true_xy = true_xy *
```

```
            tf.cast(grid_size, tf.float32) -
            \
        tf.cast(grid, tf.float32)
        true_wh = tf.math.log(true_wh /
            anchors)
        true_wh =
            tf.where(tf.math.is_inf(true_wh),
        tf.zeros_like(true_wh), true_wh)

        # 4. calculate all masks
        obj_mask = tf.squeeze(true_obj, -1)
        # ignore false positive when iou is
            over threshold
        true_box_flat =
            tf.boolean_mask(true_box,
            tf.cast(obj_mask, tf.bool))
        best_iou =
            tf.reduce_max(broadcast_iou(
        pred_box, true_box_flat), axis=-1)
        ignore_mask = tf.cast(best_iou <
            ignore_thresh, tf.float32)

        # 5. calculate all losses
        xy_loss = obj_mask * box_loss_scale
            * \
        tf.reduce_sum(tf.square(true_xy -
            pred_xy), axis=-1)
        wh_loss = obj_mask * box_loss_scale
            * \
        tf.reduce_sum(tf.square(true_wh -
            pred_wh), axis=-1)
        obj_loss =
            binary_crossentropy(true_obj,
            pred_obj)
        obj_loss = obj_mask * obj_loss + \
        (1 - obj_mask) * ignore_mask *
            obj_loss
        # TODO: use binary_crossentropy
```

```
                instead
            class_loss = obj_mask *
                sparse_categorical_crossentropy(
            true_class_idx, pred_class)

            # 6. sum over (batch, gridx, gridy,
                anchors) => (batch, 1)
            xy_loss = tf.reduce_sum(xy_loss,
                axis=(1, 2, 3))
            wh_loss = tf.reduce_sum(wh_loss,
                axis=(1, 2, 3))
            obj_loss = tf.reduce_sum(obj_loss,
                axis=(1, 2, 3))
            class_loss =
                tf.reduce_sum(class_loss,
                axis=(1, 2, 3))

            return xy_loss + wh_loss + obj_loss
                + class_loss
        return yolo_loss
```

4.6.2 Implementasi Tahap Pengujian

Implementasi tahap pengujian mengacu pada desain arsitektur YoloV3. Tahap pengujian bertujuan untuk menguji model dan bobot yang telah dibuat pada data yang lain. Pada tahap ini sistem diimplementasikan dengan menggunakan pemrograman Java.

Gambar 4.47: Halaman Utama

1. **Implementasi Akuisisi Video**
   Proses ekstraksi *frame* dilakukan setelah user menginputkan video. Proses ekstraksi *frame* diimplementasikan dalam *source code* berikut :

```
while(true){
   if(cap.grab()){
      cap.retrieve(frame);apo
   }
}
```

2. **Implementasi Pendefinisian ROI**
   Proses selanjutnya adalah pendefinisan ROI pada setiap *frame* yang telah terekstraksi. Proses pendefinisian ROI diimplementasikan dalam *source code* berikut :

```
Rect roi = new Rect(50, frame.height() /
    2, frame.width() - 100, 380);
Imgproc.rectangle(frame, roi, new
    Scalar(0, 0, 0), 3);
```

3. ***Import Trained Weights***
   Proses ini dilakukan dengan mengimport hasil *trained weights* dari proses *training*. Berikut *source code* yang diberikan :

```
String modelWeights =
    "yolov3_custom_final.weights";
String modelConfiguration =
    "yolov3_custom.cfg";
String classes = "obj.names";
```

4. **Implementasi *Tracking***
   Proses ini dilakukan *tracing* setiap plat kendaraan yang terdeteksi agar mengetahui hubungan objek dalam setiap *frame*.

```
if (deteksi == 0) {
   BanyakObjek = ind.length;
   for (int i = 0; i < ind.length; ++i) {
      int idx = ind[i];
      Rect box = boxesArray[idx];
      String nama =
          Label.get(clsIds.get(idx)) + " :
          " + String.format("%,.2f",
          confs.get(idx));

      if (clsIds.get(idx) == 0) {
         color = new Scalar(255, 0, 255);
      } else {
         color = new Scalar(0, 255, 0);
      }
      plat.add(new Plate(index, box));
      imgDeteksi.add(new Plate(index, box,
          confs.get(idx), PosFrame,
```

```
                Label.get(clsIds.get(idx))));

            index++;
        }
    }
    else{
        BanyakObjek = ind.length;
        CekPlate cekDeteksi = null;
        for (int i = 0; i < ind.length; ++i) {
            int idx = ind[i];
            Rect box = boxesArray[idx];
            String nama =
                Label.get(clsIds.get(idx)) + " :
                " + String.format("%,.2f",
                confs.get(idx));

            if (clsIds.get(idx) == 0) {
                color = new Scalar(255, 0, 255);
            } else {
                color = new Scalar(0, 255, 0);
            }

            if (plat.size() > ind.length) {
                h++;
                if (h == 10) {
                    plat =
                        plat.subList(plat.size() -
                        ind.length, plat.size());
                    h = 0;
                }
            }

            CekPlate cek = new CekPlate(plat,
                box);
            if (cek.getCekInside()) {
                index = plat.get(
                    cek.getIndexInside()).ID;
                plat.set(cek.getIndexInside(),
```

```
                new Plate(index, box));
            } else {
                index = plat.get(plat.size() -
                    1).getID() + 1;
                plat.add(new Plate(index, box));
            }
        }
```

5. **Implementasi Deteksi Plat Nomor Kendaraan**
   Pada proses ini dilakukan dengan mendeteksi objek plat nomor kendaraan dan memanfaatkan *library* openCV. berikut *source code* dalam melakukan pendeteksian objek plat nomor kendaraan :

```
        Mat area = frame.submat(roi);

        Mat blob = Dnn.blobFromImage(area,
            0.00392, sz, new Scalar(0), true,
            false);
        net.setInput(blob);
        net.forward(result, outBlobNames);
        float confThreshold = 0.25f;

        List<Integer> clsIds = new
            ArrayList<>();
        List<Float> confs = new ArrayList<>();
        List<Rect> rects = new ArrayList<>();

        Vector<Point> detections = new
            Vector<>();
        List<Plate> tampungPlate = new
            ArrayList<>();
        Vector<Rect> array = new
            Vector<Rect>();

        for (int i = 0; i < result.size();
```

```
            i++) {
          Mat level = result.get(i);
          for (int j = 0; j < level.rows();
              j++) {
            Mat row = level.row(j);
            Mat scores = row.colRange(5,
                level.cols());
            Core.MinMaxLocResult mm =
                Core.minMaxLoc(scores);
            float confidence = (float)
                mm.maxVal;
            Point classIdPoint = mm.maxLoc;
            if (confidence > confThreshold) {
              int centerX = (int)
                  (row.get(0, 0)[0] *
                  area.cols());
              int centerY = (int)
                  (row.get(0, 1)[0] *
                  area.rows());
              int width = (int) (row.get(0,
                  2)[0] * area.cols());
              int height = (int) (row.get(0,
                  3)[0] * area.rows());
              int left = centerX - width / 2;
              int top = centerY - height / 2;
              clsIds.add((int)
                  classIdPoint.x);
              confs.add((float) confidence);
              rects.add(new Rect(left, top,
                  width, height));
            }
          }
        }
```

BAB V
UJI COBA DAN PEMBAHASAN

Pada bab ini dijelaskan mengenai pengujian program dan pembahasan dari hasil uji coba. Pengujian yang dilakukan adalah pengujian program dengan input video rekaman kendaraan bergerak di jalan raya.

5.1 Data Uji Coba

Uji coba pada program dalam Tugas Akhir ini dilakukan terhadap pada dua data uji coba, diantaranya uji coba pada data *validation* dan data *testing*. Data *validation* digunakan untuk menguji tingkat keberhasilan dari model jaringan yang dibuat pada proses training, data *validation* ini berupa data *image* yang telah dilakukan anotasi data. Sedangkan data *testing* digunakan untuk menguji coba pada data yang benar-benar baru agar dapat mengetahui kinerja YOLOV3 dalam mendeteksi lokasi plat nomor kendaraan bergerak. Daftar input data uji coba tersebut disajikan pada Tabel 5.1, Tabel 5.2, dan Tabel 5.3 seperti di bawah ini :

Tabel 5.1: Data Train

Data Train	**Banyak Objek**		**Jumlah Objek**	**Jumlah Data Gambar**
	Plat Motor	**Plat Mobil**		
Primer	2279	861	3140	1792
Sekunder	0	7312	7312	5006
Total	2279	8173	10452	6798

Tabel 5.2: Data Uji Coba Validation

Data Valid	Banyak Objek		Jumlah Objek	Jumlah Data Gambar
	Plat Motor	Plat Mobil		
Primer	784	171	955	447
Sekunder	0	1549	1549	1097
Total	784	1720	2504	1544

Tabel 5.3: Data Uji Coba Testing

No.	Nama	Waktu	Tempat
1	Video 1	31 Juni 2020 11:40 WIB	Jembatan Penyeberangan Jl. Raya Anggaswangi 7 Ngepung, Sidokepung, Kec. Buduran, Kabupaten Sidoarjo
2	Video 2	19 Juni 2020 08:10 WIB	Jembatan Penyeberangan Jl. Raya Wonokromo, Kec. Wonokromo, Kota Surabaya
3	Video 3	18 Juni 2020 15:05 WIB	Jembatan Penyeberangan Jl. Basuki Rahmat, Embong Kaliasin, Kec. Genteng, Kota Surabaya

Pada data *train* dan *validation*, data primer diperoleh dari mengambil data secara langsung sedangkan data sekunder diperoleh dari *storage.googleapis.com/openimages/web/index.html*. Adapun batasan yang dilakukan pada pengujian ini diantaranya :

- Data video yang digunakan memiliki *Frame Rate per Second* (FPS) = 50 dan ukuran dimensi video yang digunakan adalah 1920×1080.
- *Confidence threshold* pada proses uji coba data *testing* = 0.5 dan *NMS threshold* = 0.4
- Perangkat yang digunakan dapat ditunjukkan pada Tabel 5.4

Tabel 5.4: Perangkat Program yang Digunakan

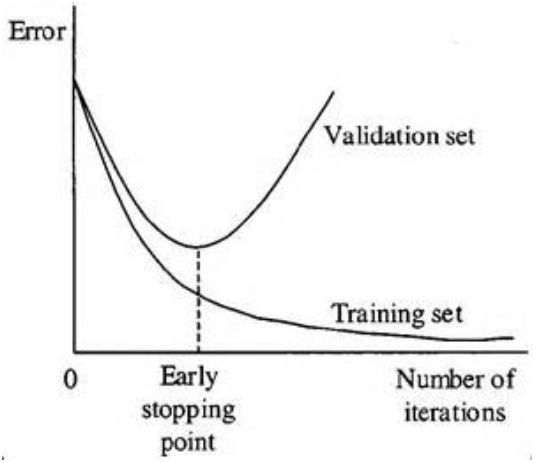
Jenis Perangkat	Kebutuhan
Perangkat Keras	1. Processor Intel(R) Core(TM) i5-2400S CPU 2.50GHz (4 CPUs) 2.5GHz 2. Memory 4.00GB RAM 3. Windows 10 Pro 64-bit (10.0, Build 10586)
Perangakt Lunak	1. Python Anaconda 3.6 2. OpenCV 4.0 3. Netbeans 8.2 dengan Bahasa pemrograman Java 4. Google Colab

5.2 Hasil Uji Coba

Berikut merupakan hasil uji coba yang dilakukan pada data *validation* dan data *testing*.

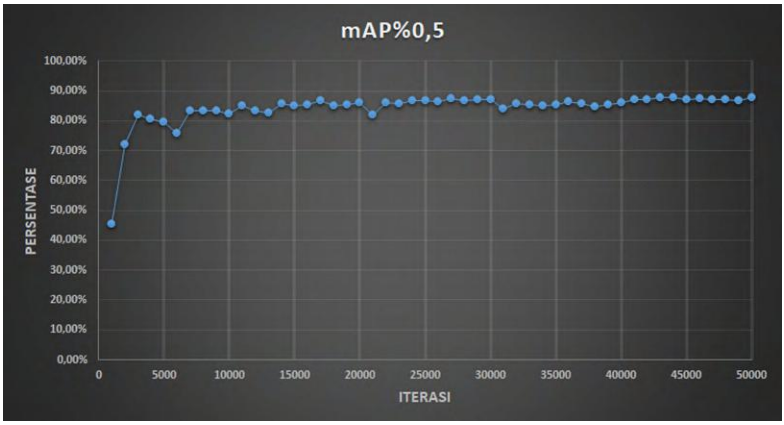
5.2.1 Hasil Uji Coba Data Valid

Pada tahap ini dilakukan pengujian terhadap proses training untuk memperoleh *trained weights* terbaik. Untuk memperoleh hasil yang sesuai, maka dilakukan pengujian pada proses training dengan menggunakan data *validation*. Skenario tahap *training* pada YOLOV3 adalah skenario yang dibuat untuk memperoleh *trained weights* pada *Early Stopping Point* agar tidak terjadi *overfitting*. *Early Stopping Point* yang dimaksud dapat dilihat pada Gambar 5.1.



Gambar 5.1: Proses Menentukan *Trained Weights*

Hasil uji coba proses training dengan menggunakan data *validation* pada penelitian ini menggunakan iterasi sebanyak 50000 dan *confidence threshold* 0.25. Dari banyaknya iterasi tersebut akan dicari *mean average precision* (mAP%0,5) terbaik. Hasil uji coba tersebut ditunjukkan grafik pada Gambar 5.2.



Gambar 5.2: Grafik mAP% pada Proses Training

Berdasarkan grafik Gambar 5.2 di atas diperoleh nilai mAP%0,5 tertinggi adalah 87.89%. Pada grafik tersebut mulai iterasi 41000 sampai 50000 diperoleh rata-rata mAP%0,5 adalah 87%, Hal ini dapat diartikan bahwa pada iterasi 41000 merupakan *Early Stopping Point* pada model yang dilatih. Jika diteruskan iterasinya model yang dibuat akan mengalami *overfitting*. Sehingga model *trained weights* yang dibuat sampai pada iterasi 41000 sudah cukup baik dalam melakukan pendeteksian plat nomor kendaraan. Adapun nilai loss yang diperoleh adalah 0.1077.

5.2.2 Pembahasan Hasil Uji Coba Data Valid

Setelah dilakukan proses uji coba training terhadap data *validation*, kemudian dilakukan evaluasi dengan menghitung presisi, recall, dan IoU akurasi. Berikut ini merupakan ilustrasi perhitungan tiap iterasi yang disediakan pada Tabel 5.5.

Tabel 5.5: Hasil Uji Coba Tiap Iterasi

Iterasi	TP	TN	FP	FN	mAP%0.5	Precision	Recall	F1-Score	IoU
1000	1179	0	725	1325	45,60%	61,92%	47,08%	53,49%	40,66%
2000	1907	0	493	597	72,26%	79,46%	76,16%	77,77%	56,96%
3000	2048	0	274	456	82,02%	88,20%	81,79%	84,87%	65,22%
4000	1985	0	266	519	80,77%	88,18%	79,27%	83,49%	65,74%
5000	2107	0	418	397	79,49%	83,45%	84,15%	83,79%	62,07%
6000	2074	0	447	430	75,94%	82,27%	82,83%	82,55%	60,30%
7000	2076	0	265	428	83,30%	88,68%	82,91%	85,70%	67,53%
8000	2096	0	288	408	83,42%	87,92%	83,71%	85,76%	65,71%
9000	2100	0	256	404	83,28%	89,13%	83,87%	86,42%	67,27%
10000	2096	0	279	408	82,46%	88,25%	83,71%	85,92%	66,68%
11000	2141	0	235	363	85,11%	90,11%	85,50%	87,75%	69,31%
12000	2098	0	241	406	83,52%	89,70%	83,79%	86,64%	68,04%
13000	2043	0	178	461	82,65%	91,99%	81,59%	86,48%	69,88%
14000	2100	0	169	404	85,93%	92,55%	83,87%	87,99%	69,78%
15000	2176	0	291	328	85,06%	88,20%	86,90%	87,55%	66,70%
16000	2145	0	231	359	85,43%	90,28%	85,66%	87,91%	68,68%
17000	2175	0	204	329	86,88%	91,42%	86,86%	89,08%	70,24%
18000	2093	0	149	411	85,08%	93,35%	83,59%	88,20%	71,91%
19000	2143	0	227	361	85,54%	90,42%	85,58%	87,94%	69,56%
20000	2113	0	155	391	86,21%	93,17%	84,38%	88,56%	71,33%
21000	2106	0	303	398	81,87%	87,42%	84,11%	85,73%	65,43%
22000	2118	0	177	386	86,22%	92,29%	84,58%	88,27%	71,28%
23000	2135	0	174	369	85,85%	92,46%	85,26%	88,72%	70,52%
24000	2132	0	175	372	86,83%	92,41%	85,14%	88,63%	71,18%
25000	2140	0	150	364	86,66%	93,45%	85,46%	89,28%	72,91%
26000	2142	0	153	362	86,58%	93,33%	85,54%	89,27%	73,01%
27000	2158	0	156	346	87,32%	93,26%	86,18%	89,58%	72,76%
28000	2150	0	160	354	86,82%	93,07%	85,86%	89,32%	72,70%
29000	2146	0	149	358	87,10%	93,51%	85,70%	89,44%	73,06%
30000	2146	0	140	358	87,29%	93,88%	85,70%	89,60%	73,34%
31000	2117	0	199	387	84,22%	91,41%	84,54%	87,84%	70,25%
32000	2120	0	195	384	85,62%	91,58%	84,66%	87,99%	70,40%
33000	2142	0	199	362	85,44%	91,50%	85,54%	88,42%	69,70%
34000	2124	0	166	380	85,21%	92,75%	84,82%	88,61%	70,92%
35000	2109	0	169	395	85,31%	92,58%	84,23%	88,21%	70,84%
36000	2135	0	165	369	86,30%	92,83%	85,26%	88,88%	71,70%
37000	2144	0	219	360	85,90%	90,73%	85,62%	88,10%	69,49%
38000	2086	0	175	418	84,70%	92,26%	83,31%	87,56%	71,15%
39000	2109	0	174	395	85,29%	92,38%	84,23%	88,11%	71,44%
40000	2119	0	173	385	86,20%	92,45%	84,62%	88,37%	71,34%
41000	2153	0	164	351	87,05%	92,92%	85,98%	89,32%	72,37%
42000	2161	0	169	343	87,16%	92,75%	86,30%	89,41%	72,23%
43000	**2158**	0	**142**	**346**	**87,89%**	**93,83%**	**86,18%**	**89,84%**	**73,43%**
44000	2151	0	152	353	87,73%	93,40%	85,90%	89,49%	73,09%
45000	2151	0	154	353	87,16%	93,32%	85,90%	89,46%	72,87%
46000	2151	0	154	353	87,38%	93,32%	85,90%	89,46%	73,15%
47000	2150	0	158	354	87,01%	93,15%	85,86%	89,36%	72,83%
48000	2152	0	154	352	87,06%	93,32%	85,94%	89,48%	72,87%
49000	2140	0	153	364	86,70%	93,33%	85,46%	89,22%	72,84%
50000	2150	0	134	354	87,67%	94,13%	85,86%	89,81%	73,87%

Berdasarkan Tabel 5.5 di atas, dengan didapatkan presisi, recall, dan F-1 score dari rumus perhitungan sebagai berikut :

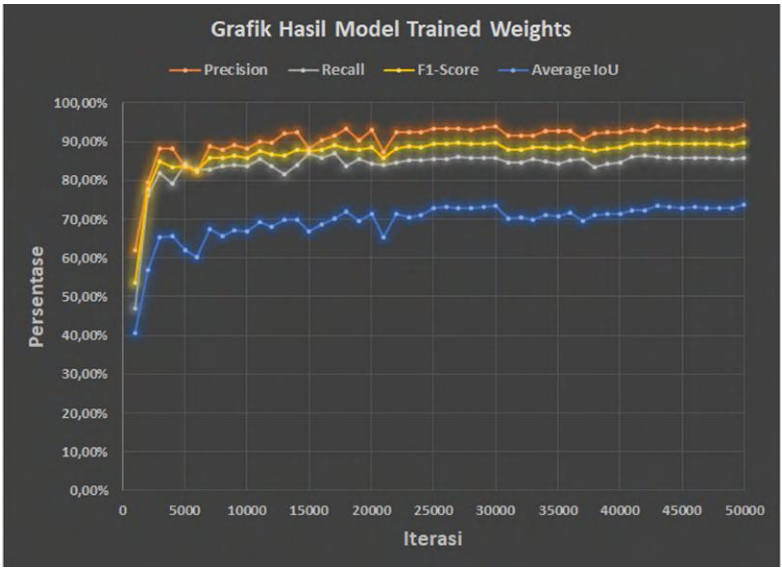
$$\text{Presisi} = \frac{TP}{TP + FP} \times 100\% \tag{5.1}$$

$$\text{Recall} = \frac{TP}{TP + FN} \times 100\% \tag{5.2}$$

$$\text{F-1 Score} = \frac{2 \times (\text{Recall} \times \text{Presisi})}{(\text{Recall} + \text{Presisi})} \times 100\% \tag{5.3}$$

Berdasarkan Tabel 5.3. dapat disimpulkan bahwa model yang berhasil dibuat sudah cukup baik dalam melakukan pendeteksian plat nomor, hal ini diketahui dari nilai mAP%0.5 tertinggi sistem yang berada pada iterasi 43000 sebesar 87.89%. Sementara nilai *precision* 93.83%, nilai *recall* 86.18%, *F1-score* 89.84% dan IoU 73.43%.

Untuk mengetahui hasil grafik dari nilai presisi, recall, F-1 score, dan IoU maka grafik tersebut dapat dilihat pada Gambar 5.3.

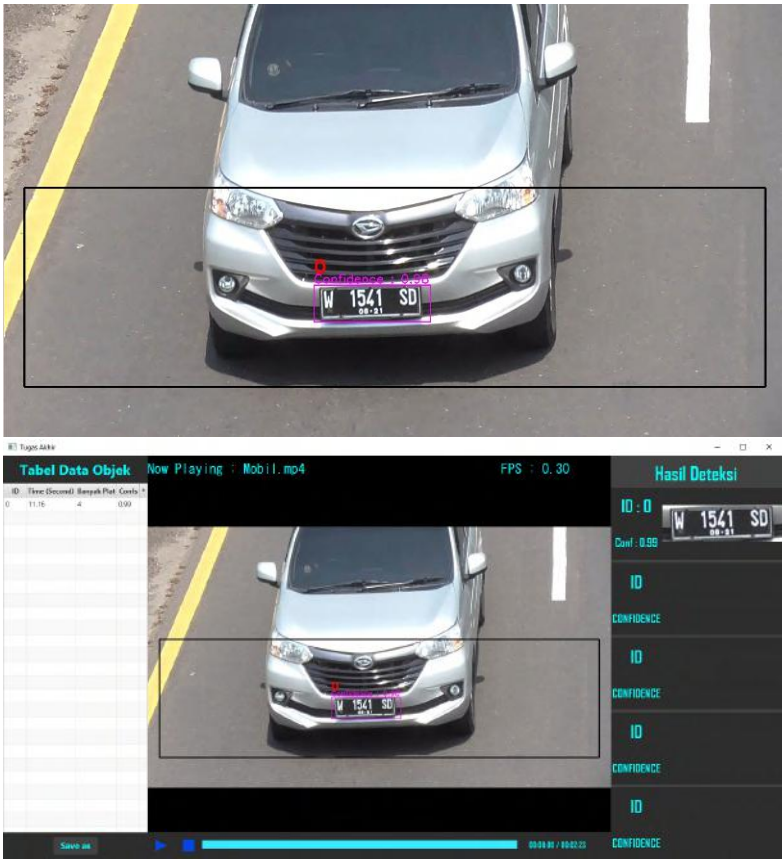


Gambar 5.3: Grafik Nilai Evaluasi Model *Trained Weights*

5.2.3 Hasil Uji Coba Data *Testing*

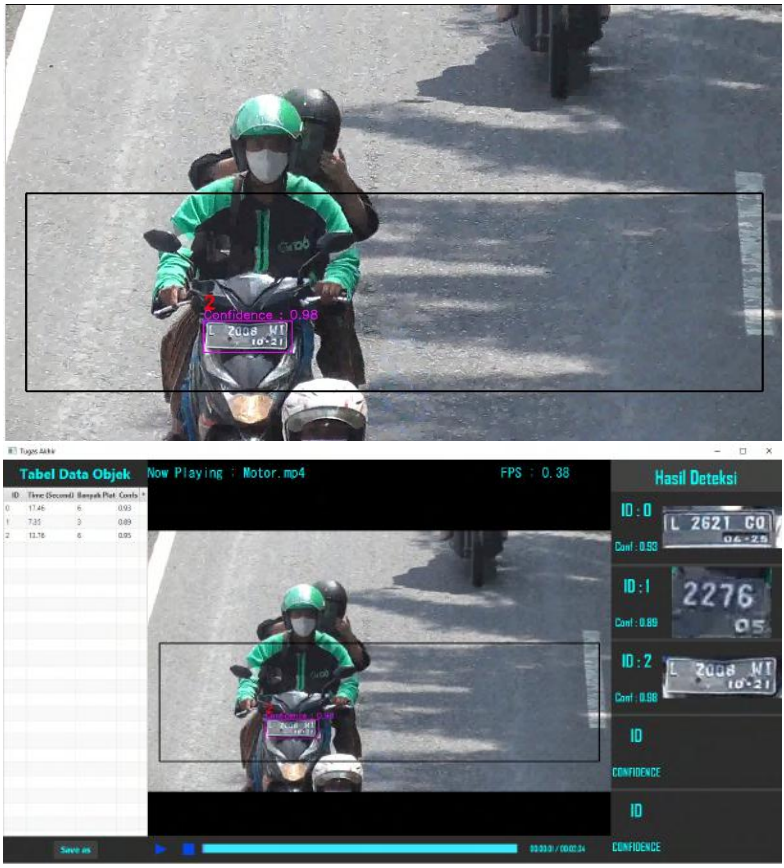
Berikut adalah hasil uji coba yang dilakukan pada tiga rekaman video kendaraan bergerak :

1. **Hasil Uji Coba Video 1**
   Pada Gambar 5.4 menunjukkan hasil dari salah satu *frame* pada Video 1 setelah dilakukan deteksi objek dan *tracking*. Terlihat bahwa kendaraan yang melewati ROI akan dilakukan deteksi plat nomor kendaraan dan ditampilkan di samping video.



Gambar 5.4: Hasil Deteksi Plat Nomor Pada Video 1

2. **Hasil Uji Coba Video 2**
   Pada Gambar 5.5 menunjukkan hasil dari salah satu *frame* pada Video 2 setelah dilakukan deteksi objek dan *tracking*. Terlihat bahwa kendaraan yang melewati ROI akan dilakukan deteksi plat nomor kendaraan dan ditampilkan di samping video.



Gambar 5.5: Hasil Deteksi Plat Nomor Pada Video 2

3. **Hasil Uji Coba Video 3**
   Pada Gambar 5.6 menunjukkan hasil daria slah satu *frame* pada Video 3 setelah dilakukan deteksi objek dan *tracking*. Terlihat bahwa kendaraan yang melewati ROI akan dilakukan deteksi plat nomor kendaraan dan

ditampilkan di samping video.

Gambar 5.6: Hasil Deteksi Plat Nomor Pada Video 3

5.2.4 Pembahasan Uji Coba Data *Testing*

Uji coba yang dilakukan pada data ini menggunakan data video sebanyak 3 rekaman video. Pada dasarnya jumlah objek yang terdeteksi harus sama dengan jumlah objek sebenarnya

dalam ROI. Pada Tabel 5.6 diberikan data hasil uji coba pada tiga video. Adapun komposisi pada 3 video tersebut diantaranya video 1 dilakukan uji coba pada objek plat mobil saja. Video 2 dilakukan uji coba pada objek plat mobil dan plat motor dengan lebih dominan banyak plat motor. Sedangkan video 3 dilakukan uji coba pada objek plat mobil dan motor yang jumlahnya tidak jauh beda. Pada video 1 pengambilan data dilakukan di jalan tol sehingga banyak objek plat motor adalah 0.

Keterangan :

- JS : Jumlah Sebenarnya
- JT : Jumlah Terdeteksi
- TT : Tidak Terdeteksi
- SD : Salah Deteksi

Tabel 5.6: Hasil Uji Coba pada 3 Video

No.	Nama	Kategori	JS	JT	TT	SD
1	Video 1	Plat Motor	0	0	0	0
		Plat Mobil	45	44	1	2
2	Video 2	Plat Motor	77	69	8	1
		Plat Mobil	4	4	0	0
3	Video 3	Plat Motor	37	30	7	2
		Plat Mobil	27	24	3	1

Data pada Tabel 5.6 didapatkan dengan menghitung secara manual dari video yang ditampilkan pada program. Perhitungan performa deteksi plat nomor menggunakan

rumus seperti di bawah ini :

$$\text{recall} = \frac{\text{Jumlah terdeteksi}}{\text{Jumlah sebenarnya}} \times 100\% \tag{5.4}$$

$$\text{precision} = \frac{\text{Jumlah terdeteksi} - \text{Salah deteksi}}{\text{Jumlah terdeteksi}} \times 100\% \tag{5.5}$$

$$\text{accuracy} = \frac{\text{Jumlah terdeteksi} - \text{Salah deteksi}}{\text{Jumlah sebenarnya}} \times 100\% \tag{5.6}$$

Pada persamaan 5.5, *recall* menunjukkan persentase dari plat nomor yang terdeteksi dibandingkan dengan jumlah plat sebenarnya pada area ROI. Sedangkan pada persamaan 5.6, *precision* deteksi menunjukkan persentase dari plat nomor yang terdeteksi dengan benar dibandingkan dengan jumlah plat yang terdeteksi seluruhnya pada area ROI dan pada persamaan 5.7,*accuracy* menunjukkan persentase dari plat nomor yang terdeteksi dengan benar dibandingkan dengan jumlah plat sebenarnya. Hasil perhitungan tersebut ditampilkan pada Tabel 5.7. Adapun contoh dalam keterangan salah deteksi dapat dilihat pada gambar 5.7.

Gambar 5.7: Contoh Salah Deteksi

Pada gambar 5.7 sistem berhasil mendeteksi sebuah objek plat tapi, tidak sesuai dengan lokasi yang diinginkan.

Tabel 5.7: Persentase Hasil Uji Coba pada 3 Video

No.	Nama	Kategori	Recall	Precision	Accuracy
1	Video 1	Plat Motor	0,00%	0,00%	0,00%
		Plat Mobil	97,78%	95,45%	93,33%
2	Video 2	Plat Motor	89,61%	98,55%	88,31%
		Plat Mobil	100,00%	100,00%	100,00%
3	Video 3	Plat Motor	81,08%	93,33%	75,68%
		Plat Mobil	88,89%	95,83%	85,19%
Rata - Rata		**Plat Motor**	**85,35%**	**95,94%**	**81,99%**
		Plat Mobil	**95,56%**	**97,10%**	**92,84%**

Dari hasil Pada Tabel 5.7, nilai rata-rata *recall* pada plat motor sebesar 85.35% dan plat mobil sebesar 95.56%. Nilai rata-rata *precision* pada plat motor sebesar 95.94% dan plat mobil sebesar 97.1%. Sedangkan nilai rata-rata *accuracy* pada plat motor sebesar 81.99% dan plat mobil sebesar 92.84%. Sehingga dari nilai rata-rata tersebut dapat disimpulkan

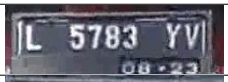
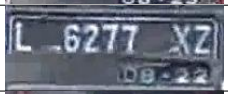
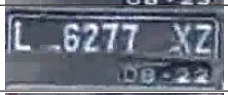
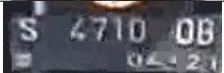
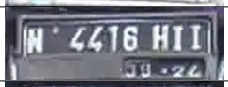
bahwa nilai *recall*, *precision* dan *accuracy* yang dihasilkan pada plat mobil lebih besar dibandingkan dengan plat motor. Hal ini dikarenakan dalam proses *training* banyaknya data dan variasi kondisi pada kelas objek plat mobil lebih banyak dibandingkan kelas objek motor. Pada proses *training* data pada objek plat mobil sebanyak 8173 sedangkan pada objek plat motor sebanyak 2279. Sehingga dari faktor tersebut hasil uji coba pada objek plat mobil lebih baik daripada plat motor.

Untuk menghitung keakuratan dari lokalisasi objek plat nomor maka akan dihitung seberapa besar nilai error hasil *cropping* prediksi plat nomor dengan lokalisasi plat nomor secara empiris atau yang dapat terlihat secara visual. Lokalisasi plat nomor secara empiris dilakukan secara manual dengan mendefiniskan ukuran *bounding box* yang sebenarnya pada hasil prediksi. Adapun hasil plat nomor yang terdeteksi dengan lokasi plat nomor secara empiris disajikan pada Tabel 5.8, 5.9 dan 5.10 secara berturut-turut pada ketiga video.

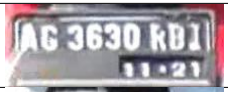
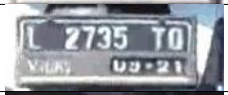
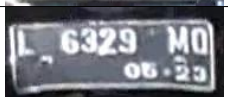
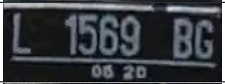
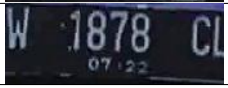
Tabel 5.8: Hasil Uji Coba pada Video 1

No.	Video	ID	Hasil Prediksi	Lokalisasi Plat
1	Video 1	10		
2	Video 1	11		
3	Video 1	13		
4	Video 1	15		
5	Video 1	16		
6	Video 1	17		
7	Video 1	21		
8	Video 1	24		
9	Video 1	26		
10	Video 1	28		

Tabel 5.9: Hasil Uji Coba pada Video 2

No.	Video	ID	Hasil Prediksi	Lokalisasi Plat
1	Video 2	0		
2	Video 2	2		
3	Video 2	5		
4	Video 2	6		
5	Video 2	7		
6	Video 2	8		
7	Video 2	13		
8	Video 2	14		
9	Video 2	16		
10	Video 2	19		

Tabel 5.10: Hasil Uji Coba pada Video 3

No.	Video	ID	Hasil Prediksi	Lokalisasi Plat
1	Video 3	4		
2	Video 3	10		
3	Video 3	15		
4	Video 3	17		
5	Video 3	18		
6	Video 3	25		
7	Video 3	26		
8	Video 3	28		
9	Video 3	33		
10	Video 3	47		

Pada Tabel 5.8, 5.9 dan 5.10 ditunjukkan kolom hasil prediksi merupakan hasil dari deteksi menggunakan YOLOV3 dengan *confidence* terbesar dari tiap *frame* pada area ROI yang kemudian dilakukan *cropping*. Sedangkan pada kolom lokalisasi plat didapatkan secara manual dari hasil deteksi tersebut dengan berdasarkan letak plat yang terlihat secara visual. Adapun perhitungan untuk mencari keakuratan dan error deteksi terhadap lokasi plat disajikan pada persamaan

5.8 dan 5.9.

$$\text{Keakuratan} = \frac{\text{Luas Area Plat Secara Empiris}}{\text{Luas Area Plat yang Terdeteksi}} \times 100\% \tag{5.7}$$

$$\text{Error} = 1 - \text{Keakuratan} \tag{5.8}$$

Sehingga didapatkan hasil perhitungan tersebut disajikan pada Tabel 5.11, 5.12 dan 5.13

Tabel 5.11: Hasil Perhitungan Keakuratan pada Video 1

No.	Video	ID	Luas Plat Terdeteksi (pixel)	Luas Plat Secara Empiris (pixel)	Keakuratan	Error
1	Video 1	10	25700	20408	79,41%	20,59%
2	Video 1	11	25833	22917	88,71%	11,29%
3	Video 1	13	23958	20569	85,85%	14,15%
4	Video 1	15	26964	20560	76,25%	23,75%
5	Video 1	16	13510	12978	96,06%	3,94%
6	Video 1	17	14720	13793	93,70%	6,30%
7	Video 1	21	19012	18493	97,27%	2,73%
8	Video 1	24	19040	18493	97,13%	2,87%
9	Video 1	26	17255	16554	95,94%	4,06%
10	Video 1	28	15708	15145	96,42%	3,58%
Rata - Rata			**20170**	**17991**	**90,67%**	**9,33%**

Tabel 5.12: Hasil Perhitungan Keakuratan pada Video 2

No.	Video	ID	Luas Plat Terdeteksi (pixel)	Luas Plat Secara Empiris (pixel)	Keakuratan	Error
1	Video 2	0	18000	14904	82,80%	17,20%
2	Video 2	2	10266	7820	76,17%	23,83%
3	Video 2	5	13800	12675	91,85%	8,15%
4	Video 2	6	17480	14214	81,32%	18,68%
5	Video 2	7	16458	12033	73,11%	26,89%
6	Video 2	8	17082	13266	77,66%	22,34%
7	Video 2	13	16936	13580	80,18%	19,82%
8	Video 2	14	17155	14416	84,03%	15,97%
9	Video 2	16	16037	15276	95,25%	4,75%
10	Video 2	19	17200	12730	74,01%	25,99%
Rata - Rata			**16041**	**13091**	**81,64%**	**18,36%**

Tabel 5.13: Hasil Perhitungan Keakuratan pada Video 3

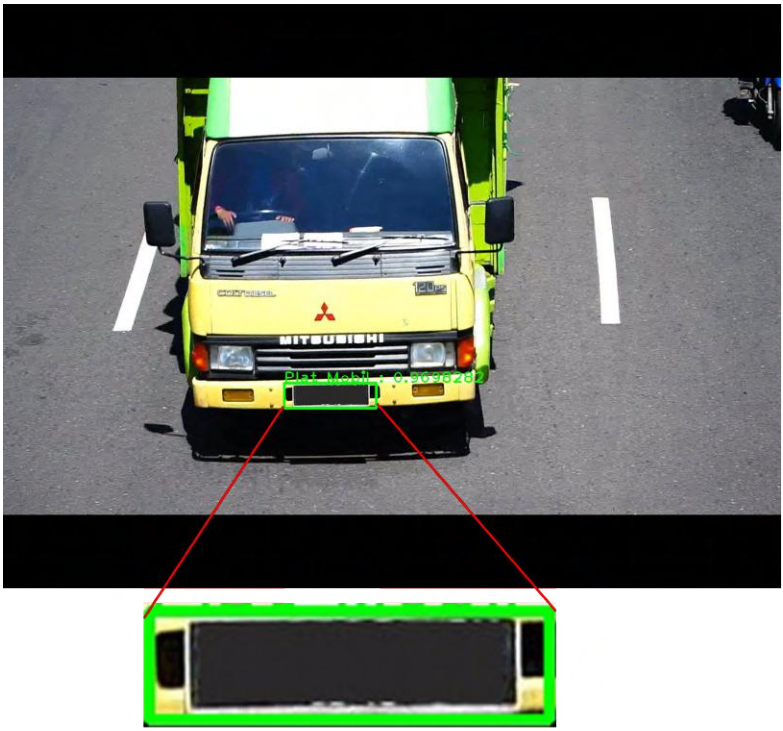
No.	Video	ID	Luas Plat Terdeteksi (pixel)	Luas Plat Secara Empiris (pixel)	Keakuratan	Error
1	Video 3	4	24304	15862	65,26%	34,74%
2	Video 3	10	14430	10962	75,97%	24,03%
3	Video 3	15	30603	25543	83,47%	16,53%
4	Video 3	17	17724	14852	83,80%	16,20%
5	Video 3	18	26230	24734	94,30%	5,70%
6	Video 3	25	29458	25270	85,78%	14,22%
7	Video 3	26	28215	27140	96,19%	3,81%
8	Video 3	28	36252	31411	86,65%	13,35%
9	Video 3	33	17577	14775	84,06%	15,94%
10	Video 3	47	27200	26016	95,65%	4,35%
Rata - Rata			**25199**	**21657**	**85,11%**	**14,89%**

Pada Tabel 5.11, 5.12 dan 5.13. dapat disimpulkan bahwa model yang telah dibuat memiliki keakuratan lokalisasi objek plat nomor pada video mobil lebih baik daripada video motor. Hal ini dapat dilihat dari rata-rata keakuratan pada video 1 lebih besar daripada video 2 dan video 3 yaitu sebesar 90.67%. Kemudian rata-rata keakuratan pada video 2 sebesar 81.64% dan video 3 sebesar 85.11%. Dari ketiga video ini didapatkan rata-rata total keakuratannya adalah 85.81%. Adapun beberapa faktor penyebab rata-rata keakuratan pada plat mobil lebih baik daripada plat motor diantaranya :

- Banyak data objek plat motor dan mobil ketika proses *training*. Karena dalam penelitian ini data yang digunakan saat proses *training* pada objek plat mobil lebih banyak dan bervariasi, oleh sebab itu ketika dilakukan proses uji coba akurasi pada plat mobil akan lebih baik daripada plat motor.

- Kondisi pengambilan video seperti jarak pengambilan video, cuaca bahkan perangkat keras yang digunakan. Ketika proses *training*, data yang digunakan sekitar 80% pada kondisi cerah dan 20% pada kondisi agak gelap, sehingga akurasi plat nomor yang terdetksi akan lebih

baik jika dilakukan pada konidisi yang cerah.



Gambar 5.8: Deteksi Plat Nomor Tanpa Karakter

Pada Gambar 5.8 ditunjukkan bahwa hasil model jaringan yang telah dibuat dapat mendeteksi plat nomor tanpa adanya karakter yg muncul pada plat nomor tersebut. Hal ini disebabkan karena pada proses *training* acuan pendefinisian plat nomor berdasarkan bentuk objek plat yaitu persegi panjang. Sehingga ketika ada plat nomor tanpa karakter, sistem tetap melakukan pendeteksian pada plat nomor tersebut.

5.2.5 Hasil Uji Coba dan Pembahasan Pendefinisian ROI

Pada bagian ini akan dilakukan uji coba terhadap penggunaan ROI berukuran besar dan kecil pada sebuah rekaman video kendaraan bergerak. Pada uji coba ini, ukuran ROI (1820×500) dikategorikan sebagai ROI berukuran besar. Sedangkan ukuran ROI (1820×200) dikategorikan sebagai ROI berukuran kecil. Adapun ilustrasi ukuran kedua ROI tersebut ditunjukkan pada Gambar 5.10.

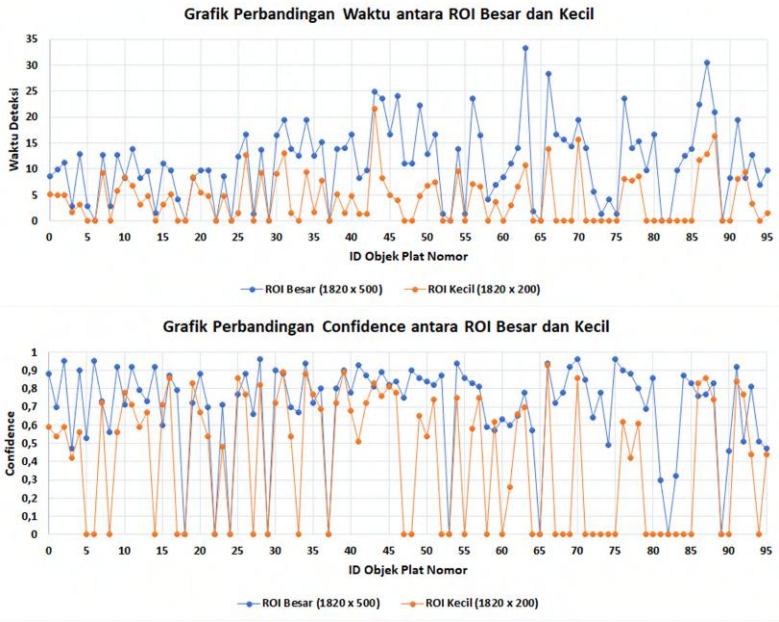
(a.) ROI Berukuran (1820 x 500)

(b.) ROI Berukuran (1820 x 200)

Gambar 5.9: Ilustrasi Perbandingan ROI (a.) dan (b.)

Pada gambar 5.9 merupakan perbedaan antara ROI besar dan kecil yg dipakai dalam uji coba. Berdasarkan uji coba tersebut didapatkan perbandingan antara ROI besar dan kecil terhadap waktu dan nilai *confidence* dari tiap objek. Hasil

perbandingan tersebut dapat dilihat pada Gambar 5.10.



Gambar 5.10: Hasil Perbandingan antara ROI besar dan ROI Kecil

Pada Gambar 5.10 dapat disimpulkan bahwa semakin besar ROI yang digunakan semakin tinggi juga nilai *confidence* yang diperoleh tetapi, dalam pengolahannya membutuhkan waktu yang lama. Hal ini dikarenakan jika ROI yang digunakan berukuran besar maka objek pembanding yang berhasil terdeteksi pada ROI semakin banyak sehingga nilai *confidence* yang didapatkan memiliki banyak variasi, akan tetapi membutuhkan waktu yang lama dalam menentukan *confidence* tertingginya. Namun, jika ROI yang digunakan berukuran kecil maka objek pembanding yang berhasil

terdeteksi pada ROI semakin sedikit sehingga nilai *confidence* yang didapatkan memiliki sedikit variasi dan butuh waktu yang cepat dalam menentukan *confidence* tertingginya. Pada Gambar 5.10 didapatkan rata - rata waktu yang dibututhkan antara ROI besar dan kecil adalah 11.391 dan 4.081 detik. Sedangkan rata - rata nilai *confidence* yang didapatkan antara ROI besar dan kecil adalah 0.696 dan 0.41.

BAB VI
PENUTUP

Pada bab ini berisi kesimpulan yang diperoleh dari pembahasan pada bab sebelumnya serta saran untuk pengembangan penelitian selanjutnya.

6.1 Kesimpulan

Berdasarkan analisis terhadap hasil pengujian program dapat diambil kesimpulan sebagai berikut :

1. Dari hasil uji coba data valid, didapatkan bahwa YoloV3 dalam melakukan uji coba model *training* terhadap data valid didapatkan *Mean Average Precision* (mAP) sebesar 87.89% dan *loss*-nya adalah 0.1077

2. Dari hasil uji coba pendeteksian plat nomor kendaraan bergerak, telah terbukti bahwa metode YoloV3 mampu mendeteksi plat nomor dengan rata-rata *recall* sebesar 86.18%. Rata-rata nilai *precision* sebesar 93.83%. Rata-rata IOU akurasi sebesar 73.43%.

3. Berdasarkan uji coba pada pendefinisian ROI besar dan kecil, didapatkan untuk penggunaan ROI besar tingkat rata- rata nilai *confidence* lebih besar yaitu 0.696 dan ROI kecil memiliki tingkat rata-rata nilai *confidence* lebih kecil yaitu 0.41. Sedangkan dalam segi waktu yang digunakan untuk menentukan lokasi plat, rata-rata waktu yang digunakan pada ROI besar jauh lebih lama yaitu 11.391 detik dibandingkan dengan ROI kecil yaitu 4.081 detik.

4. Berdasarkan uji coba pendeteksian lokalisasi plat nomor kendaraan bergerak, didapatkan hasil deteksi terhadap plat nomor yang terlihat secara empiris memiliki rata-rata akurasi sebesar 85.81%

5. Berhasil dikembangkan program yang mampu mendeteksi plat nomor kendaraan yang melintasi jalan raya melalui video dengan langkah - langkah yaitu akuisisi video, pendefinisian ROI, deteksi objek plat, *tracking*, mengambil deteksi dengan *confidence* terbesar dari tiap frame pada area ROI dan *cropping* plat nomor berdasarkan parameter *bounding box* deteksi

6.2 Saran

Adapun beberapa saran yang perlu diperhatikan diantaranya :

1. Jika melakukan penelitian yang serupa atau berbasis *deeplearning*, disarankan untuk memperbanyak lagi data *training* dalam berbagai kondisi. Sehingga ketika dilakukan pengujian, model yang dibuat juga bisa mendeteksi dalam berbagai kondisi.

2. Memakai perangkat keras GPU agar dapat meningkatkan performansi dan kecepatan program terutama ketika melakukan deteksi secara *realtime*.

3. Melakukan penelitian selanjutnya mengenai pengenalan karakter plat nomor, sehingga dari hasil deteksi dapat menjadi sebuah informasi berupa teks.

DAFTAR PUSTAKA

[1] Hui Li, Chunhua Shen, "Reading Car License Plates Using Deep Convolutional Neural Networks and LSTMs," arXiv:1601.05610v1[cs.CV], 2016.

[2] Y. Jamtsho, P. Riyamongkol and R. Waranusast, "Real-time Bhutanese license plate localization using YOLO," ICT Express, https://doi.org/10.1016/j.icte.2019.11.001, 2019.

[3] R. Laroca, E. Severo, L. A. Zanlorensi, L. S. Oliveira, G. R. Gonçalves, W. R. Schwartz and D. Menotti, "A robust real-time automatic license plate recognition based on the YOLO detector," International Joint Conference on Neural Networks (IJCNN), pp. 1-10, 2018/10/15.

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

[5] Joseph Redmon, Ali Farhadi: YOLOv3: An Incremental Improvement. CoRR abs/1804.02767 (2018)

[6] D. R. S. D. O. I. M. B. S. a. E. A. Ravy Hayu Pramestya, "Pavement Distress Classification Using Deep Learning Method Based on Digital Image," Advances in Engineering Research, vol. 193, 2019.

[7] Ardiansyah, Wahyu, "Peningkatan Performansi Metode Harris Corner Untuk Deteksi Plat Nomor Pada Video Menggunakan Maximally Stable External Regions," Institut Teknologi Sepuluh Nopember, Surabaya, 2019.

[8] Wang, Q., Rasmussen, C. dan Song, C.., "Fast Deep Detection and Tracking of Birds anad Nests," International Symposium on Visual Computing, no. Springer , pp. 146-155, 2016.

[9] RD. Kusumanto, Alan Novi Tompunu, "Pengolahan Citra Digital Untuk Mendeteksi Obyek Menggunakan Pengolahan Warna Model Normalisasi RGB," in Seminar Nasional Teknologi Informasi & Komunikasi Terapan 2011, Semantik, 2011.

[10] PMJ, A, "Agar Tidak Ditilang Karena Masalah Plat Nomor," TribrataNews, 21 August 2017. [Online]. Available: http://tribratanews.polri.go.id/?p=249778. [Accessed 20 September 2019].

[11] Everingham, M., Van Gool, L., Williams, C. K., Winn, J. dan Zisserman, A, "The Pascal Visual Object Classes (VOC) Challenge," International Journal of Computer Vision, vol. 2, no. 88, pp. 303-338, 2010.

[12] Zitnick, C. L. dan Dollar, P., "Edge boxes: Location Object Proposals From Edge," in European Conference on Computer Vision, Springer, 2014.

[13] Geethapriya. S, N. Duraimurugan, S.P. Chokkalingam, "Real-Time Object Detection with Yolo," International Journal of Engineering and Advanced Technology (IJEAT), vol. 8, no. 3S, pp. 2249-8958, 2019.

[14] I Wayan Suartika E. P, Arya Yudhi Wijaya, dan Rully Soelaiman, "Klasifikasi Citra Menggunakan Convolutional Neural Networks (CNN) pada Caltech 101," JURNAL TEKNIK ITS , vol. 5, no. 2301-9271 Print, pp. 2337-3539, 2016.

[15] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," Information Processing dan Management, vol. 45, no. 4, pp. 427.

[16] S. I. a. C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," CoRR, vol. abs/1502.03167, 2015.

BIODATA PENULIS

Nama lengkap penulis Ario Fajar Pratama atau biasa akrab diapnggil Ario, lahir di Banyuwangi pada tanggal 01 Januari 1998. Pendidikan formal yang pernah ditempuh yaitu SD Negeri 9 Jambewangi, SMP Negeri 2 Genteng dan MAN 2 Banyuwangi. Sekarang penulis menempuh pendidikan S1 di Departemen Matematika Fakultas Sains dan Analitika Data Institut Teknologi Sepuluh Nopember Surabaya dengan bidang minat Pemrograman dan Komputasi Visual. Selama kuliah, penulis aktif di dalam organisasi HIMATIKA ITS. Pada tahun 2017 – 2018 penulis menjadi Staff PSDM UKM PMI ITS, Staff Community Service HIMATIKA ITS dan menjadi pengajar tangguh pada event ITS Mengajar for Indonesia. Pada tahun 2018 – 2019 penulis menjadi Ketua Himpunan Mahasiswa Matematika ITS. Penulis juga merupakan salah satu santri Pondok Darussalam Keputih Sukolilo Surabaya dan menjadi koordinator divisi Ubudiyah. Demikian biodata tentang penulis. Jika ingin memberikan saran, kritik, dan diskusi mengenai laporan tugas akhir ini, dapat dikirimkan melalui email ariofajarpratama@gmail.com. Terimakasih