TUGAS AKHIR - KM184801

PENERAPAN METODE YOU ONLY LOOK ONCE (YOLO) UNTUK PENGENALAN KERUSAKAN JALAN DARI DATA VIDEO

ALVARO BASILY SUPRIYANTO
NRP 06111640000123

Dosen Pembimbing:
Dr. Dwi Ratna Sulistyaningrum, S.Si, MT

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

FINAL PROJECT - KM4801

APPLICATION OF YOU ONLY LOOK ONCE (YOLO) METHOD FOR ROAD DAMAGE RECOGNITION FROM VIDEO DATA

ALVARO BASILY SUPRIYANTO
NRP 06111640000123

Supervisors:
Dr. Dwi Ratna Sulistyaningrum, S.Si, MT

DEPARTMENT OF MATHEMATICS
Faculty of Science and Data Analytics
Institut Teknologi Sepuluh Nopember
Surabaya 2020

LEMBAR PENGESAHAN

PENERAPAN METODE YOU ONLY LOOK ONCE (YOLO) UNTUK PENGENALAN KERUSAKAN JALAN DARI DATA VIDEO

APPLICATION OF YOU ONLY LOOK ONCE (YOLO) METHOD FOR ROAD DAMAGE RECOGNITION FROM VIDEO DATA

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
Untuk memperoleh gelar Sarjana Matematika
Pada bidang studi Analisis dan Aljabar
Program Studi S1 Departemen Matematika
Fakultas Matematika, Komputasi, dan Sains Data
Institut Teknologi Sepuluh Nopember Surabaya

Oleh:
ALVARO BASILY SUPRIYANTO
NRP. 06111640000123

Menyetujui,
Dosen Pembimbing

Dr. Dwi Ratna Sulistyaningrum, S.Si, MT
NIP. 19690405 199403 2 003

Mengetahui
Kepala Departemen Matematika
FSAD ITS

Subchan, Ph.D
NIP. 19710513 199702 1 001
Surabaya, 19 Agustus 2020

PENERAPAN METODE *YOU ONLY LOOK ONCE* (YOLO) UNTUK PENGENALAN KERUSAKAN JALAN DARI DATA VIDEO

Nama Mahasiswa	:	Alvaro Basily Supriyanto
NRP	:	06111640000123
Jurusan	:	Matematika FSAD-ITS
Pembimbing	:	Dr. Dwi Ratna Sulistyaningrum, S.Si, MT

Abstrak

Dalam melakukan pemeliharaan jalan, diperlukan data kondisi jalan untuk menentukan keputusan berikutnya dalam melakukan pemeliharaan jalan. Untuk mendapatkan data tersebut diperlukan suatu mekanisme kerja yang efektif dan efisien dalam mengambil data tersebut. Salah satu cara yang dapat membantu petugas dalam mengambil data kondisi jalan dengan menggunakan suatu program pengenalan jenis kerusakan jalan. Pada Tugas Akhir ini telah dilakukan pengenalan kerusakan jalan dari data video dengan menggunakan metode You Only Look Once (YOLO) yang merupakan salah satu metode deep learning yang memiliki keunggulan kecepatan dalam melakukan pengenalan objek. Terdapat tiga tahapan umum yang dilakukan pada Tugas Akhir ini. Tahapan pertama adalah mengumpulkan data kerusakan jalan dan memberikan anotasi, tahapan kedua adalah melakukan training model sehingga model dapat mengenali kerusakan jalan, dan tahapan terakhir adalah melakukan uji coba untuk mengetahui kinerja dari metode You Only Look Once dalam mengenali kerusakan jalan. Dari hasil uji coba didapatkan nilai mean average precision (mAP) tertinggi sebesar 79.06% pada 665 citra dengan 1437 kerusakan jalan.

Kata Kunci *You Only Look Once, Pengenalan Kerusakan Jalan, Deep Learning*

APPLICATION OF YOU ONLY LOOK ONCE (YOLO) METHOD FOR ROAD DAMAGE RECOGNITION FROM VIDEO DATA

Name	:	Alvaro Basily Supriyanto
NRP	:	06111640000123
Department	:	Mathematics FMCDS-ITS
Supervisor	:	Dr. Dwi Ratna Sulistyaningrum, S.Si, MT

Abstract

For road maintenance, road condition data is needed to determine the next decision to maintain the road and to obtain an effective and efficient data method for collecting data is needed. One way that operators can help to collect road data conditions is to use a damaged road recognition program. In this studied, the recognition of road damage from video data has been done using You Only Look Once (YOLO) which is one of the object detection methods who has speed advantages. There are three stages that have been done in this studied. The first stage is collecting the road damage data and giving annotation to the data, the second stage is to train the model these the model can recognition road damage, and the third stage is testing the model to measure the performance this model in recognition road damage. From testing stages shows the highest average precision (mAP) is 79.06% from 665 image with 1437 road damage.

Keywords *You Only Look Once, Road Damage Recognition, Deep Learning*

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT yang telah memberikan banyak nikmat sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul

PENERAPAN METODE *YOU ONLY LOOK ONCE* (YOLO) UNTUK PENGENALAN KERUSAKAN JALAN DARI DATA VIDEO

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

Dalam penyusunan Tugas Akhir ini dapat diselesaikan berkat bantuan dan dukungan dari berbagai pihak. Oleh karena itu penulis ingin menyampaikan ucapan terima kasih yang setinggi-tingginya kepada:

1. Subchan, Ph.D selaku Kepala Departemen Matematika Institut Teknologi Sepuluh Nopember

2. Dr. Dwi Ratna Sulistyaningrum, S.Si, MT selaku Dosen Pembimbing yang telah memberikan bimbingan, dan waktu luangnya dalam mengerjakan Tugas Akhir.

3. Dr. Budi Setiyono, S.Si, MT, Mohammad Iqbal, S.Si, M.Si, dan Drs. Iis Herisman, M.Si selaku Dosen Penguji yang telah memberikan masukan-masukan.

4. Seluruh jajaran dosen dan staf Jurusan Matematika ITS yang sudah memberikan ilmu dan bantuan selama menjalani studi di Departemen Matematika ITS.

5. Bapak, Ibu, dan seluruh keluarga yang sudah memberikan banyak motivasi, pengorbanan materil maupun immateril selama penulis menumpuh studi di

Departemen Matematika Institut Teknologi Sepuluh Nopember

6. Teman-teman Matematika ITS Angkatan 2016 yang sudah memberikan inspirasi dan cerita selama perkuliahan.

7. Pihak-pihak lain yang turut membantu penulis untuk menyelesaikan Tugas Akhir.

Pada Tugas Akhir ini tentunya masih terdapat kekurangan-kekurangan, oleh karena itu penulis sangat berharap atas kritik dan saran dari pembaca sehingga Tugas Akhir dapat lebih baik lagi. Penulis sangat berharap Tugas Akhir ini dapat memberikan manfaat kepada pembaca.

Surabaya, 20 Juni 2020

Penulis

DAFTAR ISI

HALAMAN JUDUL i

LEMBAR PENGESAHAN v

ABSTRAK vii

ABSTRACT ix

KATA PENGANTAR xi

DAFTAR ISI xiii

DAFTAR GAMBAR xvii

DAFTAR TABEL xxi

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 Tugas Akhir 5

BAB II TINJAUAN PUSTAKA 8
- 2.1 Penelitian Terdahulu 8
- 2.2 Jalan 10
  - 2.2.1 Jenis Kerusakan Jalan 11
- 2.3 Pengolahan Citra Digital 15
- 2.4 *Deep Learning* 18
- 2.5 *Convolutional Neural Network* 19
- 2.6 *You Only Look Once* (YOLO) 21

2.6.1 Cara Kerja *You Only Look Once* (YOLO) 22
2.6.2 Arsitektur YOLOv3 24
2.6.3 *Anchor Box* 27
2.6.4 *Lost Function* 28
2.7 *Batch Normalization* 29
2.8 *Confusion Matrix* 30
2.8.1 *Intersection Over Union* (IoU) 33

BAB III METODE PENELITIAN 35
3.1 Tahapan Penelitian 35
3.1.1 Studi literatur 35
3.1.2 Pengambilan Data dan Persiapan Data 35
3.1.3 *Training* Model Pengenalan Kerusakan Jalan 37
3.1.4 Implementasi Program 37
3.1.5 Uji Coba 37
3.1.6 Penarikan Kesimpulan 37
3.1.7 Penulisan Laporan Tugas Akhir 38
3.2 Diagram Blok Pengenalan Kerusakan Jalan Dengan Metode YOLOv3 38

BAB IV PERANCANGAN DAN IMPLEMENTASI 42
4.1 Pengambilan Data dan Persiapan Data 42
4.1.1 Pengambilan Data 42
4.1.2 Persiapan Data Latih 43
4.1.3 Pembagian Data Latih 44
4.2 Penentuan *Anchor Box* 46
4.3 *Training* Model Pengenalan Kerusakan Jalan 47
4.3.1 Arsitektur Jaringan 47
4.3.2 Parameter *Training* 63
4.4 Implementasi Program 64
4.4.1 Implementasi Program Pengambilan Data dan Persiapan Data 64

4.4.2 Implementasi Program *Training* Model Pengenalan Kerusakan Jalan ... 67
4.4.3 Implementasi Program Pengenalan Kerusakan Jalan ... 70
4.4.4 Implementasi Program *Interface* ... 72

BAB V UJI COBA DAN PEMBAHASAN 75
5.1 Pengujian *Trained Weight* ... 75
5.1.1 Data Pengujian *Trained Weight* ... 75
5.1.2 Skenario Tahap Pelatihan Model Pengenalan Jalan ... 76
5.1.3 Hasil *Training* Beserta Pengujiannya ... 77
5.2 Pengujian YOLOv3 Dengan Data Video ... 82
5.2.1 Data Pengujian YOLOv3 Dengan Data Video ... 82
5.2.2 Skenario Pengujian Metode YOLOv3 Dengan Data Video ... 84
5.2.3 Hasil Pengujian Metode YOLOv3 Dengan Data Video ... 86

BAB VI PENUTUP 94
6.1 Kesimpulan ... 94
6.2 Saran ... 95

DAFTAR PUSTAKA 97

LAMPIRAN 100

BIODATA PENULIS 107

DAFTAR GAMBAR

Gambar 2.1 Contoh retak pada jalan 12
Gambar 2.2 Contoh distorsi pada jalan [1] 13
Gambar 2.3 Contoh cacat permukaan pada jalan . . . 13
Gambar 2.4 Contoh pengausan pada jalan [1] 14
Gambar 2.5 Contoh kegemukan pada jalan [1] 14
Gambar 2.6 Contoh penurunan pada bekas penanaman utilitas pada jalan [1] 15
Gambar 2.7 Ilustrasi bagaimana proses pengolahan citra pada domain spasial [2]. 17
Gambar 2.8 Tiga layer dari *feedforward neural network* (FNN) sederhana, terdiri dari *input layer*, *hidden layer*, dan *output layer*[3]. 20
Gambar 2.9 Arsitektur CNN sederhana, terdiri dari lima layer[3]. 21
Gambar 2.10 Ilustrasi Bagaimana YOLO bekerja [4]. 23
Gambar 2.11 Ilustrasi *Non-Max Suppression* dalam menghilangkan kotak pembatas *(bounding box)* yang tidak diperlukan. . 25
Gambar 2.12 Arsitektur YOLO v3 26
Gambar 2.13 Ilustrasi *Convolutional Layer* pada jaringan YOLOv3 27
Gambar 2.14 Ilustrasi dari *Residual Unit* 27
Gambar 2.15 Ilustrasi *Anchor Box.*[5] 28
Gambar 2.16 Ilustrasi dari *Intersection Over Union (IoU)* 33

Gambar 3.1 Diagram Metode Penelitian 36

Gambar 3.2 Diagram Blok Pengenalan Kerusakan Jalan Dengan Metode YOLOv3 40

Gambar 4.1 Posisi *smartphone* di mobil saat melakukan pengambilan data. 43
Gambar 4.2 Total Kerusakan Jalan Berdasarkan Jenisnya 44
Gambar 4.3 Ilustrasi citra yang sudah diberikan anotasi dengan format `<id-kelas> <x-pusat> <y-pusat> <panjang> <lebar>` 45
Gambar 4.4 Ilustrasi kluster yang dihasilkan dengan *k-means clustering* pada data berukuran 416×416 (Kiri) dan data berukuran 512×512 (Kanan). 47
Gambar 4.5 Ilustrasi proses *convolution* untuk mendapatkan nilai pada kolom 1 dan baris 1 52
Gambar 4.6 Ilustrasi proses *convolution* untuk mendapatkan nilai pada kolom 2 dan baris 1 53
Gambar 4.7 Ilustrasi *Residual Unit* 60
Gambar 4.8 Diagram Alir Dari Program Pengenalan Kerusakan Jalan 71
Gambar 4.9 *Interface* dari program pengenalan kerusakan jalan 73

Gambar 5.1 Jumlah data uji berdasarkan jenis kerusakan 76
Gambar 5.2 Grafik nilai *loss* dan *mAP* pada skenario pertama 77
Gambar 5.3 Grafik nilai *loss* dan *mAP* pada skenario kedua 78

Gambar 5.4 Contoh data yang digunakan dari setiap skenario 85
Gambar 5.5 Contoh dari Hasil Pengenalan Kerusakan Jalan Dengan YOLOv3 Pada Skenario 1 86
Gambar 5.6 Grafik nilai *f1-score* dari setiap skenario 89
Gambar 5.7 Contoh Terjadinya Pengenalan Ganda Terhadap Satu Objek Kerusakan Jalan 91
Gambar 5.8 Contoh Permasalahan Saat Melakukan Pengenalan Kerusakan *Alligator Crack* 92
Gambar 5.9 Contoh Kerusakan *Lateral Crack* Yang Gagal Dihitung 92

DAFTAR TABEL

Tabel 4.1 Arsitektur YOLOv3 Secara Detail 48
Tabel 4.1 Lanjutan Arsitektur YOLOv3 Secara Detail 49
Tabel 4.2 Parameter *Training* YOLOv3 63

Tabel 5.1 Hasil Pengujian *Trained Weight* Dengan Parameter $Input = (416 \times 416)$ dan $Threshold = 0.5$ 79
Tabel 5.2 Hasil Pengujian *Trained Weight* Dengan Parameter $Input = (512 \times 512)$ dan $Threshold = 0.5$ 80
Tabel 5.3 Hasil Pengujian *Trained Weight* Dengan $Threshold = 0.5$ dan $Threshold = 0.3$ . . . 81
Tabel 5.4 Jumlah Kerusakan Yang Terdapat Pada Setiap Video Uji 83
Tabel 5.5 Spesifkasi *Hardware* Dari Setiap Kondisi . 85
Tabel 5.6 Hasil Pengujian Metode YOLOv3 Dari Setiap Skenario Dengan Ukuran Input 416×416 87
Tabel 5.7 Hasil Pengujian Metode YOLOv3 Dari Setiap Skenario Dengan Ukuran Input 512×512 88
Tabel 5.8 Hasil Perhitungan Kerusakan dan Kecepatan Deteksi YOLOv3 90

Lampiran 1 Hasil Pengujian Metode YOLOv3 Pada Video 1 101

Lampiran 2Hasil Pengujian Metode YOLOv3 Pada Video 2 102
Lampiran 3Hasil Pengujian Metode YOLOv3 Pada Video 3 103
Lampiran 4Hasil Pengujian Metode YOLOv3 Pada Video 4 104
Lampiran 5Hasil Pengujian Metode YOLOv3 Pada Video 5 105
Lampiran 6Hasil Pengujian Metode YOLOv3 Pada Video 6 106

BAB I
PENDAHULUAN

Bab ini menjelaskan alasan penulisan tugas akhir kedalam suatu latar belakang. Kemudian didapatkan rumusan masalah yang ingin ditinjau lebih lanjut, tujuan yang ingin dicapai dan manfaat yang diharapatkan didapatkan dalam penulisan tugas akhir. Selain itu terdapat beberapa batasan masalah yang digunakan dalam pengerjaan tugas akhir ini.

1.1 Latar Belakang

Perkembangan zaman menuntut pemerintah untuk membangun dan melakukan pemeliharaan infrastruktur. Menurut *World Economy Forum* dalam laporannya yang berjudul "*The Global Competititvenes Report 2019*", infrastruktur menjadi salah satu sektor penting yang dapat mempengaruhi kesiapan suatu negara dalam menghadapi persaingan global[6]. Hal tersebut dirasa wajar dikarenakan infrastruktur yang baik dapat memudahkan masyarakat untuk melakukan mobilisasi secara aman dan nyaman, membantu masyarakat dalam menjangkau fasilitas-fasilitas sosial seperti rumah sakit ataupun sekolah, menekan harga logistik, hingga aksesbilitas listrik di daerah pedalaman. Oleh karena itu, menjaga dan meningkatkan infrastruktur merupakan salah satu tugas yang harus dilaksanakan pemerintah.

Salah satu infrastruktur yang penting adalah jalan, jalan adalah prasarana transportasi darat yang meliputi segala bagian jalan, termasuk bangunan pelengkap dan pelengkapnya yang diperuntukan bagi lalu lintas, yang berada pada permukaan tanah, di atas permukaan tanah, di bawah

permukaan tanah dan/atau air, serta di atas permukaan air, kecuali jalan kereta api, jalan lori, dan jalan kabel[7]. Namun yang masih disayangkan adalah kualitas jalan yang ada di Indonesia berada di posisi ke-60, masih berada di bawah Malaysia (19), Thailand (55). Posisi tersebut dirasa sangat wajar jika melihat persentase kondisi jalan yang dapat dikatakan tidak mantap (kondisi jalan tidak dalam keaadan baik dan sedang) di jalan nasional sebesar 10,62%, jalan provinsi sebesar 30,18%, dan jalan kabupaten sebesar 43,07 % [7]. Dengan kata lain, kondisi jalan di Indonesia masih memerlukan perhatian khusus karena selain salah satu fasilitas inti dalam mobilisasi di masyarakat.

Berdasarkan prosedur pemeliharaan jalan, dalam melakukan pemeliharaan jalan terdapat tiga tahapan yaitu perencanaan, pelaksanaan, dan evaluasi pelaksaan. Pada tahap perencanaan, terdapat beberapa dokumen yang diperlukan dan salah satunya adalah data kondisi jalan[8].Untuk mendapatkan data kondisi jalan, diperlukan suatu mekanisme kerja yang efektif dan efisien. Jika dalam mendapatkan data kondisi jalan tidak dilakukan secara efektif dan efisien dapat mengganggu lalu lintas. dalam menciptakan suatu sistem yang efektif dan efisien dapat dibantu dengan kemampuan kecerdasan buatan komputer untuk membantu menyelesaikan permasalahan tersebut. Secara sederhana sistem tersebut harus mampu menganilisa jenis kerusakan jalan dan menghasilkan output berupa informasi jenis kerusakan hingga lokasi kerusakan.

Beberapa penelitian yang berkaitan dengan sistem deteksi kerusakan jalan seperti penelitian yang berjudul "*Detecting Potholes Using Simple Image Processing Techniques and Real World Footage*" oleh S Nienaber, M Booysen dan R Kroon (2015). Penelitian ini melakukan deteksi terhadap kerusakan lubang (*Pothole*) pada jalan dengan pendekatan

image processing untuk melakukan klasifikasi lubang jalan dengan pengelohan citra secara sederhana [9]. Selain deteksi kerusakan jalan dengan pendekatan *image processing*, perkembangan jaman juga ikut memadukan deteksi kerusakan jalan dengan *deep learning*. Penggunaan *deep learning* memiliki kelebihan pada nilai akurasi yang jauh lebih besar dan dapat bekerja dengan data yang lebih bervariatif karena *deep learning* bergantung dengan *train data* yang kita berikan. Walaupun menawarkan banyak kelebihan, tentunya banyak hal yang harus dipertaruhkan dalam penggunaan *deep learning* seperti beban komputasi yang harus dikerjakan jauh lebih besar dari penggunaan *computer vision* secara tradisional [10]. Beberapa penelitian yang menggunakan *deep learning* seperti penelitian yang dilakukan oleh Hiroya Maeda, Yoshihide Sekomot, Toshikazu Seto, Takehiro Kashiyama, dan Hiroshi Omata dengan judul *"Road Damage Detection and Classification Using Deep Neural Networks with Smartphone Images: Road damage detection and classification"*. Penelitian ini melakukan pendeteksian beberapa jenis kerusakan jalan melalui sebuah smartphone dengan *Single Shot MultiBox Detextor* (SSD) [11]. Kemudian penelitian yang dilakukan oleh Young-Jin Cha, Wooram Choi, dan Oral Büyüköztürk dengan judul *"Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks"*. Penelitian ini melakukan pendeteksian kerusakan jalan dengan *convulational neural networks* (CNN)[12]. *You Only Look Once* atau yang biasa dikenal YOLO diperkenalkan pada tahun 2015. YOLO diklaim unggul dalam permasalahan *real-time object detection* karena kemampuannya yang dapat melakukan pengenalan objek hingga 45 *frame/s* [4].

Atas dasar tersebut penulis tertarik untuk melakukan penelitian terkait pengaplikasian metode *You Only Look Once* (YOLO) untuk pengenalan kerusakan jalan melalui input

video. penggunaan YOLO dalam penulisan tugas akhir ini dipilih berdasarkan keunggulannya dalam permasalahan *real-time object detection.*

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut disusun rumusan masalah pada penulisan tugas akhir:

1. Bagaimana melakukan pengenalan kerusakan jalan dari data video dengan menggunakan metode *you only look once*?

2. Bagaimana kinerja metode *you only look once* dalam melakukan pengenalan kerusakan jalan?

1.3 Batasan Masalah

Batasan masalah yang digunakan dalam penulisan tugas akhir sebagai berikut:

1. Input dalam bentuk file video.

2. Pengambilan video diarahkan searah dengan jalan.

3. Video diambil ketika pencahayaan dalam kondisi yang baik

4. Hanya kerusakan jalan yang ditemukan pada jalan dengan perkerasan lentur (aspal)

5. Kerusakan yang dikenali yaitu Retak Garis Memanjang (*Longitudinal Cracking*), Retak Garis Melintang (*Lateral Cracking*), Retak Kulit Buaya (*Alligator Crack*), dan Lubang (*Potholes*)

1.4 Tujuan

Berdasarkan rumusan masalah, didapatkan tujuan dari penulisan Tugas Akhir sebagai berikut:

1. Melakukan pengenalan kerusakan jalan dengan menggunakan metode *You Only Look Once* (YOLO).

2. Mengetahui kinerja metode *you only look once* (YOLO) dalam menghadapi permasalahan pengenalan kerusakan jalan.

1.5 Manfaat

Manfaat yang diharapkan bisa diperoleh dari Tugas Akhir sebagai berikut:

1. Untuk membantu pihak terkait (Dinas Pekerjaan Umum dan Perumahan Rakyat, Pemerintah daerah) dalam melakukan inspeksi kerusakan jalan secara otomatis dengan harapannya dapat membantu pengambilan keputusan dalam pemeliharaan jalan tersebut.

2. Sebagai refrensi untuk penetliatan dimasa mendatang.

1.6 Sistematika Penulisan Tugas Akhir

Sistematika penulisan dalam laporan Tugas Akhir sebagai berikut:

BAB I: PENDAHULUAN
Bab ini menjelaskan latar belakang pada penulisan tugas akhir, dilanjutkan dengan rumusan masalah, batasan masalah, tujuan dan manfaat yang ingin didapatkan dalam penulisan Tugas Akhir.
BAB II: TINJAUAN PUSTAKA
Bab ini berisikan mengenai dasar-dasar teori yang dapat mendukung dalam penulisan tugas akhir ini. Dasar-dasar teori yang digunakan antara lain

penjelasan mengenai jalan beserta jenis kerusakananya, pengolahan citra digital, *deep learning*, *Convolutional Neural Network*, *you only look Once (yolo)*, *batch normalization*, *confusion matrix*, *intersection over union*.

BAB III: METODOLOGI PENELITIAN

Bab ini menjelaskan tentang metodologi penelitian yang digunakan dalam penulisan tugas akhir. Metodologi Penelitian diperlukan untuk patokan utama dalam pengerjaan tugas akhir agar pengerjaan menjadi lebih teratur.

BAB IV: PERANCANGAN DAN IMPLEMENTASI

Bab ini menjelaskan bagaimana metode *You Only Look Once* dipersiapkan untuk dapat melakukan pengenalan terhadap kerusakan jalan. Dimulai dari proses mempersiapkan data kerusakan jalan hingga implementasi metode *You Only Look Once* kedalam suatu program.

BAB V: UJI COBA

Bab ini menjelaskan mengenai pengujian metode *You Only Look Once* dalam melakukan pengenalan terhadap kerusakan jalan. Pengujian diperlukan guna mengetahui seberapa baik kinerja metode *You Only Look Once* yang sudah dipersiapkan sebelumny.

BAB VI: PENUTUP

Pada bab ini berisikan kesimpulan yang diperoleh dalam pengerjaan Tugas Akhir. Selain kesimpulan, diberikan saran-saran yang diharapkan dapat menjadi masukan untuk pengembangan penelitian selanjutnya.

Halaman ini sengaja dikosongkan

BAB II
TINJAUAN PUSTAKA

Dalam penulisan tugas akhir, perlu dijelaskan dasar-dasar teori yang dapat menunjang dalam menyelesaikan masalah dalam tugas akhir ini.

2.1 Penelitian Terdahulu

Terdapat beberapa penelitian yang berkaitan dengan topik tugas akhir. Seperti penelitian yang dilakukan oleh S Nienaber, M Booysen dan R Kroon dengan judul "*Detectiong Potholes Using Simple Image Processing Techniques and Real World Footage*". Pada penelitian tersebut dibahas deteksi kerusakan jalan dengan pendekatan *image processing* sederhana dalam melakukan pengenalan kerusakan lubang jalan (*potholes*). Pada penelitian tersebut dalam melakukan deteksi lubang jalan digunakan teknik *image processing* seperti *canny filter* dan deteksi kontur. Dengan memanfaatkan teknik tersebut didapatkan hasil presisi sebesar 81.8% dan *recall* sebesar 74.4% [9]. Selain dengan pemanfaatan *image processing*, beberapa penelitian melakukan deteksi dan pengenalan kerusakan jalan dengan pendekatan *deep learning*. Seperti yang dilakukan oleh Hiroya Maeda, Yoshihide Sekomot, Toshikazu Seto, Takehiro Kashiyama, dan Hiroshi Omata dengan judul *"Road Damage Detection and Classification Using Deep Neural Networks with Smartphone Images: Road damage detection and classification"*. Pada penelitian tersebut dibangun suatu pendeteksi dan pengenalan kerusakan jalan dengan metode*convolutional neural networks* untuk melakukan

train data dengan delapan kelas berbeda. Kemudian membandingkan nilai *accuracy* dan *runtime speed* antara SSD Inception V2 dan SSD MobileNet dengan menggunakan server GPU dan *smarthphone*[11]. Kemudian penelitian yang dilakukan oleh Young-Jin Cha, Wooram Choi, dan Oral Büyüköztürk dengan judul *Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks*. Dalam penelitian tersebut dilakukan pendeteksian dan pengenalan keusakan keretakan jalan dengan memanfaatkan *convolutional neural network*. Dari penelitian tersebut tercatat nilai akurasi hingga 98.22% dari 32 K citra dan mendapatkan akurasi sebesar 97.95% dari 8 K citra. Dari penelitian tersebut juga didapatkan kesimpulan bawha pengguna metode CNN memiliki kinerja yang sangat kuat jika dibandingkan dengan metode deteksi tepi yang cukup terkenal, seperti *canny filter* dan *sobel filter*[12]. Selain itu di lingkungan ITS juga terdapat penelitian yang berkaitan dengan deteksi dan pengenalan kerusakan jalan. Seperti penelitian yang dilakukan oleh Syifa Laili Hapsari Oktavian dengan judul "Penerapan Metode *Faster Regional - Convolutional Neural Network (Faster R-CNN)* Untuk Klasifikasi Jenis Kerusakan Jalan" yang melakukan klasifikasi jenis kerusakan jalan dengan metode *Faster R-CNN* dari data citra. Dari penelitian tersebut didapatkan rata-rata akurasi hingga 95,73%. Dengan kata lain metode tersebut dapat melakukan klasifikasi kerusakan jalan secara baik [13]. Selanjutnya adalah penelitian yang dilakukan oleh Dwi Ratna Sulistyaningrum, Ravy Hayu Pramestya, Budi Setiyono, Daniel Oranova, Imam Mukhlash, dan Ervina Ahyudanari dengan judul "Pavement Distress Classification Using Deep Learning Method Based on Digital Image ". Pada penelitian tersebut melakukan deteksi dan pengenalan kerusakan jalan dengan memanfaatkan metode YOLO, khususnya YOLOv1-

tiny, dengan *input* berupa citra [14].

2.2 Jalan

Jalan adalah prasarana transportasi darat yang meliputi segala bagian jalan, termasuk bangunan pelengkap dan perlengkapannya yang diperuntukkan bagi lalu lintas, yang berada pada permukaan tanah, di atas permukaan tanah, di bawah permukaan tanah dan/atau air, serta di atas permukaan air, keculai jalan kereta api, jalan lori, dan jalan kabel [7].

Secara fisik, jalan memiliki beberapa jenis konstruksi perkerannya yang dibagi berdasarkan bahan pengikat [8]:

1. Konstruksi perkerasan lentur (*flexible pavement*), yaitu perkerasan yang menggunakan aspal sebagai bagan pengikat. Lapisan-lapisan perkersannya bersifat memikul dan menyebarkan beban lalu lintas ke tanah dasar.

2. Konstruksi perkerasan kaku (*rigid paveent*), yaitu perkerasan yang menggunakan semen (*portland cement*) sebagai bahan pengikat. Pelat beton dengan atau tanpa tulangan diletakkan diatas tanah dasar dengan atau tanpa lapis pondasi bawah. Beban lalu lintas sebagian besar dipikul oleh pelat beton.

3. Konstruksi perkerasan komposit (*composite pavement*), yaitu perkerasan kaku yang dikombinasikan dengan perkerasan lentur.

Kerusakan sangat mungkin terjadi pada jalan seiring berjalannya waktu, kerusakan dapat disebabkan oleh beberapa faktor [8]:

1. Lalu lintas yang dapat berupa peningkatan beban, dan repetisi beban.

2. Air, yang dapat berasal dari air hujan, sistem drainase jalan yang tidak baik, naiknya air akibat sifat kapilaritas.

3. Material konstruksi perkerasan. Maksudnya adalah dapat disebabkan oleh sifat material itu sendiri atau dapat disebabkan sistem pengolahan bahan yang tidak baik.

4. Iklim, Indonesia beriklim tropis dimana memiliki suhu udara dan curah hujan yang umumnya tinggi menjadi salah satu penyebab kerusakan jalan.

5. Kondisi tanah dasar yang tidak stabil. Kemungkinan disebabkan oleh sistem pelaksanaan yang kurang baik, atau dapat juga disebabkan oleh sifat tanah dasarnya yang memang jelek.

6. Proses pemadatan lapisan di atas tanah dasar yang kurang baik.

2.2.1 Jenis Kerusakan Jalan

Menurut Manual Pemeliharaan Jalan Nomor : 03/MN/B/1983 yang dikeluarkan oleh Direktorat Jenderal Bina Marga, kerusakan jalan dapat dibedakan sebagai berikut [15]:

1. Retak (*Crack*)
   Retak dapat dibedakan kedalam beberapa kelompok, seperti retak halus (hair cracks) yang memiliki lebar celah lebih kecil atau sama dengan 3 mm. Retak kulit buaya *(alligator crack)* yang memiliki lebar celah lebih besar atau sama dengan 3 mm namun berangkai membentuk serangkaian kotak-kotak kecil yang menyerupai kulit buaya. Retak pinggir *(edge*

crack), retak memanjang jalan, dengan atau tanpa cabang yang mengarah ke bahu dan terletak dekat bahu. Retak sambungan bahu dan perkerasan *(edge joint crack)*, retak memanjang, umumnya terjadi pada sambungan bahu dengan perkerasan. Retak sambungan jalan *(lane joint crack)*, retak memanjang yang terjadi pada sambungan 2 lajur lalu lintas. Retak sambungan pelebaran jalan *(widening crack)*, retak memanjang yang terjadi pada sambungan antara perkerasan lama dengan perkerasan pelebaran. Retak refleksi *(reflection crack)*, retak memanjang, melintang, diagonal, atau membentuk kotak. Retak susut *(shrinkage crack)*, retak yang saling bersambungan membentuk kotak-kotak besar dengan sudut tajam. Retak selip *(slippage crack)*, retak yang bentuknya melengkung seperti bulan sabit.

Gambar 2.1: Contoh retak pada jalan

2. Distorsi (*Distortion*)
   Distorsi atau perubahan bentuk dapat terjadi akibat lemahnya tanah dasar, pemadatan yang kurang pada

lapis pondasi, sehingga terjadi penambahan pemadatan akibat beban lalu lintas. Distorsi dapat dibedakan atas: Alur *(ruts)*, Keriting *(corrugation)*, Sungkur *(shoving)*, Amblas *(grade depressions)*, dan Jembut *(upheavel)*

Gambar 2.2: Contoh distorsi pada jalan [1]

3. Cacat Permukaan (*Disintegration*)
   Cacat permukaan atau *disintegration* adalah bentuk kerusakan pada permukaan jalan. Adapun beberapa kerusakan yang masuk kedalam cacat kerusakan seperti lubang *(potholes)*. pelepasan butir *(raveling)*, dan pengelupasan lapisan permukaan *(stripping)*.

Gambar 2.3: Contoh cacat permukaan pada jalan

4. Pengausan (*Polished Aggregate*)
   Pengausan mengakibatkan permukaan jalan menjadi licin sehingga membahayakan lalu lintas. Pengausan terjadi karena agregat berasal dari material yang tidak tahan aus terhadap roda kendaran agregate yang digunakan berbentuk bulat dan licin, tidak berbentuk kubik.

Gambar 2.4: Contoh pengausan pada jalan [1]

5. Kegemukan (*Bleeding or Flushing*)
   Kegemukan mengakibatkan permukaan jalan menjadi licin. Berbeda dengan pengausan yang terjadi karena material permukaan jalan yang aus, pada temperatur tinggi aspal akan menjadi lunak dan akan terjadi jejak roda. Hal ini disebabkan oleh pemakaian kadar aspal yang tinggi pada campuran aspal.

Gambar 2.5: Contoh kegemukan pada jalan [1]

6. Penurunan pada bekas penanaman utilitas(*utility cut depression*)
   Penurunan pada bekas penanaman utilitas terjadi karena pemadatan yang tidak memenuhi syarat. Terjadi di sepanjang bekas penanaman utilitas. Diperbaiki dengan dibongkar kembali dan diganti dengan lapis yang sesuai.

Gambar 2.6: Contoh penurunan pada bekas penanaman utilitas pada jalan [1]

2.3 Pengolahan Citra Digital

Citra dapat didefinisikan sebagai fungsi dua dimensi $f(x, y)$ dengan x dan y mempresentasikan titik koordinat, dan untuk setiap amplitudo f yang berpasangan dengan kordinat (x, y) disebut sebagai intensitas atau tingkat keabuan pada citra di titik tersebut. Jika x, y dan amplitudo f adalah terbatas dan bernilai diskrit maka dapat disebut sebagai citra digital. Citra digital terdiri dari sejumlah elemen yang disebut

pixel dimana setiap *pixel* memiliki lokasi dan nilai tertentu[2].

Pengolahan citra digital mengacu pada pengolahan/pemrosesan citra digital dengan input citra dan output dapat berupa citra atau atribut dari citra.

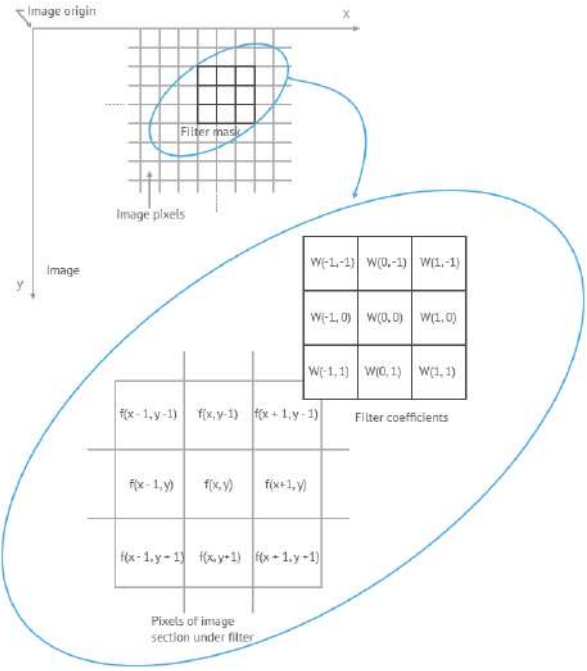
Pengolahan/pemrosesan citra memiliki dua tujuan. Yaitu melakukan peningkatan kualitas citra secara visual, dan mempersiapkan citra dalam pengambilan fitur pada citra tersebut.

Dalam mengolah suatu citra dapat melalui dua domain, yaitu domain spasial dan domain frekuensi. konsep pengolahan dengan domain spasial adalah pemrosesan citra secara langsung ke *pixel* sedangkan domain frekuensi berdasarkan perubahan intensitas *pixel* ke *pixel*. Pengolahan citra dengan domain spasial dapat dinotasikan dengan ekspresi sebagai berikut:

$$g(x, y) = T[f(x, y)] \tag{2.1}$$

dengan $f(x, y)$ merupakan suatu citra *input*, $g(x, y)$ merupakan *output*, dan T merupakan operator pada f yang didefinisikan pada ketetanggan dari titik (x, y). Umumnya ketetanggan berbentuk persegi, berada ditengah (x, y), dan ukurannya lebih kecil dari citra.

Spatial filtering mengacu kepada operasi ketetanggaan yang bekerja dengan nilai-nilai *pixel* pada ketetanggaan di citra tersebut dengan nilai yang sesuai pada *subimage* dengan memiliki dimensi yang sama dengan ketetanggaannya. *Subimage* dapat disebut sebagai filter, *mask, kernel, template, atau window* dimana nilai yang berada pada filter sebagai suatu koefisien. Terdapat dua jenis utama pada *spatial domain filtering*. Yaitu *linear spatial filtering* dan *nonlinear spatial filtering*. *linear spatial filtering* dapat diilustrasikan pada Gambar 2.1:



Gambar 2.7: Ilustrasi bagaimana proses pengolahan citra pada domain spasial [2].

Pada Gambar 2.7 diilustrasikan mekanisme *linear spatial filtering* dengan ukuran ketetanggan sebesar 3×3. Untuk setiap pixel (x, y) pada citra $f(x, y)$ didapatkan *output*, $g(x, y)$, dari filter adalah penjumlahan produk dari koefisien pada filter dan *pixel* citra yang dicakup oleh filter tersebut.

$$\begin{aligned} g(x,y) = \omega(-1,-1)f(x-1,y-1) + \omega(-1,0)f(x-1,y) \\ + \cdots + \omega(0,0)f(x,y) + \cdots + \omega(+1,1)f(x+1,y1) \\ + \omega(+1,+1)f(x+1,y+1) \end{aligned} \quad (2.2)$$

meninjau koefisien pusat filter, $\omega(0,0)$, sejajar dengan *pixel* pada (x,y). Secara umum, *linear spatial filtering* pada sebuah citra ukuran $M \times N$ dengan filter ukuran $m \times n$ diilustrasikan pada persamaan 2.3.

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t) \tag{2.3}$$

dengan $a = (m-1)/2, b = (n-1)/2$.

2.4 *Deep Learning*

Deep Learning merupakan salah satu bagian (*machine learning*). *Deep learning* akan melakukan pembelajaran yang jauh lebih dalam dengan terdapatnya *layer-layer* pada model. Penggunaan *deep learning* memiliki peningkatan kinerja yang baik dalam pengenalan suara, pengenalan objek visual, melakukan deteksi objek, dan masih banyak lainnya. Metode *machine learning* konvensional terbatas pada kemampuannya dalam memproses data dalam bentuk data mentah (*raw data*) dan direpresentasikan kedalam bentuk yang dapat dipelajari oleh model sehingga *input* perlu diidentifikasikan kedalam bentuk yang dapat dipahami agar mendapatkan sebuah kesimpulan dari model tersebut.

Metode *deep learning* akan melakukan pembelajaran dari satu *layer* ke *layer* lain. Hasil dari pembelajaran pada satu *layer* akan menghasilkan *fitur* yang lebih sederhana namun mempertahankan/memperkuat *input* penting. Sebagai contoh pada permasalahan pengenalan objek pada citra. Suatu *input* pada citra merupakan nilai-nilai yang terdapat pada setiap *pixel* di citra tersebut. Ketika *input* memasuki *layer* pertama, maka layer tersebut akan mempelajari dari seluruh nilai pada *input* apakah ada tepi dari objek tertentu. Kemudian pada *layer* ke-dua akan melakukan pengenalan terhadap motif dari *fitur* sebelumnya dan begitu seterusnya

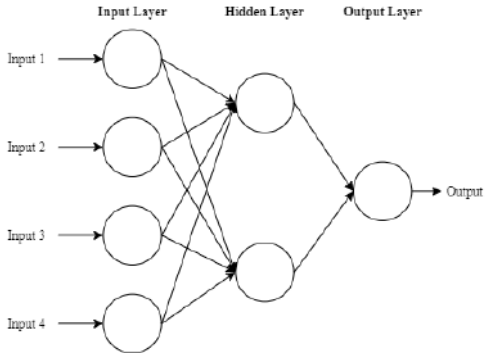
hingga model dapat melakukan pendeteksian dari *fitur* yang sudah lebih sederhana dan lebih mudah dikenali oleh model[16].

2.5 *Convolutional Neural Network*

Artificial Neural Network (ANN) atau jaringan syaraf tiruan adalah sistem pemrosesan komputasi yang terinspirasi dari cara sistem biologis berkerja. ANN terdiri dari sejumlah *node* komputasi yang saling terhubung (atau yang biasa dikenal dengan *neurons*) yang bekerja dengan cara saling mendistribusi antara satu sama lain untuk mengumpulkan suatu pembelajaran dari suatu yang bertujuan untuk meningkatkan hasil akhirnya.

Struktur dasar dari ANN dapat digambarkan pada Gambar 2.8, Dimulai dengan memberikan suatu inputan, biasanya dalam bentuk vektor multidimensi, kemudian akan didistribusikan ke *hidden layers*. *Hidden layer* akan membuat suatu keputusan dari layer sebelumnya dan menimbang bagaimana perubahan stokastik pada *node* tersebut. Hal tersebut dapat disebut sebagai proses pembelajaran. Model yang memiliki banyak *hidden layers* yang saling dihubungkan biasa disebut dengan *deep learning* [3].

Convolutional Neural Network berfokus kepada *input* berupa citra. Salah satu perbedaan utama adalah neuron yang merupakan layer pada CNN terdiri dari neuron yang sudah diatur kedalam tiga dimensi, yaitu dimensi spasial dari input (tinggi dan lebar) dan kedalaman (*depth*). Kedalaman disini tidak merujuk jumlah layer yang digunakan pada ANN, namun dimensi ketiga dari volume aktivasi. Tidak seperti ANN standar, neuron dalam *layer* tertentu hanya akan terhubung ke wilayah kecil dari *layer* sebelumnya. Dalam praktiknya, asumsi sebuah input akan memiliki dimensi $64 \times 64 \times 3$ (tinggi, lebar, dan kedalaman) yang mengarah kepada lapisan akhir yang terdiri dari dimesi $1 \times 1 \times n$ (dengan n



Gambar 2.8: Tiga layer dari *feedforward neural network* (FNN) sederhana, terdiri dari *input layer*, *hidden layer*, dan *output layer*[3].

mewakili kemungkinan jumlah kelas)

CNN pada dasarnya terdiri dari tiga tipe layer. Yaitu *convolutional layer, pooling layer*, dan *fully-connected layer*. Salah satu bentuk paling sederhana dari arsitektur CNN untuk klasifikasi MNIST diilustrasikan pada Gambar 2.9.

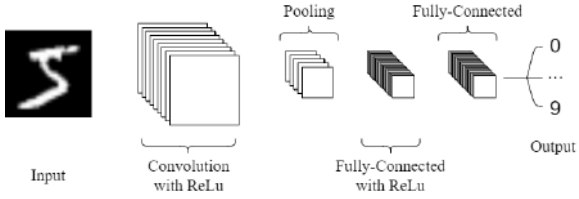
Fungsi inti dari contoh CNN tersebut dijabarkan kedalam empat bagian:

1. Karena CNN merupakan salah satu bentuk dari ANN, maka *input layer* akan menyimpan nilai *pixel* dari citra tersebut.

2. *Convolutional layer* akan menentukan output dari neuron yang terhubung ke region lokal dari input melalui perhitungan dari produk skalar antara *weight* dan region yang terhubung ke volume input. *Rectified linear unit* (atau yang biasa dikenal ReLu) bertujuan untuk menerapkan fungsi aktivasi dari *output* yang didapatkan pada layer sebelumnya.

3. *Pooling Layer* kemudian akan melakukan *downsampling* sepanjang dimensi spasial dari input yang diberikan, bertujuan untuk mengurangi jumlah parameter dalam aktivasi tersebut.

4. *Fully-Connected Layer* akan melakukan tugas yang sama seperti yang biasa ditemukan pada ANN dan berusaha untuk menghasilkan skor kelas dari aktivasi, yang nantinya akan digunakan untuk klasifikasi.



Gambar 2.9: Arsitektur CNN sederhana, terdiri dari lima layer[3].

2.6 *You Only Look Once* (YOLO)

You only look once (YOLO) merupakan salah satu sistem deteksi objek yang pertama kali diperkenalkan pada tahun 2015 dengan memiliki keunggulan kecepatan jika dibandingkan dengan sistem deteksi yang lainnya. Berbeda dengan deteksi objek yang lainnya, YOLO menyatukan komponen-kompnen deteksi objek kedalam suatu jaringan syaraf tunggal dimana YOLO langsung melakukan pemrosesan dari citra secara langsung. Pengembangan sistem deteksi YOLO masih terus dikembangkan dan hingga pada tahun 2018, YOLOv3 diperkenalkan dengan beberapa peningkatan khususnya pada tingkat akurasi yang jauh lebih tinggi namun memiliki sedikit penurunan kecepatan karena

kompleksitas pada arsitektur yang digunakan pada YOLOv3 jauh lebih besar.

2.6.1 Cara Kerja *You Only Look Once* (YOLO)

YOLO memanfaatkan semua fitur yang terdapat pada citra untuk memprediksi kotak pembatas (*bounding box*). Dimulai dengan membagi citra *input* ke dalam $S \times S$ petak kemudian pada setiap petak tersebut akan dilihat jika terdapat pusat objek yang berada pada petak, maka petak tersebut akan bertugas dalam mendeteksi objek tersebut. Setiap petak akan memprediksi B kotak pembatas (*bounding box*) dan *Confidence score* untuk setiap *bounding box* tersebut. Secara umum *confidence score* didefinisikan sebagai berikut:

$$ConfidenceScore = Pr(Object) * IoU_{pred}^{truth} \tag{2.4}$$

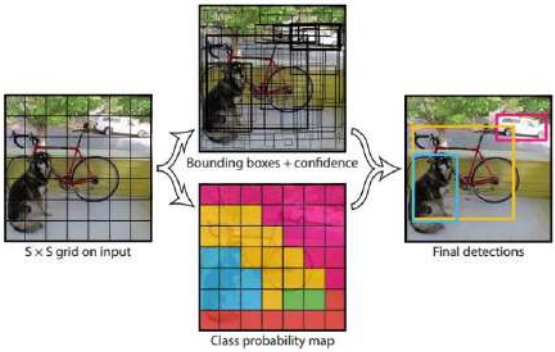
dengan $Pr(Object)$ merupakan probabilitas terdapatnya objek pada petak dan IoU_{pred}^{truth} merupakan nilai *intersect of union* antara *bounding box* ($pred$) dengan *ground truth* ($truth$). Jika tidak terdapat objek apapun pada *bounding box* tersebut, maka *confidence score* akan bernilai 0. Jika terdapat objek, maka *confidence score* dari *bounding box* tersebut sama dengan nilai dari *intersection of union* antara prediksi *bounding box* dengan *ground truth box*. Setiap *bounding box* terdiri dari lima prediksi: x, y, w, h, *confidence box* dan *conditional class probabilities*. Koordinat (x, y) mempresentasikan koordinat dari *bounding box* berdasarkan posisi pusat *bounding box* pada petak, Panjang w dan tinggi h merepresentasikan lebar dan tinggi dari *bounding box*, dan *Confidence score* merepresentasikan nilai IoU antara *predict box* dengan *ground truth box*.

Selanjutnya, setiap petak akan memprediksi C probabilitas kelas bersyarat (*conditional class probabilities*), $Pr(Class|Object)$. Probabilitas tersebut menggambarkan besar probabilitas kelas pada petak tersebut.

Saat melakukan test, probabilitas kelas bersyarat, *conditional class probabilities*, kemudian dikalikan dengan *confidence score*.

$$Pr(Class_i|Object) * Pr(Object) * IoU_{pred}^{truth} = Pr(Class_i) * IoU_{pred}^{truth} \quad (2.5)$$

sehingga menghasilkan *confidence score* untuk setiap kelas atau *class-specific confidence scores* pada setiap *bounding box*, nilai tersebut menggambarkan probabilitas *bounding box* dalam mengklasifikasikan objek tersebut [4].



Gambar 2.10: Ilustrasi Bagaimana YOLO bekerja [4].

Setelah *bounding box* didapatkan, ada kemungkinan dimana satu objek dapat dideteksi oleh beberapa *bounding box* dikarenakan beberapa kotak sel dapat menilai dirinya menjadi pusat dari objek tersebut, oleh karena itu seluruh *bounding box* yang sudah didapatkan sebelumnya akan melewati *Non-Max Suppression* untuk menghilangkan kotak pembatas yang tidak diperlukan. Illustrasi tugas Non-Max Suppression dapat dilihat pada Gambar 2.11

Misalkan terdapat *bounding box* pada kelas c adalah $B^c = b_1^c, b_2^c, \ldots, b_n^c$ dengan *confidence score* pada setiap *bounding box* adalah $S^c = S_1^c, S_2^c, \ldots, S_n^c$. Kemudian diberikan *NMS threshold* N_t maka algoritma dari *non-max suppression* dapat ditulis sebagai berikut [17]:

Algoritma 1: Non-max suppression

Input: B^c, S^c, N^t

$D^c \leftarrow \{\}$
while $B^c \neq empty$ **do**
  $m \leftarrow argmax S^c$
  $M \leftarrow b_m^c$
  $D^c \leftarrow D^c \cup M$
  $B^c \leftarrow B^c - M$
  for $b_i^c \in B^c$ **do**
    if $IoU(M, b_i^c) \geq N_t$ **then**
      $B^c \leftarrow B^c - b_i^c$
      $S^c \leftarrow S^c - s_i^c$
    end
  end
end
Output: D^c, S^c

2.6.2 Arsitektur YOLOv3

Dari desain jaringannya untuk mengekstrasi sebuah fitur pada citra, YOLOv3 menggunakan Darknet-53. Berbeda dengan YOLO generasi seperti YOLOv1 menggunakan *convolutional neural network* yang tediri dari 24 *convolutional layers*, 4 *max pool layers* dan 2 *fully connected layers* [4] ataupun YOLO v2 yang memanfaatkan *convolutional neural network* yang terdiri dari 19 *convolutional layers*, 5 *max pool layers* [18], jaringan YOLOv3 menggunakan *fully convolutional network* (FCN) untuk mengekstrasi fitur dari citra sehingga pada jaringan YOLOv3 tidak terdapat *max*

Gambar 2.11: Ilustrasi *Non-Max Suppression* dalam menghilangkan kotak pembatas *(bounding box)* yang tidak diperlukan.

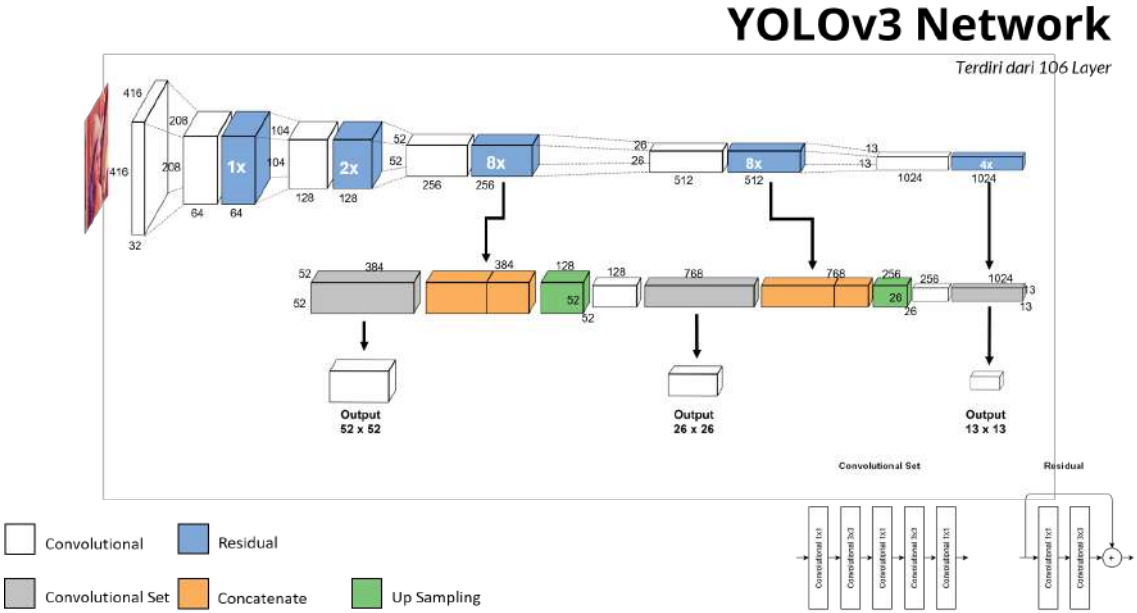
pool layer ataupun *fully connected layer*. Secara sederhana arsitektur jaringan Yolo v3 diilustrasikan pada Gambar 2.12

Arsitektur dari YOLOv3 terdiri dari beberapa komponen:

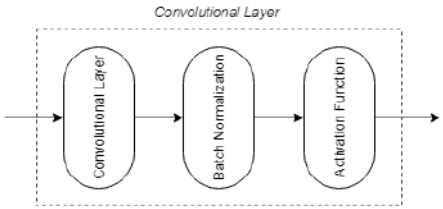
1. ***Convolutional Layer***
   Untuk setiap *Convolutional Layer* tersusun tiga layer. Yaitu *Convolutional Layer*, *Batch Normalization*, dan *Activation Function*.

2. ***Residual Unit***
   Residual Unit yang terdapat pada jaringan YOLOv3 tersusun atas *Convolutional Layer* dengan ukuran filter 3x3, *Convolutional Layer* dengan ukuran filter 1x1, dan sebuah *skip connection*.

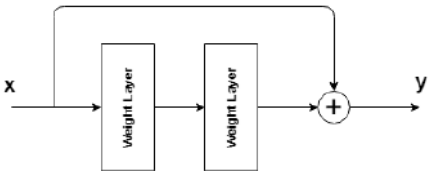
3. ***Up-Sampling***
   Up-Sampling yang terdapat pada jaringan YOLOv3 diperlukan untuk mengatur ukuran output.



Gambar 2.12: Arsitektur YOLO v3



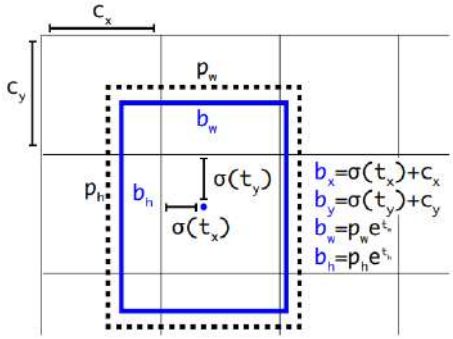
Gambar 2.13: Ilustrasi *Convolutional Layer* pada jaringan YOLOv3



Gambar 2.14: Ilustrasi dari *Residual Unit*

2.6.3 *Anchor Box*

Untuk menghasilkan suatu *bounding box*, YOLOv3 menggunakan *Anchor Box* sebagai inisialisai dari *Bounding Box*. Dengan begitu untuk menghasilkan sebuah *Bounding Box* detektor hanya perlu memilih *Anchor Box* yang memiliki ukuran paling mendekati dengan objek dan dilakukan penyesuaian. Saat melakukan pengujian pada *dataset* COCO, YOLOv3 memanfaatkan *K-Means Clustering* untuk menentukan *anchor box* yang digunakan. *Anchor box* tersebut berukuran $(10\times13),(16\times30),(33\times23),(30\times61),(62\times45),(59\times119),(116\times90),(156\times198),(373\times326)$ [5].



Gambar 2.15: Ilustrasi *Anchor Box.*[5]

Dikarenakan YOLOv3 memiliki *anchor box* sebagai inisiator dari *bounding box*, jaringan akan memprediksi nilai t_x, t_y, t_w, t_h sehingga posisi dan ukuran *bounding box* didapatkan dengan menggunakan persamaan berikut:

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_w &= P_w e^{t_w} \\ b_h &= P_h e^{t_h} \end{aligned} \tag{2.6}$$

Dengan c_x dan c_y merupakan posisi dari petak dan P_w, P_h merupakan ukuran dari *anchor box.*

2.6.4 *Lost Function*

Terdapat dua *loss function* yang digunakan pada YOLOv3, yaitu *Mean Squared Error* (MSE) dan *Binary Cross Entropy* (BCE). *Mean squared error* digunakan untuk mencari *loss* pada *bounding box* (x, y, w, h), *binary cross entropy* digunakan untuk mencari *objectness loss* dan *classification loss.*

Mean squared error didefinisikan sebagai berikut:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i) \tag{2.7}$$

dengan y_i sebagai vektor dari nilai sebenarnya dan $\hat{y}_i$ sebagai vektor prediksi yang didapatkan.

Binary cross entropy didefinisikan sebagai berikut:

$$BCE = -(ylog(\hat{y}) + (1-y)log(1-\hat{y})) \tag{2.8}$$

dengan y merupakan nilai sesungguhnya dalam bentuk biner (0 atau 1) dan $\hat{y}$ merupakan probabilitas dari hasil prediksi.

2.7 *Batch Normalization*

Batch Normalization diperkenalkan pada tahun 2015 yang bertujuan untuk meningkatkan kecepatan, kinerja, dan stabilitas dalam suatu jaringan syaraf tiruan. *Batch normalization* digunakan untuk mengatasi permasalahan distribusi aktivasi yang terus berubah selama proses *training* yang mengakibatkan penurunan kecepatan pada proses *training* karena setiap layer harus beradaptasi kembali dengan distribusi yang baru. Permasalahan tersebut dikenal sebagai *internal convariate shift*. Permasalahan tersebut dapat diatasi dengan menguragi *internal covariance shift* dengan melakukan normalisasi dengan memperbaiki *means* dan *variance* pada setiap *layer* [19]. *Batch normalization* dapat digunakan dengan mengikuti algoritma berikut [19]:

Algoritma 2: Batch Normalization

Input: Nilai x dari *mini-batch*:$\mathcal{B} = \{x_1 \dots m\}$
Parameters pembelajaran: γ, β

Output: $y_i = BN_{\gamma,\beta}(x_i)$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2)$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}} + \epsilon}}$$

$$y_i \leftarrow \gamma\hat{x} + \beta \equiv BN_{\gamma,\beta}(x_i)$$

2.8 *Confusion Matrix*

Confusion matrix adalah sebuah matrik yang mendeskripsikan bagaimana kinerja suatu model berdasarkan test data yang diberikan. *Confusion matrix* berisikan empat macam kondisi [20], yaitu:

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

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

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

4. False Negative (FN), kondisi dimana model mengklasifikasikan suatu data sebagai *false* dan kelas sebenarnya dari data tersebut adalah *true*

Untuk mengukur kinerja pada permasalahan pengenalan dan deteksi pada objek visual, kondisi-kondisi tersebut dapat diintepretasikan sebagai berikut [21].

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 sangat memungkinkan terdapat banyak deteksi tanpa objek dalam setiap data. Oleh karena itu kondisi TN dapat diabaikan, $TN = 0$.

Pengukuran nilai presisi (*precision*), *recall* dan *f1-score* dapat digunakan untuk mengetahui kinerja dari model dalam melakukan deteksi dan pengenalan suatu objek [20].

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 berikut:

$$Precision = \frac{TP}{TP + FP} \tag{2.9}$$

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

$$Recall = \frac{TP}{TP + FN} \tag{2.10}$$

F1-Score merupakan rangkuman dari nilai presisi dan nilai *recall*. *F1-Score* dapat dihitung melalui persamaan berikut

$$f1 - Score = \frac{(\beta^2 + 1) Precision \cdot Recall}{\beta^2 \cdot Precision + Recall} \tag{2.11}$$

dengan $\beta \in \mathbb{R}^+$

Selain nilai presisi, *recall*, dan *f1-score*, terdapat metode pengukuran kinerja lainnya seperti salah satunya *average precision*. *Average precision* akan merangkum bentuk dari kurva presisi dengan *recall* dan didefinisikan sebagai presisi rata-rata pada 11 level *recall* dengan jarak yang sama $r = [0, 0.1, 0.2, \ldots, 1.0]$ [21].

$$AP = \frac{1}{11} \sum_{r \in [0, 0.1, \ldots, 1.0]} P_{interp}(r) \tag{2.12}$$

Presisi pada setiap *recall* r kemudian akan di interpolasi dengan mengambil nilai presisi maksimal dengan menggunakan persamaan berikut.

$$P_{interp}(r) = \max_{\tilde{r}:\tilde{r} \geq r} p(\tilde{r}) \tag{2.13}$$

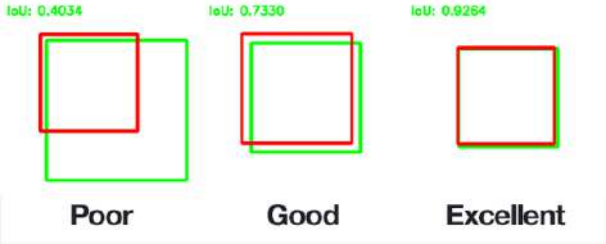
dengan $p(\tilde{r})$ merupakan presisi pada *recall* $\tilde{r}$.

2.8.1 *Intersection Over Union* (IoU)

Intersection Over Union adalah salah satu matrik evaluasi untuk berbagai macam tugas seperti segmentasi, *object detection*, dan *tracking*. Deteksi objek terdiri dari dua tugas, *localization*, dimana untuk mencari lokasi dari objek pada citra, dan klasifikasi untuk menentukan kelas pada objek tersebut. Tujuan dari *localization* pada deteksi objek adalah membuat sebuah kotak pembatas *(bounding box)* disekeliling objek. IoU akan menghitung seberapa baik prediksi kotak pembatas *(bounding box)* berdasarkan *ground truth* [22].

$$IoU_A^B = \frac{|A \cap B|}{|A \cup B|} \tag{2.14}$$

dengan A merupakan kotak prediksi dan B merupakan *ground truth box*.



Gambar 2.16: Ilustrasi dari *Intersection Over Union (IoU)*

Halaman ini sengaja dikosongkan

BAB III
METODE PENELITIAN

Pada bab ini berisi penjelasan tentang metode penelitian yang digunakan dalam tugas akhir. Metode penelitian penting tentunya karena akan dijadikan patokan utama dalam pengerjaan tugas akhir dan memudahkan alur pengerjaan menjadi lebih teratur.

3.1 Tahapan Penelitian

Terdapat langkah-langkah yang akan digunakan dalam menyelesaikan permasalahan pada Tugas Akhir ini. Secara sederhana dapat dilihat pada Gambar 3.1. Berikut merupakan penjelasan dari setiap langkahnya:

3.1.1 Studi literatur

Tahapan studi literatur diperlukan untuk mengumpulkan lebih banyak lagi refrensi-refrensi yang dapat menunjang pengerjaan tugas akhir sehingga dapat meningkatkan pemahaman dalam pengerjaan Tugas Akhir. Refrensi tersebut dapat berasal dari berbagai macam literatur seperti buku, tugas akhir, hingga jurnal ilmiah yang terkait dengan kerusakan jalan, dan pengenalan kerusakan jalan dengan metode YOLOv3.

3.1.2 Pengambilan Data dan Persiapan Data

Tahapan ini terdiri dari pengambilan data dan persiapan data. Pengambilan data diperlukan untuk mendapatkan data primer yang akan digunakan sebagai data latih dan data uji. Data yang akan digunakan terdiri dari kerusakan retak garis memanjang (*longitudinal cracking*), retak garis melintang

Gambar 3.1: Diagram Metode Penelitian

(*lateral cracking*), retak kulit buaya (*alligator cracking*), dan lubang (*potholes*). Pengambilan data dilakukan ketika kondisi pencahayaan baik dan data pada jalan dengan permukaan perkerasan lentur. Tahapan berikutnya adalah persiapan data yang meliputi *labelling data* dan *split data*. *Labelling Data* atau yang biasa disebut anotasi data merupakan tahapan yang diperuntukan untuk memberikan kotak pembatas *(bounding box)* disekitar objek yang akan dideteksi pada citra dan menghubungkannya dengan kelas. *Split Data* adalah tahapan yag diperuntukan untuk memisahkan antara *train data* dengan *test data*. *train data* difungsikan untuk *training* model

dan *test data* untuk menguji kinerja dari model.

3.1.3 *Training* Model Pengenalan Kerusakan Jalan

Setelah mendapatkan data yang diperlukan, Tahapan selanjutnya adalah *training* model dengan memanfaatkan data yang sudah diperoleh. Metode yang akan digunakan adalah metode YOLOv3. Dalam mempersiapkan model pengenalan akan digunakan *Darknet*, suatu *Framework Deep Learning*. Untuk memulai *training* pada model, akan digunakan *pre-trained weight* sebagai *starting weights* dan dari tahapan ini akan diambil *trained weight* dengan nilai *mean average precision* tertinggi.

3.1.4 Implementasi Program

Setelah mengetahui performansi dari model tersebut dan model tersebut sudah mendapatkan hasil yang baik, tahapan selanjutnya adalah mengimplementasikan model tersebut kedalam suatu program dengan *Input* sebuah video dan *output* dari program tersebut adalah suatu video yang sudah diberikan kotak pembatas *(bounding box)* pada bagian kerusakan jalan yang terdeteksi beserta jenis kerusakannya. Untuk mengimplementasikan program digunakan bahasa Python. Selain itu, library OpenCV digunakan dalam melakukan pengolahan citra dan PyQT5 sebagai *interface*.

3.1.5 Uji Coba

Untuk mengetahui kinerja dari YOLOv3 maka langkah selanjutnya adalah melakukan uji coba dengan mengukur nilai akurasi, presisi, *recall*, *average precision*, dan waktu yang diperlukan untuk melakukan pengenalan pada data video.

3.1.6 Penarikan Kesimpulan

Pada tahap ini dilakukan penarikan kesimpulan dari proses pada penulisan tugas akhir ini. Dari kesimpulan yang

didapatkan akan diberikan saran-saran sebagai acuan dalam melakukan pengembangan.

3.1.7 Penulisan Laporan Tugas Akhir

Pada tahap ini dilakukan penulisan laporan tugas akhir berdasarkan dari seluruh tahapan yang sudah dilakukan dalam pengerjaan tugas akhir.

3.2 Diagram Blok Pengenalan Kerusakan Jalan Dengan Metode YOLOv3

Diagram blok pengenalan kerusakan jalan dengan metode YOLOv3 ditunjukkan pada gambar 3.2. Berikut merupakan penjelasan dari diagram alir pengenalan kerusakan jalan dengan metode YOLOv3.

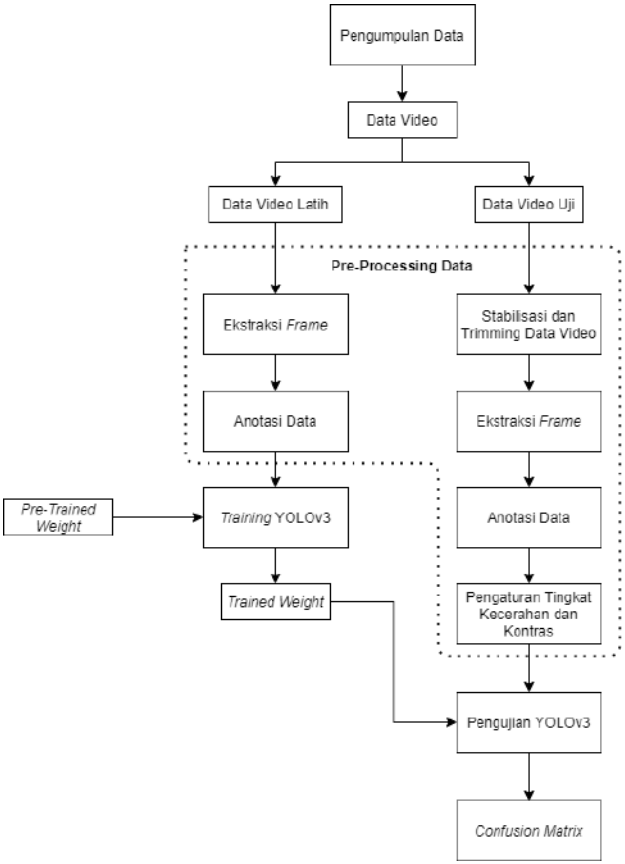
1. Pengumpulan Data:
   Tahap pengumpulan data bertujuan untuk mendapatkan data video yang akan digunakan dalam proses *training* dan pengujian YOLOv3. Adapun spesifikasi dari data video diantaranya memiliki format MPEG-4 (.mp4), resolusi video sebesar 1980 × 1080 (1080p) dan 1280 × 720 (720p), dan *framerate* video sebesar 30 FPS

2. *Pre-Processing Data*:
   Tahap *pre-processing data* bertujuan untuk mempersiapkan data sebelum proses *training* dan pengujian. Pada dasarnya, *pre-processing data* pada data video latih dan data video uji terdiri dari ekstraksi *frame* video dan melakukan anotasi data. Namun pada *pre-processing data* data video latih, terdapat tahapan penyortiran yang bertujuan untuk mendapatkan *frame* yang memiliki kerusakan jalan. Kemudian pada *pre-processing data* data video uji, terdapat

tahapan stabilisasi dan *trimming* pada data video dan pengaturan tingkat kecerahan dan kontras.

3. *training* YOLOv3:
   Tahap *training* YOLOv3 bertujuan untuk mendapatkan *trained weight*. Didalam tahapan *training*, data yang digunakan akan dibagi menjadi dua bagian, bagian pertama digunakan sebagai data latih dalam proses *training* dan bagian kedua akan digunakan sebagai data pengujian untuk menentukan *trained weight* terbaik. Kemudian digunakan *pre-trained weight* sebagai *weight* awal pada proses *training* model.

4. Pengujian YOLOv3:
   Tahap pengujian YOLOv3 bertujuan untuk mengetahui seberapa baik kinerja dari metode YOLOv3 dalam melakukan pengenalan kerusakan jalan. Dari tahap pengujian akan didapatkan *confusion matrix*



Gambar 3.2: Diagram Blok Pengenalan Kerusakan Jalan Dengan Metode YOLOv3

Halaman ini sengaja dikosongkan

BAB IV
PERANCANGAN DAN IMPLEMENTASI

Pada bab ini akan dijelaskan mengenai rancangan metode *You Only Look Once* agar dapat digunakan dalam untuk mengenali kerusakan jalan. Setelah itu akan dijelaskan pula bagaimana metode *You Only Look Once* diimplementasikan kedalam suatu program.

4.1 Pengambilan Data dan Persiapan Data

Untuk mempersiapkan metode *You Only Look Once* agar dapat digunakan dalam untuk mengenali kerusakan jalan. Diperlukan data yang akan digunakan sebagai data latih sehingga model dapat mengenali kerusakan jalan dan data uji agar dapat mengetahui bagaimana kinerja model dalam mengenali kerusakan jalan.

4.1.1 Pengambilan Data

Metode YOLOv3 memerlukan *dataset* citra agar metode tersebut dapat mengenali objek tertentu. Oleh karena itu tahapan pengambilan data diperlukan agar metode YOLOv3 dapat mengenali kerusakan jalan. Data yang akan digunakan adalah data primer yang terdiri dari beberapa jenis kerusakan jalan. Yaitu Retak Garis Memanjang *Longitudinal Crack*, Retak Garis Melintang (*Lateral Crack*), Retak Kulit Buaya (*Alligator Crack*), dan Lubang (*Pothole*). Lokasi pengambilan data berada di daerah Kabupaten Bogor, Kota Bogor, dan Kota Depok. Data diambil dengan melakukan perekaman di sepanjang jalan yang dilalui dengan menggunakan kamera *smartphone* yang diletakkan pada *dashboard* mobil yang berjalan pada kecepatan 20 km/h hingga 40 km/h. Data

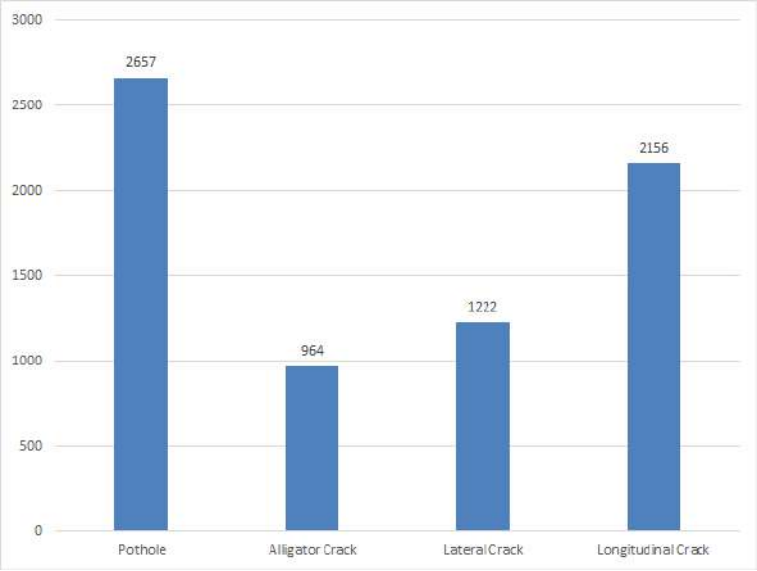
yang berhasil didapatkan sebanyak 56 video. Seluruh video berformat MPEG-4 (.mp4) dengan resolusi 1920 × 1080 (1080p) dan 1280 × 720 (720p) dan *framerate* 30 FPS.

Gambar 4.1: Posisi *smartphone* di mobil saat melakukan pengambilan data.

4.1.2 Persiapan Data Latih

Data video yang sudah didapatkan pada proses sebelumnya belum dapat digunakan dikarenakan *input* yang digunakan adalah citra. Oleh karena itu *frame* pada video akan diakuisi, dilanjutkan dengan penyortiran pada *frame* untuk mengambil *frame* yang memiliki kerusakan jalan pada jalan dengan perkerasan lunak, dan terakhir adalah diberikan anotasi/label terhadap kerusakan jalan yang terdapat pada *frame* tersebut. Hasil anotasi dari satu kerusakan jalan diatur dalam format `<id-kelas> <x-pusat>`

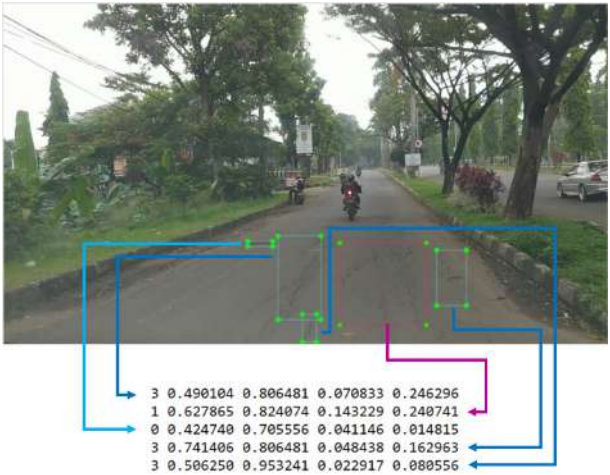
`<y-pusat> <panjang> <lebar>`, kemudian anotasi-anotasi yang terdapat pada citra tersebut akan disimpan kedalam satu file .txt dimana untuk setiap baris mewakili satu kerusakan. Satu file .txt tersebut disimpan dengan nama yang sama dengan pasangan citranya sehingga untuk satu file anotasi akan mewakili satu citra. Dalam penulisan Tugas Akhir, didapatkan 3321 citra jalan dengan 6999 kerusakan jalan. Berikut rincian total dari kerusakan jalan yang akan digunakan dalam penulisan Tugas Akhir.



Gambar 4.2: Total Kerusakan Jalan Berdasarkan Jenisnya

4.1.3 Pembagian Data Latih

Setelah didapatkan satu *dataset* yang akan digunakan dalam *training* model, tahap selanjutnya adalah membagi *dataset* tersebut menjadi dua *dataset* terpisah yang akan



Gambar 4.3: Ilustrasi citra yang sudah diberikan anotasi dengan format `<id-kelas> <x-pusat> <y-pusat> <panjang> <lebar>`

digunakan menjadi data latih dengan data uji. data latih akan digunakan dalam melakukan *training* model dan data uji akan digunakan untuk mengevaluasi *trained weight* terbaik dari seluruh *trained weight* yang didapatkan pada proses *training*. *Dataset* akan dibagi secara acak dengan perbandingan 80:20, 80% untuk data latih dan 20% untuk data uji. Proporsi data latih memiliki proporsi lebih banyak dibandingkan dengan data uji dikarenakan model harus mampu melakukan pengenalan dalam kondisi varians yang rendah, semakin banyak data latih maka semakin kecil pula varians pada data tersebut. Ketika proses latih mendapatkan eror yang minimum dengan kondisi varians data yang kecil, maka kinerja dari model tersebut dapat dikatakan baik. Diperlukannya data uji untuk melihat apakah model dalam melakukan

pengenalan pada data yang belum pernah diberikan. Jika model dapat mengenali sebagian objek secara benar pada data yang belum pernah diberikan maka model tersebut dapat dikatakan baik.

4.2 Penentuan *Anchor Box*

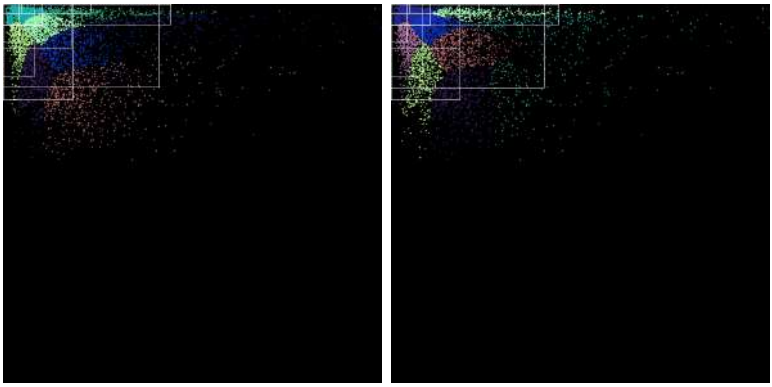
Sebelum melakukan *training*, diperlukan *anchor box* yang sesuai dengan objek yang akan dikenali. Untuk mendapatkan *anchor box* tersebut. Digunakan *K-Means clustering* untuk mendapatkan *anchor box*. *Inputan* yang akan diberikan adalah lebar dan tinggi dari setiap anotasi. Untuk mengetahui bagaimana *k-means clustering* bekerja, berikut merupakan algoritma dari *k-means clustering*.

Algoritma 3: K-Means Clustering Algorithm

Input: Each Annotation Box: $[w, h]$
Output: Centroid of the Cluster K

Choose the number of cluster k;
Select k random points from the data as centroids;
while *stopping criteria has not yet reached* **do**
    Assign all the points to the closest cluster centroid;
    Recompute the centroids
end

Dengan *k-means clustering* dengan 9 kluster, didapatkan centroid dari setiap kluster: $(21 \times 10), (17 \times 42), (96 \times 10), (43 \times 24), (34 \times 79), (75 \times 49), (184 \times 22), (76 \times 105), (171 \times 91)$ pada data dengan ukuran 416×416 dan $(26 \times 12), (21 \times 52), (118 \times 12), (52 \times 29), (42 \times 97), (92 \times 60), (227 \times 28), (92 \times 129), (206 \times 113)$ pada data dengan ukuran 512×512. Seluruh centroid yang didapatkan akan digunakan sebagai *anchor box* dalam mengenali kerusakan jalan.



Gambar 4.4: Ilustrasi kluster yang dihasilkan dengan *k-means clustering* pada data berukuran 416 × 416 (Kiri) dan data berukuran 512 × 512 (Kanan).

4.3 *Training* Model Pengenalan Kerusakan Jalan

Setelah memiliki data dengan anotasinya, tahapan selanjutnya adalah *training* model agar dapat mengenali kerusakan jalan. Proses *training* bertujuan untuk mendapatkan *trained weight* yang sesuai dengan *dataset*.

4.3.1 Arsitektur Jaringan

Dalam proses *training* ini digunakan jaringan YOLOv3 seperti pada Gambar 2.12. Berikut jaringan YOLOv3 secara detail dengan ukuran input (416 × 416) pada Tabel 4.1

Perlu diperhatikan bahwa untuk setiap *convolutional layer* pada jaringan diikuti dengan *batch normalization layer* dan *activation layer*. *Residual unit* tersusun atas *convolutional layer* dengan ukuran filter 1x1, *convolutional layer* dengan ukuran filter 3x3, dan diakhiri dengan operasi penjumlahan dengan layer input. Tidak ada pengurangan dimensi pada *residual unit* dikarenakan pergeseran/stride dari kedua filter sebanyak 1. Keberadaan *upsample layer* pada jaringan diperlukan karena adanya operasi *concatenate* sehingga perlu

penyeragaman lebar dan tinggi sebelum masuk kedalam operasi *concatenate*.

Tabel 4.1: Arsitektur YOLOv3 Secara Detail

No	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	27	1x1	1	linear	13x13x1024	13x13x27
82	YOLO						
83	Route - 79					13x13x512	13x13x512
84	Convolutional	256	1x1	1	leaky	13x13x512	13x13x256

Tabel 4.1: Lanjutan Arsitektur YOLOv3 Secara Detail

No	Layer	Total Filter	Ukuran Filter	Stride	Fungsi Aktivasi	Input	Output
85	Upsample					13x13x256	26x26x256
86	Route - 79 61						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	27	1x1	1	linear	26x26x256	26x26x27
94	YOLO						
95	Route - 91					26x26x256	26x26x256
96	Convolutional	128	1x1	1	leaky	26x26x256	26x26x128
97	Upsample					26x26x128	52x52x128
98	Route - 97 36						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	27	1x1	1	linear	52x52x256	52x52x27
106	YOLO						

Sebagai ilustrasi proses *training* berjalan, akan diilustrasikan mekanisme kerja dari beberapa layer penyusun pada jaringan.

a ***Convolutional Layer***

Sebagai ilustrasi, Diberikan sebuah *input* citra berukuran 416x416 dengan *channel red, green,* dan *blue.*

$$Red = \begin{bmatrix} 167 & 87 & \dots & 92 & 203 \\ 248 & 210 & \dots & 136 & 220 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 118 & 249 & \dots & 28 & 92 \\ 117 & 119 & \dots & 152 & 88 \end{bmatrix}$$

$$Green = \begin{bmatrix} 107 & 40 & \dots & 244 & 219 \\ 135 & 82 & \dots & 47 & 208 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 226 & 237 & \dots & 205 & 125 \\ 58 & 101 & \dots & 157 & 127 \end{bmatrix}$$

$$Blue = \begin{bmatrix} 231 & 234 & \dots & 168 & 195 \\ 134 & 239 & \dots & 239 & 27 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 15 & 63 & \dots & 135 & 77 \\ 169 & 168 & \dots & 61 & 88 \end{bmatrix}$$

Kemudian *input* tersebut akan diberikan operasi konvolusi dengan parameter sebagai berikut.

```
batch_normalize=1      stride=2
filters=64             pad=1
size=3                 activation=leaky
```

Seperti yang sudah disinggung sebelumnya, untuk setiap *convolutional layer* akan diikutin dengan *batch normalization* dan *activation function.* Dari parameter diatas, didapatkan bahwa layer tersebut memiliki ukuran filter 3x3 yang terdiri dari 64 filter, *stride* = 2, *pad* = 1, dan menggunakan fungsi aktivasi *leaky.* Parameter `pad = 1` merepresentasikan ukuran

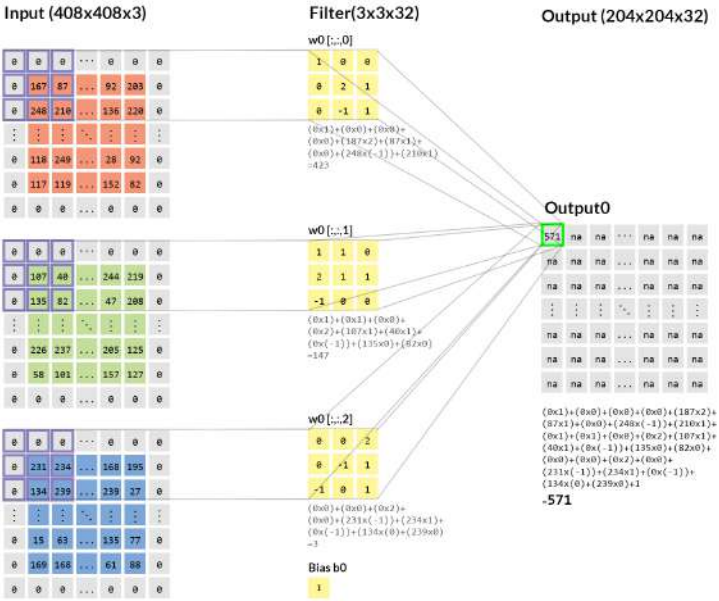
padding. Jika `pad = 1` maka akan dilakukan *padding* sebesar $(filter_size-1)/2$. Sebaliknya, jika `pad = 0`, maka tidak akan dilakukan *padding*. Sebagai ilustrasi berikut merupakan hasil *padding* sebesar 1.

$$Red = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 167 & 87 & \dots & 92 & 203 & 0 \\ 0 & 248 & 210 & \dots & 136 & 220 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 118 & 249 & \dots & 28 & 92 & 0 \\ 0 & 117 & 119 & \dots & 152 & 88 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 \end{bmatrix}$$

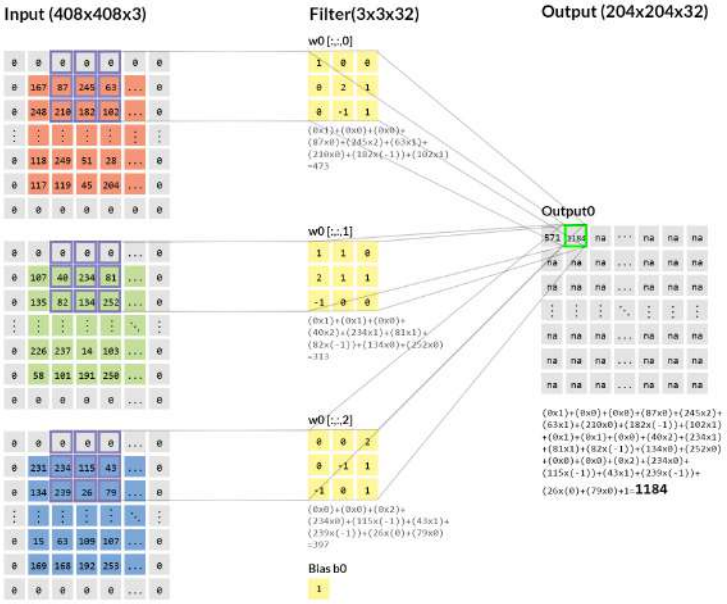
$$Green = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 107 & 40 & \dots & 244 & 219 & 0 \\ 0 & 135 & 82 & \dots & 47 & 208 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 226 & 237 & \dots & 205 & 125 & 0 \\ 0 & 58 & 101 & \dots & 157 & 127 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 \end{bmatrix}$$

$$Blue = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 231 & 234 & \dots & 168 & 195 & 0 \\ 0 & 134 & 239 & \dots & 239 & 27 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 15 & 63 & \dots & 135 & 77 & 0 \\ 0 & 169 & 168 & \dots & 61 & 88 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 \end{bmatrix}$$

Setelah dilakukan *padding*, maka tahap berikutnya adalah penerapan konvolusi. Akan diberikan ilustrasi bagaimana proses konvolusi berjalan dengan filter berukuran 3x3 dengan *stride* sebesar 2 pada gambar 4.4 dan gambar 4.5



Gambar 4.5: Ilustrasi proses *convolution* untuk mendapatkan nilai pada kolom 1 dan baris 1



Gambar 4.6: Ilustrasi proses *convolution* untuk mendapatkan nilai pada kolom 2 dan baris 1

Untuk mendapatkan hasil konvolusi di titik (0,0), Dengan matrix input X berukuran $n \times n$ dengan *channel* 3, filter W berukuran $n \times n \times 3$, bias b, dan hasil konvolusi Y, maka dapat melakukan perhitungan seperti pada persamaan 4.1.

$$\begin{aligned} Y(0,0) =& [(X(0,0,0) \times W(0,0,0)) + \cdots + (X(2,2,0) \\ & \times W(2,2,0)) + \cdots + (X(3,3,0) \times W(3,3,0))] \\ & + [(X(0,0,1) \times W(0,0,1)) + \cdots + (X(2,2,1) \\ & \times W(2,2,1)) + \cdots + (X(3,3,1) \times W(3,3,1))] \\ & + [(X(0,0,2) \times W(0,0,2)) + (X(1,1,2) \\ & \times W(1,1,2)) + \cdots + (X(2,2,2) \times W(2,2,2))] \\ & + b \\ =& [(0 \times 1) + \cdots + (167 \times 2) + \cdots + (210 \times 1)] + \\ & [(0 \times 1) + \cdots + (107 \times 1) + \cdots + (82 \times 0)] + \\ & [(0 \times 0) + \cdots + (231 \times (-1)) + \cdots + (239 \times 1)] \\ & + 1 \\ =& 423 + 147 + 3 + 1 = 574 \end{aligned} \tag{4.1}$$

Perlu diperhatikan penggunaan *stride* sebesar 2 mengakibatkan filter bergeser sejauh dua *pixel*. Maka untuk mendapatkan hasil konvolusi pada titik (1,0) dapat dilakukan seperti pada persamaan 4.2 .

Proses perhitungan tersebut dilakukan hingga mendapatkan seluruh nilai. Jika pada *convolutional layer* memiliki filter sebanyak n, maka perhitungan dilanjutkan dengan cara yang sama terhadap filter lainnya dan menghasilkan sebuah layer yang baru.

Dari *convolutional layer*, didapatkan sebuah *feature* berukuran $208 \times 208 \times n$ dimana 208×208 didapatkan dari proses konvolusi dengan *stride* sebesar 2 ($ukuran_input/stride$) dan n bergantung dari berapa

banyak filter yang digunakan pada layer tersebut.

$$\begin{aligned}
Y(1,0) =&[(X(2,0,0) \times W(0,0,0)) + \cdots + (X(3,2,0) \\
&\times W(2,2,0)) + \cdots + (X(4,3,0) \times W(3,3,0))] \\
&+ [(X(2,0,1) \times W(0,0,1)) + \cdots + (X(3,2,1) \\
&\times W(2,2,1)) + \cdots + (X(4,3,1) \times W(3,3,1))] \\
&+ [(X(2,0,2) \times W(0,0,2)) + (X(3,1,2) \\
&\times W(1,1,2)) + \cdots + (X(4,2,2) \times W(2,2,2))] \\
&+ b \\
=&[(0 \times 1) + \cdots + (245 \times 2) + \cdots + (102 \times 1)]+ \\
&[(0 \times 1) + \cdots + (234 \times 1) + \cdots + (252 \times 0)]+ \\
&[(0 \times 0) + \cdots + (115 \times (-1)) + \cdots + (79 \times 1)] \\
&+ 1 \\
=&473 + 313 + 397 + 1 = 1184
\end{aligned} \tag{4.2}$$

Setelah didapatkan *feature* dari layer sebelumnya, tahapan selanjutnya adalah melakukan *batch normalization*. Untuk setiap *feature* dari layer sebelumnya memiliki ukuran tensor (B, H, W, C). Dengan B merupakan banyaknya sampel citra yang digunakan dalam satu kali iterasi, H dan W merupakan panjang dan lebar dari *feature*, dan C merupakan *channel* / kedalaman dari *feature*.

Untuk mengetahui bagaimana *batch normalization* bekerja, akan diilustrasikan mekanisme dari layer tersebut. Diketahui terdapat sebuah input X berukuran $1 \times 208 \times 208 \times 1$. Maka akan dicari nilai *mean* dan *variace* untuk setiap

channel.

$$X(:,:,0) = \begin{bmatrix} 53 & 62 & 6 & \dots & 240 & 183 & 177 \\ 206 & 161 & 169 & \dots & 196 & 214 & 242 \\ 99 & 180 & 73 & \dots & 45 & 142 & 69 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 61 & 115 & 93 & \dots & 126 & 86 & 167 \\ 209 & 75 & 87 & \dots & 233 & 44 & 112 \\ 27 & 215 & 73 & \dots & 239 & 153 & 32 \end{bmatrix}$$

$$\mu_{\mathcal{B}} = \frac{1}{n}\sum_{i=1}^{n} x_i = 127.329 \tag{4.3}$$

$$\sigma^2_{\mathcal{B}} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 = 5421.075 \tag{4.4}$$

Setelah didapatkan nilai *mean* dan *variace*, maka akan dilakukan normalisasi terhadap seluruh nilai dari *input*.

$$\hat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}}}} \tag{4.5}$$

Kemudian didapatkan:

$$\begin{aligned} \hat{x}_i(0,0,0) &= \frac{53 - 127.329}{\sqrt{5421.075}} \\ \hat{x}_i(1,0,0) &= \frac{62 - 127.329}{\sqrt{5421.075}} \\ &\vdots \\ \hat{x}_i(2,2,0) &= \frac{73 - 127.329}{\sqrt{5421.075}} \\ &\vdots \\ \hat{x}_i(207,207,0) &= \frac{32 - 127.329}{\sqrt{5421.075}} \end{aligned} \tag{4.6}$$

Dari perhitungan diatas didapatkan $\hat{X}$ pada *channel* 0:

$$\hat{X} = \begin{bmatrix} -1.01 & -0.887 & -1.648 & \dots & 1.53 & 0.756 & 0.675 \\ 1.068 & 0.457 & 0.566 & \dots & 0.933 & 1.177 & 1.557 \\ -0.385 & 0.715 & -0.738 & \dots & -1.118 & 0.199 & -0.792 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ -0.901 & -0.167 & -0.466 & \dots & -0.018 & -0.561 & 0.539 \\ 1.109 & -0.711 & -0.548 & \dots & 1.435 & -1.132 & -0.208 \\ -1.363 & 1.191 & -0.738 & \dots & 1.517 & 0.349 & -1.295 \end{bmatrix}$$

Selanjutnya untuk mendapatkan $BN_{\gamma,\beta}(\hat{x}_i)$ dapat dihitung dengan melakukan perkalian antara $\hat{x}_i$ dengan γ dan dilanjutkan dengan penjumlahan oleh β

$$BN_{\gamma,\beta}(\hat{x}_i) = \gamma\hat{x}_i + \beta \tag{4.7}$$

akan didefinisikan γ dan β.

$$\beta = \begin{bmatrix} 0.311 & 0.073 & 0.558 & \dots & 0.782 & 0.429 & 0.935 \\ 0.466 & 0.387 & 0.833 & \dots & 0.266 & 0.296 & 0.11 \\ 0.242 & 0.224 & 0.666 & \dots & 0.45 & 0.474 & 0.905 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0.895 & 0.838 & 0.338 & \dots & 0.117 & 0.976 & 0.888 \\ 0.695 & 0.317 & 0.271 & \dots & 0.317 & 0.973 & 0.28 \\ 0.809 & 0.068 & 0.998 & \dots & 0.558 & 0.843 & 0.912 \end{bmatrix}$$

$$\gamma = \begin{bmatrix} 0.452 & 0.135 & 0.669 & \dots & 0.291 & 0.433 & 0.695 \\ 0.883 & 0.589 & 0.186 & \dots & 0.527 & 0.017 & 0.102 \\ 0.785 & 0.757 & 0.768 & \dots & 0.427 & 0.056 & 0.068 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0.478 & 0.376 & 0.229 & \dots & 0.008 & 0.874 & 0.687 \\ 0.104 & 0.312 & 0.024 & \dots & 0.074 & 0.135 & 0.767 \\ 0.726 & 0.749 & 0.366 & \dots & 0.154 & 0.251 & 0.025 \end{bmatrix}$$

Didapatkan *batch normalization* berdasarkan persamaan 4.7.

$$BN = \begin{bmatrix} 0.138 & 0.07 & -0.25 & \dots & 1.489 & 0.757 & 1.326 \\ 1.381 & 0.766 & 0.658 & \dots & 0.775 & 0.365 & 0.273 \\ 0.692 & 0.917 & 0.277 & \dots & -0.077 & 0.151 & -0.649 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ -0.328 & 0.236 & 0.071 & \dots & 0.005 & 0.326 & 1.166 \\ 0.875 & 0.087 & -0.125 & \dots & 0.529 & -0.966 & 0.708 \\ -0.377 & 0.83 & -0.371 & \dots & 1.001 & 0.545 & -1.156 \end{bmatrix}$$

Perlu diperhatikan bahwa perhitungan diatas merupakan ilustrasi untuk mendapatkan nilai *batch normalization* dari satu *channel* dan satu *batch*, oleh karena itu perhitungan diatas terus dilakukan untuk setiap *channel* dengan mengambil nilai dari seluruh data sampel.

Setelah *feature* dari *batch normalization* didapatkan, tahapan berikutnya adalah menerapkan fungsi aktivasi. Berikut adalah ilustrasi dari penerapan fungsi aktivasi *leaky ReLU* terhadap *feature* dari layer sebelumnya.

Diketahui bahwa fungsi aktivasi *Leaky ReLU* memiliki persamaan sebagai berikut:

$$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

Didefiniskan X sebagai matrix ukuran 208×208.

$$X = \begin{bmatrix} 0.138 & 0.07 & -0.25 & \dots & 1.489 & 0.757 & 1.326 \\ 1.381 & 0.766 & 0.658 & \dots & 0.775 & 0.365 & 0.273 \\ 0.692 & 0.917 & 0.277 & \dots & -0.077 & 0.151 & -0.649 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ -0.328 & 0.236 & 0.071 & \dots & 0.005 & 0.326 & 1.166 \\ 0.875 & 0.087 & -0.125 & \dots & 0.529 & -0.966 & 0.708 \\ -0.377 & 0.83 & -0.371 & \dots & 1.001 & 0.545 & -1.156 \end{bmatrix}$$

Maka penerapan fungsi aktivasi terhadap matrix X dijabarkan sebagai berikut:

$$\begin{aligned} X(0,0) &= f(0.138) = 0.138 \\ &\vdots \\ X(2,2) &= f(0.277) = 0.277 \\ &\vdots \\ X(206,206) &= f(-0.966) = -0.966 \times 0.01 = -0.0097 \\ &\vdots \\ X(207,207) &= f(-1.156) = -1.156 \times 0.01 = -0.01156 \end{aligned} \quad (4.8)$$

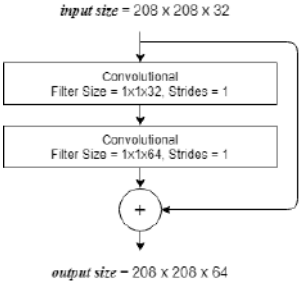
dari perhitungan tersebut didapatkan

$$X_{leaky} = \begin{bmatrix} 0.138 & 0.07 & -0.0025 & \dots & 1.489 & 0.757 & 1.326 \\ 1.381 & 0.766 & 0.658 & \dots & 0.775 & 0.365 & 0.273 \\ 0.692 & 0.917 & 0.277 & \dots & -0.0007 & 0.151 & -0.0065 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ -0.0034 & 0.236 & 0.071 & \dots & 0.005 & 0.326 & 1.166 \\ 0.875 & 0.087 & -0.0013 & \dots & 0.529 & -0.0097 & 0.708 \\ -0.0038 & 0.83 & -0.0037 & \dots & 1.001 & 0.545 & -0.0117 \end{bmatrix}$$

Perlu diperhatikan bahwa ilustrasi diatas merupakan penerapan fungsi aktivasi terhadap 1 layer. Fungsi aktivasi diterapkan di seluruh layer / *channel*.

b *Residual Unit*

Salah satu layer yang terdapat pada jaringan YOLOv3 adalah *residual unit*. Berikut merupakan ilustrasi dari *residual unit*.



Gambar 4.7: Ilustrasi *Residual Unit*

Diketahui terdapat *input* X berukuran $208 \times 208 \times 32$, dengan struktur *residual unit* seperti berikut. Proses yang terdapat pada layer konvolusi sama seperti pada proses layer konvolusi yang lain. Yang perlu diperhatikan adalah parameter dari layer tersebut. Dari dua layer tersebut, didapatkan *feature* berukuran $208 \times 208 \times 64$. Setelah didapatkan *feature* dari layer sebelumnya, tahapan selanjutnya adalah menerapkan operasi penjumlahan antara *feature* dengan *input*. Karena ukuran *feature* dan *input* sama, maka penjumlahan dilakukan sesuai dengan posisi baris, kolom, dan *channel*. Secara matematis *residual unit* dapat diilustrasikan dengan persamaan berikut.

$$y = f(x) + x \tag{4.9}$$

dengan $f(x)$ merupakan *feature* yang didapatkan dari dua layer konvolusi pada *residual unit*, dan x merupakan *input* awal. Dari *residual unit* didapatkan *feature* berukuran $208 \times 208 \times 64$.

c *Upsample*

Didalam jaringan YOLOv3 terdapat *upsample layer* yang berfungsi untuk memperbesar ukuran *input*. YOLOv3 menggunakan *bilinear* untuk melakukan *upsampling*. Sebagai

ilustrasi, diberikan sebuah *input* X berukuran 2×2 dan akan dilakukan *upsampling* dengan *factor* sebanyak 2.

$$X = \begin{bmatrix} 3.138 & 42.07 \\ 7.381 & 10.766 \end{bmatrix} \tag{4.10}$$

Ketika X akan diperbesar menjadi 2 kali ($factor = 2$), maka X akan berubah menjadi seperti berikut:

$$X = \begin{bmatrix} 3.138 & null & null & 42.07 \\ null & null & null & null \\ null & null & null & null \\ 7.381 & null & null & 10.766 \end{bmatrix} \tag{4.11}$$

Dari hasil perbesaran, terlihat bahwa terdapat kolom kosong / *null*. Tahap pertama adalah melakukan interpolasi linear searah sumbu x. Akan diilustrasikan mencari nilai $X(1, 0)$ dan $X(2, 0)$. Diketahui bahwa terdapat dua titik $(0, 3.138)$ dan $(3, 42.07)$ akan dicari kemiringan m dari dua titik tersebut.

$$\begin{aligned} m &= \frac{y_1 - y_0}{x_1 - x_0} \\ &= \frac{42.07 - 3.138}{3 - 0} = 12.977 \end{aligned} \tag{4.12}$$

Akan dicari nilai $X(1, 0)$

$$\begin{aligned} m &= \frac{y_1 - y_0}{x_1 - x_0} \\ 12.977 &= \frac{y_1 - 3.138}{1 - 0} \\ y_1 &= 12.977 + 3.138 = 16.115 \end{aligned} \tag{4.13}$$

didapatkan $X(1,0) = 16.115$ Akan dicari nilai $X(2,0)$

$$\begin{aligned} m &= \frac{y_1 - y_0}{x_1 - x_0} \\ 12.977 &= \frac{y_1 - 3.138}{2 - 0} \\ y_1 &= (12.977 \times 2) + 3.138 = 29.092 \end{aligned} \tag{4.14}$$

didapatkan $X(2,0) = 29.092$, sehingga

$$X = \begin{bmatrix} 3.138 & 16.115 & 29.092 & 42.07 \\ null & null & null & null \\ null & null & null & null \\ 7.381 & null & null & 10.766 \end{bmatrix} \tag{4.15}$$

Cara yang sama dilakukan untuk mendapatkan sisi horizontal lainnya. Sehingga didapatkan:

$$X = \begin{bmatrix} 3.138 & 16.115 & 29.092 & 42.07 \\ null & null & null & null \\ null & null & null & null \\ 7.381 & 8.509 & 9.637 & 10.766 \end{bmatrix} \tag{4.16}$$

Setelah seluruh nilai pada sisi horizontal didapatkan, tahap berikutnya adalah melakukan interpolasi linear searah sumbu y.

$$X = \begin{bmatrix} 3.138 & 16.115 & 29.092 & 42.07 \\ 4.552 & 13.58 & 22.607 & 31.606 \\ 5.966 & 11.045 & 16.132 & 21.272 \\ 7.381 & 8.509 & 9.637 & 10.766 \end{bmatrix} \tag{4.17}$$

d *Concatenate*

Terdapat operasi *Concatenate* didalam jaringan YOLOv3. *Concatenate* adalah operasi pengabungan dua *input*. Syarat dari opearsi *concatenate* adalah memiliki panjang dan lebar yang sama. Jika diketahui sebuah input A dengan ukuran

$(64 \times 64 \times 128)$ akan dilakukan operasi *concatenate* dengan B yang memiliki ukuran $(64 \times 64 \times 256)$, karena panjang dan lebar A dan B sama, maka operasi *concatenate* bisa dilakukan dan menghasilkan sebuah *feature* berukuran $(64 \times 64 \times 384)$

4.3.2 Parameter *Training*

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

Tabel 4.2: Parameter *Training* YOLOv3

Parameter	Nilai
batch	32
subdivisions	8
width	416
height	416
channels	3

Parameter	Nilai
momentum	0.9
decay	0.0001
learning_rate	0.001
burn_in	1000
policy	steps
steps	6500,8000,9000
scales	0.5,0.1,0.1

Parameter	Nilai
angle	0
saturation	1.5
exposure	1.5
hue	0.1

Proses training diatur dengan *batch* sebesar 32 dan *subdivisions* 8. 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 x 416 x 3.

Untuk mencegah lonjakan perubahan pada saat pembaharuan bobot, parameter *momentum* diatur sebesar 0.9 dan *decay* diatur sebesar 0.0001 untuk menghindari overfitting. Dilanjutkan dengan nilai *learning rate* awal sebesar 0.0001. 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 dibawah 1000 ($i < 1000$) sebesar

$learning_rate_{new} = learning_rate_{old} \times (\frac{i}{burn_in})^2$. 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 6500, 8000 dan 9000, *learning rate* akan diperbaharui menjadi $learning_rate_{new} = learning_rate_{old} \times scales$.

Untuk mendapatkan hasil pengenalan yang optimal, pada saat proses *training*, citra dari dataset akan dipilih secara acak untuk dirubah warnanya dengan mengatur *exposure*,*saturation*,*hue*.

4.4 Implementasi Program

Pada subbab ini akan dijelaskan bagaimana pengimplementasian program pada tugas akhir. Implementasi program tersebut diantaranya adalah implementasi program pengambilan data dan persiapan data, implementasi program *training* model pengenalan kerusakan jalan, implementasi program pengenalan objek, dan terakhir implementasi program *interface*.

4.4.1 Implementasi Program Pengambilan Data dan Persiapan Data

Untuk mendapatkan data yang siap digunakan dalam proses training atau uji coba, diperlukan mekanisme dan penyesuain dalam mendapatkan data tersebut. Beberapa implementasi program tersebut diantaranya adalah ekstraksi *frame* dari data video, anotasi data, dan pengaturan kecerahan dan kontras pada data. Berikut merupakan penjabarannya:

a Ekstraksi *Frame* dari Data Video

Seperti yang sudah dijabarkan sebelumnya, bahwa data yang diambil merupakan data video. Untuk melakukan *training* diperlukan sebuah citra dari objek yang ingin dideteksi. Oleh karena itu ektraksi *frame* dari data video

diperlukan untuk mendapatkan citra dari video tersebut. Proses ekstraksi *frame* dari data video diimplementasikan kedalam suatu program.

```
import cv2
from absl import app,flags,logging

FLAGS = flags.FLAGS
flags.DEFINE_string('video_path','./video/','path input video')
flags.DEFINE_string('name','null','Video Name')

def main(argv):
  if(FLAGS.video_path == './video/'):
    logging.info('Check your video path correctly!')
  else:
    if(FLAGS.name == 'null'):
      logging.info('Check your file name correctly!')
    else:
      cap = cv2.VideoCapture(FLAGS.video_path)
      logging.info('Video loaded '+FLAGS.video_path)
      i = 0
      n = 0
      while(cap.isOpened()):
        ret,frame = cap.read()
        if ret == False:
          break
        if(i%10 == 0):
          cv2.imwrite('./frame/'+FLAGS.name+'/'+FLAGS.name+'_'+str(i)+'.jpeg',frame)
          n += 1
        i += 1
      cap.release()
      cv2.destroyAllWindows()
      logging.info('Total frame        :'+str(i))
      logging.info('Total frame Saved :'+str(n))
if __name__ == '__main__':
  app.run(main)
```

Dari program diatas, terdapat dua *package* yang digunakan. `absl` digunakan sebagai basis aplikasi sederhana, `cv2` sebagai *package* pengolahan citra. Terdapat dua *input* pada program ini, diantaranya adalah `video_path` sebagai direktori lokasi video dan `name` sebagi nama citra pada *output*. Secara sederhana program ini akan membaca setiap *frame* pada video, kemudian setiap *frame* ke-10 akan disimpan menjadi sebuah citra pada lokasi yang sudah ditentukan.

b Anotasi Data

Pada tugas akhir ini, anotasi data dibantu dengan menggunakan aplikasi *LableImg*. Untuk mendapatkan file anotasi, tahap pertama adalah membuka direktori *folder* yang berisikan file citra yang ingin diberika anotasi, tahap selanjutnya adalah melakukan anotasi dari setiap gambar secara manual, setelah gambar sudah diberikan anotasi, tahap berikutnya adalah melakukan *save*. Dari proses anotasi tersebut didapatkan sebuah file berformat `.txt` yang memiliki nama *file* yang sama. Pada *file* tersebut setiap barisnya akan mewakili satu objek dan baris tersebut memiliki format penulisan `<id-kelas> <x-pusat><y-pusat> <panjang> <lebar>`.

c Pengaturan Kecerahan dan Kontras Pada Data

Implementasi program ini digunakan untuk mendapatkan sebuah citra baru memanipulasi pencahayaan dan kontras pada citra tersebut. Data tersebut digunakan untuk mengetahui kinerja model dalam mengenali objek pada berbagai kondisi pencahayaan dan kontras.

Pengaturan kecerahan atau kontras dapat dilakukan dengan menggunakan persamaan seperti berikut:

$$new\ image(x, y) = old\ image(x, y) * \alpha + \beta \tag{4.18}$$

dengan α dan β sebagai konstanta. Berikut merupakan *soruce code* dari program tersebut.

```
import cv2
import numpy as np
img = cv2.imread("D_11180.jpeg")

def brightness_contrast(img,brightness=0,contrast =
    0):
    if brightness!=0:
        buf = cv2.addWeighted(img,1,img,0,
    brightness)
    else:
        buf = img.copy()

    if contrast!=0:
        f = 259*(contrast + 255)/(255*(259-contrast
    ))
        alpha_c = f
        gamma_c = 127*(1-f)

        buf = cv2.addWeighted(buf, alpha_c, buf, 0,
     gamma_c)
        print(alpha_c,gamma_c)
    return buf

img_filter1 = brightness_contrast(img,-50,0)
cv2.imwrite("D_11180_0.jpeg",img_filter1)
```

Secara sederhana, *source code* akan membaca sebuah citra, kemudian nilai pada citra tersebut akan diatur kecerahan dan kontrasnya dengan melakukan operasi titik pada setiap *pixel* di citra tersebut. Setelah diatur kecerahan dan kontrasnya, tahap selanjutnya adalah menyimpan citra tersebut pada lokasi yang sudah didtentukan.

4.4.2 Implementasi Program *Training* Model Pengenalan Kerusakan Jalan

Setelah seluruh data sudah diberikan anotasi, tahap selanjutnya adalah melakukan *training* untuk mendapatkan

trained weight. Pada tahap ini digunakan *darknet* sebagai *framework* jaringan syaraf tiruan. Proses *training* dilakukan pada *Virtual Machine Google Colaboratory*. Berikut merupakan *source code* pada tahap *training* model pengenalan kerusakan jalan.

a Cloning dan melakukan konfigurasi pada *Darknet*

Tahap pertama yang perlu dilakukan adalah melakukan *import / clone darknet* dari *repository*. Setelah *darknet* berhasil tersimpan pada *virtual machine*, tahap berikutnya adalah melakukan konfigurasi agar *darknet* berjalan dengan menggunakan GPU dan *OpenCV*.

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

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

# make darknet (build)
!make
```

command `!git clone` digunakan untuk melakukan *clone* dan disimpan pada *virtual machine*. kemudian terdapat *command* `!sed` yang digunakan untuk mengganti kalimat dengan kalimat lain pada *file* tertentu. *command* terakhir adalah `!make` yang digunakan untuk melakuka eksekusi terhadap *file* `Makefile`

b Sinkronisasi *Virtual Machine* dengan *Google Drive*

Setelah *darknet* sudah tersedia di *virtual machine*, tahap selanjutnya adalah melakukan sinkronisasi *virtual machine* dengan *google drive*. Hal ini diperlukan agar *virtual machine* dapat mengakses *Google Drive*. Proses tersebut dapat dilakukan dengan menggunakan *source code* berikut.

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

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

Command `%cd` berfungsi untuk berpindah ke direktori lain. Selanjutnya adalah `drive.mount('/content/gdrive')` yang berfungsi untuk melakukan sinkronisasi antara *virtual machine* dengan *Google Drive*. Tahap terakhir adalah *command* `!ln -s /content/gdrive/My Drive/ /mydrive` yang digunakan untuk memperingkas direktori `/content/gdrive/My Drive/` menjadi `/mydrive`

Setelah *Google Drive* berhasil disinkronisasi, direktori file yang terdapat di *Google Drive* dapat diakses melalui direktori `/mydrive`

c *Import* Data Latih, Data Uji, dan File-File Konfigurasi Lainnya Kedalam *Virtual Machine*

Tahap selanjutnya adalah melakukan *import*. Hal tersebut dapat dilakukan melalui *source code* berikut.

```
%cd /content

!cp /mydrive/yolov3-608x608/obj.zip ../
!unzip ../obj.zip -d darknet/data/obj

%cd /content/darknet
!cp /mydrive/yolov3-608x608/yolov3_custom_608.cfg
    ./cfg
!cp /mydrive/yolov3-608x608/obj.names ./data
!cp /mydrive/yolov3-608x608/obj.data ./data
!cp /mydrive/yolov3-608x608/train.txt ./data
!cp /mydrive/yolov3-608x608/valid.txt ./data
```

command `%cp` digunakan untuk melakukan duplikasi dan `!unzip` digunakan untuk mengekstraksi *file* `.zip`.

d *Import Pretrained Weight*

Untuk memulai *training*, akan digunakan *preretrained weight* yang berfungsi sebagai bobot awal dalam melakukan *training*. Untuk melakukan *import* dapat digunakan *source code* berikut.

```
%cd /content/darknet/
# upload pretrained convolutional layer weights
!wget http://pjreddie.com/media/files/darknet53.
    conv.74
```

Command `!wget` digunakan untuk melakukan *download file*.

e *Training* Model

Setelah semua persiapan sudah dilakukan sebelumnya, langkah terakhir adalah memulai *training*. Untuk memulia *training* dapat dilakukan dengan menggunakan *source code* berikut.

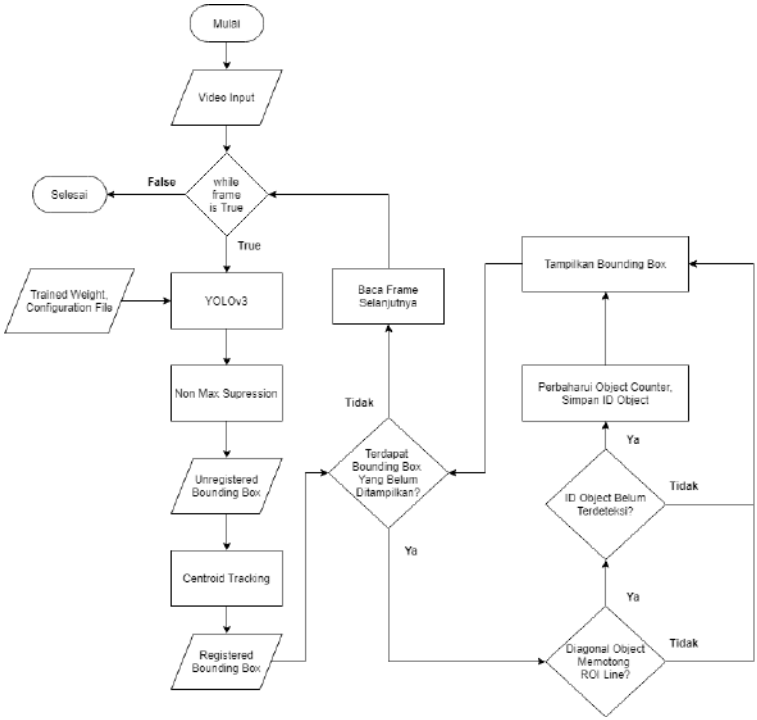
```
%cd /content/darknet/
# train your custom detector
!./darknet detector train data/obj.data cfg/
    yolov3_custom_608.cfg darknet53.conv.74 -
    dont_show -map
```

Command tersebut memiliki beberapa parameter diantaranya direktori *file* `.data`, direktori *file* `.cfg`, direktori *pretrained weight*, `-dont_show` berfungsi untuk tidak menampilkan *loss-windows*, dan `-map` digunakan untuk menyimpan grafik *loss*.

4.4.3 Implementasi Program Pengenalan Kerusakan Jalan

Implementasi program pengenalan kerusakan jalan bertujuan untuk membuat suatu program yang dapat melakukan pengenalan kerusakan jalan dan menghitung kerusakan jalan yang dikenali dari data video. Secara

sederhana implementasi program pengenalan kerusakan jalan memiliki rancangan pada gambar 4.7.



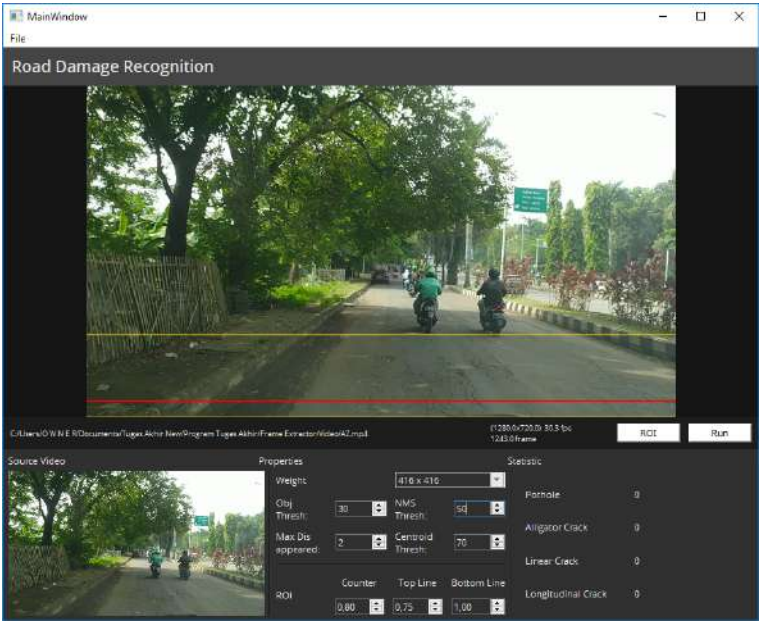
Gambar 4.8: Diagram Alir Dari Program Pengenalan Kerusakan Jalan

Video *input* yang masuk akan dibaca setiap *frame* dam untuk setiap *frame* akan dilakukan pengenalan kerusakan jalan dengan menggunakan YOLOv3, dari proses tersebut didapatkan *bounding box*. Tahapan berikutnya adalah membersihkan *bounding box* yang sebelumnya sudah didapatkan dengan menggunakan *Non-Max Supression* untuk menghindari duplikasi *bounding box* pada satu objek. Setelah

didapatkan *bounding box* yang sudah dibersihkan sebelumnya, tahapan berikutnya adalah melakukan *tracking* dengan menggunakan *centroid tracking* untuk menentukan nomor id dari *bounding box*. Penomoran ini diperlukan untuk membedakan antara satu objek dengan objek yang lain. Setelah tahapan *tracking*, tahap berikutnya adalah melakukan perhitungan objek. Suatu objek dapat dihitung ketika garis diagonal dari objek tersebut memotong *Region of Interest Line* dan objek tersebut belum dihitung sebelumnya. Tahap terakir pada program pengenalan ini adalah menampilkan *bounding box*. Proses penampilan *bounding box* dilakukan untuk setiap objek yang berhasil dikenali dan seluruh proses pengenalan kerusakan jalan terus dilakukan hingga seluruh *frame* selesai dikenali.

4.4.4 Implementasi Program *Interface*

Program *interface* dibangun untuk mempermudah penggunaan metode YOLOv3 dalam melakukan pengenalan kerusakan jalan. Berikut merupakan *interface* dari program pengenalan kerusakan jalan. Pembuatan program *interface* menggunakan *python* sebagai bahasa pemrograman dan dibantu oleh beberapa *package* untuk mempermudah pembuatan program. *package* tersebut diantaranya adalah pyQt5 sebagai modul utama dalam membangun *interface* dan OpenCv sebagai modul pengolahan citra pada program tersebut.



Gambar 4.9: *Interface* dari program pengenalan kerusakan jalan

Halaman ini sengaja dikosongkan

BAB V
UJI COBA DAN PEMBAHASAN

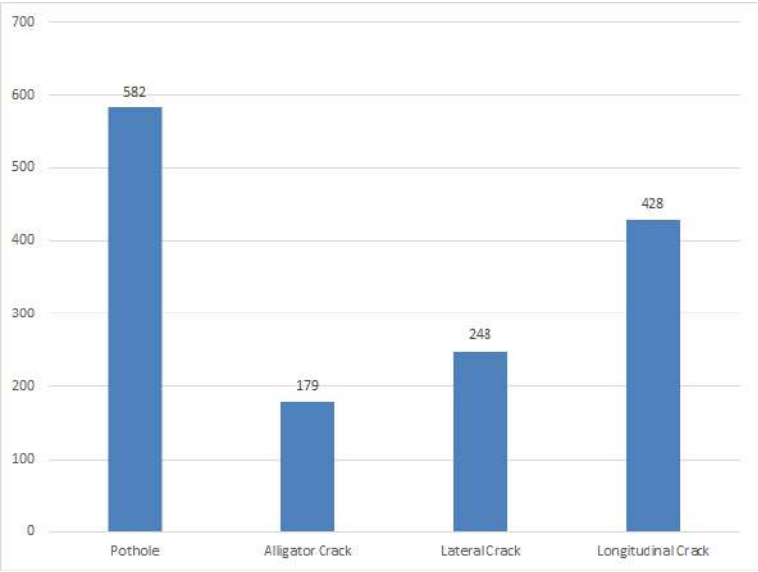
Pada bab ini akan dijelaskan mengenai uji coba dalam melakukan pengenalan kerusakan jalan. Terdapat dua pengujian yang akan dilakukan, yang pertama adalah melakukan pengujian terhadap *trained weight* sebagai acuan dalam mengambil *trained weight* terbaik, dan yang kedua adalah melakukan pengujian kinerja YOLOv3 dalam melakukan pengenalan kerusakan jalan pada data video.

5.1 Pengujian *Trained Weight*

Pada subbab ini akan dilakukan pengujian terhadap *trained weight* yang didapatkan dari proses *training*. Tujuan dari pengujian ini untuk menentukan *trained weight* terbaik dari proses *training*.

5.1.1 Data Pengujian *Trained Weight*

Data pengujian diambil dari keseluruhan data citra dengan perbandingan data *training* dan data uji sebesar 80:20 yang dipilih secara *random*. Berdasarkan perbandingan tersebut didapatkan data uji sebanyak 665 data citra dengan 1437 kerusakan jalan. Berikut merupakan jumlah data berdasarkan jenis kerusakan jalan.



Gambar 5.1: Jumlah data uji berdasarkan jenis kerusakan

5.1.2 Skenario Tahap Pelatihan Model Pengenalan Jalan

Terdapat dua skenario *training* yang akan digunakan untuk mendapatkan *trained weight*. Dua skenario tersebut memiliki perbedaan pada parameter ukuran *input* yang akan digunakan pada saat *training*. Berikut merupakan dua skenario yang akan digunakan:

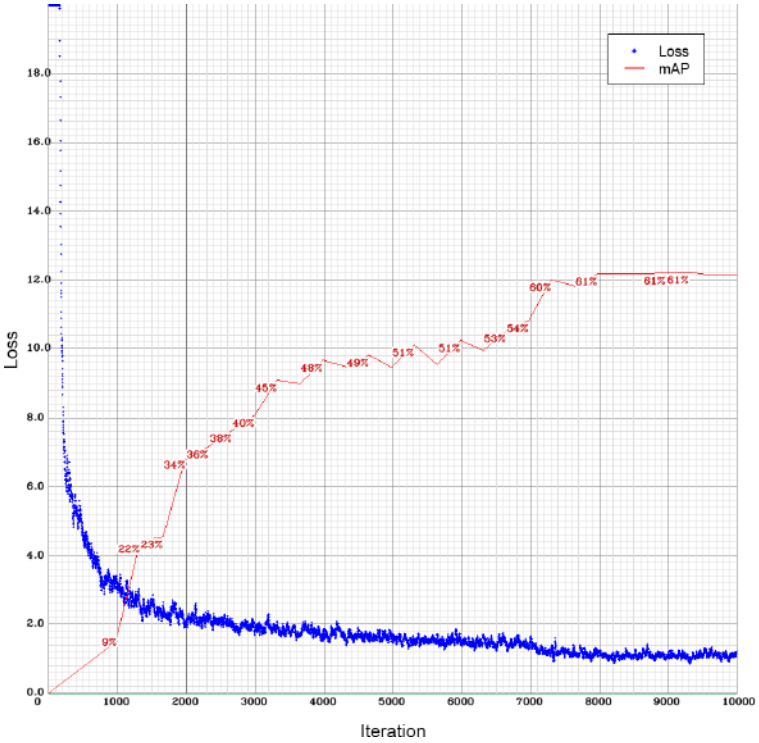
1. Skenario 1: Parameter ukuran *input* berukuran 416×416
2. Skenario 2: Parameter ukuran *input* berukuran 512×512

Untuk mengetahui kinerja dari setiap skenario, maka *trained weight* yang didapatkan dari dua skenario tersebut akan diuji coba dengan *threshold* 0.5 dan jumlah iterasi maksimal sebesar 10000 (240 *epoch*). Dari dua skenario

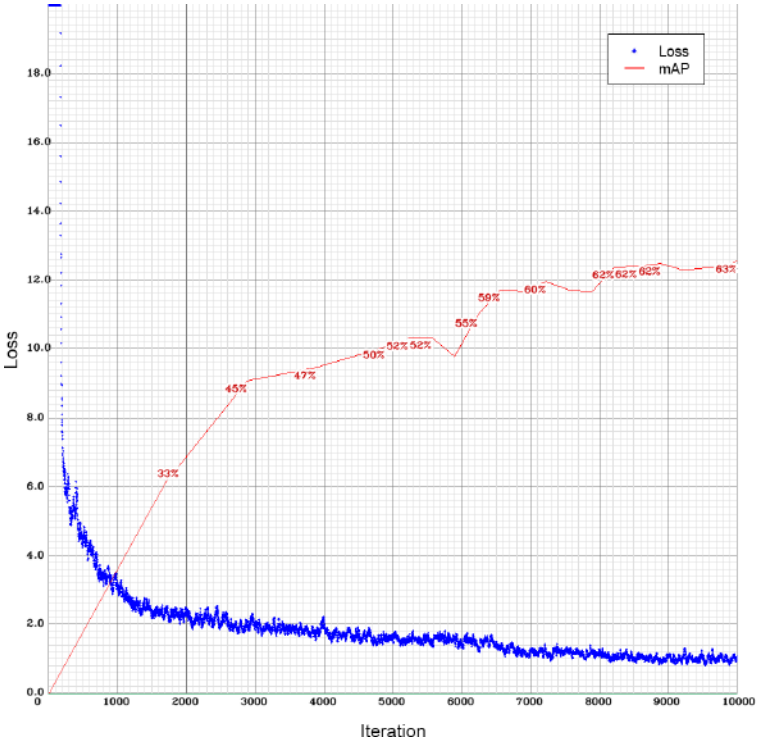
tersebut akan diukur kinerjanya berdasarkan nilai *loss* dan *mean average precision* dan kemudian *trained weight* yang memiliki nilai mAP tertinggi dari setiap skenario akan disimpan untuk digunakan sebagai *trained weight* pengenalan kerusakan jalan.

5.1.3 Hasil *Training* Beserta Pengujiannya

Setelah dilakukan *training* kemudian akan dilakukan evaluasi dengan meninjau grafik nilai *loss* dan nilai *mAP* selama proses *training* pada setiap skenario.



Gambar 5.2: Grafik nilai *loss* dan *mAP* pada skenario pertama



Gambar 5.3: Grafik nilai *loss* dan *mAP* pada skenario kedua

Berdasarkan dua grafik tersebut, pada skenario pertama didapatkan nilai *loss* pada iterasi terakhir sebesar 0.861752 dengan *mAP* tertinggi didapatkan pada iterasi ke 9302 dengan *mAP* sebesar 61.14% dan *loss* sebesar 0.8319. Kemudian pada skenario kedua didapatkan *mAP* tertinggi pada iterasi ke 10000 dengan *mAP* 62.86% dengan *loss* sebesar 0.9868. Untuk rincian dari hasil *training* dapat dilihat pada Tabel 5.1 dan Tabel 5.2

Tabel 5.1: Hasil Pengujian *Trained Weight* Dengan Parameter $Input = (416 \times 416)$ dan $Threshold = 0.5$

Iteration	Loss	Class	Confusion Matrix			IoU Threshold = 0.5			
			TP	FP	FN	Precision	Recall	F1 Score	$AP_{0.5}$
		Pothole	68	95	514	41.72%	11.68%	18.26%	16.73%
		Alligator Crack	16	59	163	21.33%	8.94%	12.60%	8.86%
1000	2.984719	Lateral Crack	1	1	247	50.00%	0.40%	0.80%	1.19%
		Longitudinal Crack	23	64	405	26.44%	5.37%	8.93%	7.51%
			108	**219**	**1329**	**33.03%**	**7.52%**	**12.24%**	**8.57%**
		Pothole	200	165	382	54.79%	34.36%	42.24%	37.60%
		Alligator Crack	85	64	94	57.05%	47.49%	51.83%	48.42%
2000	2.066975	Lateral Crack	76	194	172	28.15%	30.65%	29.34%	19.87%
		Longitudinal Crack	155	135	274	53.45%	36.13%	43.12%	38.27%
			516	**558**	**922**	**48.04%**	**35.88%**	**41.08%**	**36.04%**
		Pothole	235	235	347	50.00%	40.38%	44.68%	39.96%
		Alligator Crack	114	161	65	41.45%	63.69%	50.22%	55.62%
3000	2.14019	Lateral Crack	64	180	184	26.23%	25.81%	26.02%	12.81%
		Longitudinal Crack	153	116	275	56.88%	35.75%	43.90%	40.14%
			566	**692**	**871**	**44.99%**	**39.39%**	**42.00%**	**37.13%**
		Pothole	294	502	288	36.93%	50.52%	42.67%	34.53%
		Alligator Crack	100	117	79	46.08%	55.87%	50.51%	48.32%
4000	1.733564	Lateral Crack	110	191	138	36.54%	44.35%	40.07%	29.73%
		Longitudinal Crack	195	154	233	55.87%	45.56%	50.19%	46.21%
			699	**964**	**738**	**42.03%**	**48.64%**	**45.10%**	**39.70%**
		Pothole	332	211	250	61.14%	57.04%	59.02%	56.91%
		Alligator Crack	117	86	62	57.64%	65.36%	61.26%	61.08%
5000	1.862938	Lateral Crack	106	118	142	47.32%	42.74%	44.92%	34.96%
		Longitudinal Crack	220	165	208	57.14%	51.40%	54.12%	53.40%
			775	**580**	**662**	**57.20%**	**53.93%**	**55.52%**	**51.59%**
		Pothole	288	106	294	73.10%	49.48%	59.02%	58.40%
		Alligator Crack	117	74	62	61.26%	65.36%	63.24%	61.18%
6000	1.250441	Lateral Crack	111	129	137	46.25%	44.76%	45.49%	38.63%
		Longitudinal Crack	206	122	222	62.80%	48.13%	54.50%	52.08%
			722	**431**	**715**	**62.62%**	**50.24%**	**55.75%**	**52.57%**
		Pothole	334	126	248	72.61%	57.39%	64.11%	63.98%
		Alligator Crack	109	39	70	73.65%	60.89%	66.67%	64.01%
7000	0.954088	Lateral Crack	116	92	132	55.77%	46.77%	50.88%	44.77%
		Longitudinal Crack	200	97	228	67.34%	46.73%	55.17%	50.30%
			759	**354**	**678**	**68.19%**	**52.82%**	**59.53%**	**55.77%**
		Pothole	401	157	181	71.86%	68.90%	70.35%	68.08%
		Alligator Crack	124	78	55	61.39%	69.27%	65.09%	68.62%
8000	1.195701	Lateral Crack	141	107	107	56.85%	56.85%	56.85%	52.36%
		Longitudinal Crack	235	139	193	62.83%	54.91%	58.60%	54.91%
			901	**481**	**536**	**65.20%**	**62.70%**	**63.92%**	**60.99%**
		Pothole	402	140	180	74.17%	69.07%	71.53%	69.74%
		Alligator Crack	127	60	52	67.91%	70.95%	69.40%	67.72%
9000	1.226271	Lateral Crack	137	101	111	57.56%	55.24%	56.38%	51.40%
		Longitudinal Crack	240	147	188	62.02%	56.07%	58.90%	55.15%
			906	**448**	**531**	**66.91%**	**63.05%**	**64.92%**	**61.00%**
		Pothole	406	140	176	74.36%	69.76%	71.99%	70.16%
		Alligator Crack	127	60	52	67.91%	70.95%	69.40%	67.71%
9302	0.8319	Lateral Crack	137	102	111	57.32%	55.24%	56.26%	51.72%
		Longitudinal Crack	239	148	189	61.76%	55.84%	58.65%	54.98%
			909	**450**	**528**	**66.89%**	**63.26%**	**65.02%**	**61.14%**
		Pothole	406	140	176	74.36%	88.65%	80.88%	69.44%
		Alligator Crack	127	66	52	65.80%	53.81%	59.21%	66.24%
10000	0.861752	Lateral Crack	139	107	109	56.50%	42.77%	48.69%	52.27%
		Longitudinal Crack	242	152	186	61.42%	41.09%	49.24%	55.23%
			914	**465**	**347**	**66.28%**	**72.48%**	**69.24%**	**60.80%**

Tabel 5.2: Hasil Pengujian *Trained Weight* Dengan Parameter $Input = (512 \times 512)$ dan $Threshold = 0.5$

Iteration	Loss	Class	Confusion Matrix			IoU Threshold = 0.5			
			TP	FP	FN	Precision	Recall	F1 Score	F1 Score
		Pothole	147	319	435	31.55%	25.26%	28.05%	19.94%
		Alligator Crack	38	87	141	30.40%	21.23%	25.00%	18.53%
1000	2.630967	Lateral Crack	24	116	224	17.14%	9.68%	12.37%	6.19%
		Longitudinal Crack	41	74	387	35.65%	9.58%	15.10%	13.22%
			250	**596**	**1187**	**29.55%**	**17.40%**	**21.90%**	**14.47%**
		Pothole	202	211	380	48.91%	34.71%	40.60%	34.71%
		Alligator Crack	93	139	86	40.09%	51.96%	45.26%	43.80%
2000	3.486763	Lateral Crack	36	139	212	20.57%	14.52%	17.02%	9.52%
		Longitudinal Crack	126	94	302	57.27%	29.44%	38.89%	36.74%
			457	**583**	**980**	**43.94%**	**31.80%**	**36.90%**	**31.19%**
		Pothole	316	174	266	64.49%	54.30%	58.96%	56.42%
		Alligator Crack	89	54	90	62.24%	49.72%	55.28%	55.76%
3000	1.959354	Lateral Crack	77	65	171	54.23%	31.05%	39.49%	32.77%
		Longitudinal Crack	118	77	310	60.51%	27.57%	37.88%	38.14%
			600	**370**	**837**	**61.86%**	**41.75%**	**49.85%**	**45.77%**
		Pothole	294	142	288	67.43%	50.52%	57.76%	55.99%
		Alligator Crack	104	76	75	57.78%	58.10%	57.94%	57.56%
4000	1.776204	Lateral Crack	85	124	163	40.67%	34.27%	37.20%	31.03%
		Longitudinal Crack	154	66	274	70.00%	35.98%	47.53%	49.58%
			637	**408**	**800**	**60.96%**	**44.33%**	**51.33%**	**48.54%**
		Pothole	305	171	277	64.08%	52.41%	57.66%	56.30%
		Alligator Crack	105	47	74	69.08%	58.66%	63.44%	61.99%
5000	1.160109	Lateral Crack	105	94	143	52.76%	42.34%	46.98%	40.78%
		Longitudinal Crack	193	114	235	62.87%	45.09%	52.52%	47.74%
			708	**426**	**729**	**62.43%**	**49.27%**	**55.08%**	**51.70%**
		Pothole	361	157	221	69.69%	62.03%	65.64%	63.21%
		Alligator Crack	105	58	74	64.42%	58.66%	61.40%	59.76%
6000	1.780058	Lateral Crack	102	78	146	56.67%	41.13%	47.66%	42.95%
		Longitudinal Crack	177	76	251	69.96%	41.36%	51.98%	50.67%
			745	**369**	**692**	**66.88%**	**51.84%**	**58.41%**	**54.15%**
		Pothole	393	148	189	72.64%	67.53%	69.99%	66.11%
		Alligator Crack	118	59	61	66.67%	65.92%	66.29%	62.40%
7000	0.666198	Lateral Crack	134	101	114	57.02%	54.03%	55.49%	47.60%
		Longitudinal Crack	223	128	205	63.53%	52.10%	57.25%	52.62%
			868	**436**	**569**	**66.56%**	**60.40%**	**63.33%**	**57.18%**
		Pothole	407	222	175	64.71%	69.93%	67.22%	65.76%
		Alligator Crack	128	73	51	63.68%	71.51%	67.37%	66.11%
8000	1.123044	Lateral Crack	126	118	122	51.64%	50.81%	51.22%	41.40%
		Longitudinal Crack	236	134	192	63.78%	55.14%	59.15%	57.18%
			897	**547**	**540**	**62.12%**	**62.42%**	**62.27%**	**57.61%**
		Pothole	424	150	158	73.87%	72.85%	73.36%	70.80%
		Alligator Crack	129	66	50	66.15%	72.07%	68.98%	68.85%
9000	1.208502	Lateral Crack	131	114	117	53.47%	52.82%	53.14%	45.18%
		Longitudinal Crack	238	152	190	61.03%	55.61%	58.19%	56.24%
			922	**482**	**515**	**65.67%**	**64.16%**	**64.91%**	**60.27%**
		Pothole	413	131	169	75.92%	70.96%	73.36%	71.46%
		Alligator Crack	128	60	51	68.09%	71.51%	69.75%	68.04%
10000	0.9868	Lateral Crack	144	105	104	57.83%	58.06%	57.95%	54.33%
		Longitudinal Crack	241	117	187	67.32%	56.31%	61.32%	56.87%
			926	**413**	**511**	**69.16%**	**64.44%**	**66.71%**	**62.68%**

Pada skenario pertama, kinerja terbaik didapatkan pada iterasi ke 9302 dengan nilai *mAP* sebesar 61.14%, nilai presisi sebesar 66.89%, nilai *recall* sebesar 63.26%, dan *f1 score* sebesar 65.02%. Kemudian pada skenario kedua, kinerja terbaik didapatkan pada iterasi ke 10000 dengan nilai *mAP* sebesar 62.68%, nilai presisi sebesar 69.16%, nilai *recall* sebesar 64.44%, dan nilai *f1 score* sebesar 66.71%.

Namun dari kedua *trained weight* tersebut, rata-rata IOU dari seluruh deteksi pada *trained weight* terbaik di skenario pertama sebesar 49.88%, dan rata-rata IOU dari seluruh deteksi pada *trained weight* terbaik di skenario kedua sebesar 53.30%. Perlu diperhatikan bahwa rata-rata IOU dari seluruh deteksi yang didapatkan berada dikisaran 50%, sehingga penerapan *threshold* sebesar 0.5 cukup tinggi.

Jika menurunkan *threshold* dari 0.5 menjadi 0.3, kinerja yang didapatkan sebagai berikut

Tabel 5.3: Hasil Pengujian *Trained Weight* Dengan $Threshold = 0.5$ dan $Threshold = 0.3$

416 x 416			
Iteration	Class	$AP_{0.3}$	$AP_{0.5}$
9302	Pothole	83.07%	70.16%
	Alligator Crack	80.65%	67.71%
	Lateral Crack	75.75%	51.72%
	Longitudinal Crack	68.61%	54.98%
		77.02%	**61.14%**

512 x 512			
Iteration	Class	$AP_{0.3}$	$AP_{0.5}$
10000	Pothole	84.97%	71.46%
	Alligator Crack	83.44%	68.04%
	Lateral Crack	77.37%	54.33%
	Longitudinal Crack	70.44%	56.87%
		79.06%	**62.68%**

Terlihat bahwa *mean average precision* dengan *threshold* sebesar 0.3 dapat mencapai 77.02% pada skenario pertama dan 79.06% pada skenario kedua.

Dari kedua skenario tersebut, semakin kecilnya nilai *loss* tidak mengakibatkan semakin membaiknya kinerja *trained weight* dalam mengenali kerusakan jalan dengan data uji. Dengan mengacu hasil kinerja diatas, *trained weight* pada iterasi 9302 di skenario pertama dan iterasi 10000 di skenario kedua dipilih sebagai *trained weight* terbaik.

5.2 Pengujian YOLOv3 Dengan Data Video

Pada subbab ini akan dilakukan pengujian metode YOLOv3 dalam melakukan pengenalan kerusakan jalan dengan menggunakan data video yang sudah dilakukan proses ekstraksi *frame* sebelumnya. Tujuan dari tahapan ini untuk mengetahui kinerja metode YOLOv3 dalam melakukan pengenalan kerusakan jalan pada skenario-skenario yang sudah disiapkan.

5.2.1 Data Pengujian YOLOv3 Dengan Data Video

Data pengujian terdiri dari 6 video berukuran 1280×720 (720p) dengan total durasi 127 detik (3809 *frame*). Jumlah kerusakan jalan yang terdapat pada setiap video dapat dilihat pada Tabel 5.4.

Tabel 5.4: Jumlah Kerusakan Yang Terdapat Pada Setiap Video Uji

No Video	Durasi (Detik)	Total Frame	*Class*	Total Objek	Total Anotasi
1	25.0	751	Pothole	18	809
			Alligator Crack	0	0
			Lateral Crack	0	0
			Longitudinal Crack	0	0
				18	**809**
2	16.9	506	Pothole	11	672
			Alligator Crack	4	153
			Lateral Crack	0	0
			Longitudinal Crack	0	0
				15	**825**
3	58.2	1747	Pothole	1	111
			Alligator Crack	14	548
			Lateral Crack	0	0
			Longitudinal Crack	19	835
				34	**1494**
4	10.0	299	Pothole	5	260
			Alligator Crack	1	34
			Lateral Crack	1	16
			Longitudinal Crack	0	0
				7	**310**
5	12.0	360	Pothole	2	78
			Alligator Crack	0	0
			Lateral Crack	1	45
			Longitudinal Crack	0	0
				3	**123**
6	4.9	146	Pothole	2	0
			Alligator Crack	0	0
			Lateral Crack	2	44
			Longitudinal Crack	3	24
				7	**68**

5.2.2 Skenario Pengujian Metode YOLOv3 Dengan Data Video

Data Video uji akan diatur tingkat kecerahan dan kontrasnya sehingga menghasilkan sebuah data video uji baru dengan pengaturan tingkat kecerahan dan kontras. Dengan langkah tersebut akan didapatkan sembilan skenario yang masing-masing skenario memiliki perbedaan tingkat kecerahan dan kontras. Berikut merupakan detail dari setiap skenario:

1. Skenario 1: Data uji tanpa pengaturan kecerahan dan kontras
2. Skenario 2: Data uji dengan pengaturan kontras sebesar 50
3. Skenario 3: Data uji dengan pengaturan kontras sebesar -50
4. Skenario 4: Data uji dengan pengaturan kecerahan sebesar 50
5. Skenario 5: Data uji dengan pengaturan kecerahan sebesar 50 dan kontras sebesar 50
6. Skenario 6: Data uji dengan pengaturan kecerahan sebesar 50 dan kontras sebesar -50
7. Skenario 7: Data uji dengan pengaturan kecerahan sebesar -50
8. Skenario 8: Data uji dengan pengaturan kecerahan sebesar -50 dan kontras sebesar 50
9. Skenario 9: Data uji dengan pengaturan kecerahan sebesar -50 dan kontras sebesar -50

Pengukuran kecepatan metode YOLOv3 dilakukan pada dua kondisi, kondisi pertama ketika pengenalan dilakukan menggunakan CPU pada laptop, dan kondisi kedua dilakukan menggunakan GPU pada *google colaboratory*. Detail dari spesifikasi *hardware* yange digunakan dapat dilihat pada Tabel 5.5.

Tabel 5.5: Spesifkasi *Hardware* Dari Setiap Kondisi

Kondisi	Hardware	Spesifikasi
CPU	CPU	AMD Ryzen 3 2200 U, Base Clock 2.5ghz
	GPU	Radeon Vega 3 Graphics
	Memory	12gb DDR4 Dual Channel
	Disk Space	SSD 112gb
GPU	CPU	Intel(R) Xeon(R) CPU, Base Clock 2.30GHz
	GPU	Tesla P100-PCIE-16GB
	Memory	13gb
	Disk Space	68.40gb

Gambar 5.4: Contoh data yang digunakan dari setiap skenario

Pengujian dilakukan dengan menggunakan dua *trained weight* terbaik yang didapatkan pada saat tahapan sebelumnya. Untuk mengetahui kinerja dari YOLOv3 maka akan dicari nilai akurasi, presisi, *recall*, dan mAP sebagai tolak ukur dengan *threshold* sebesar 0.3.

5.2.3 Hasil Pengujian Metode YOLOv3 Dengan Data Video

Setelah dilakukan pengujian terhadap metode YOLOv3, hasil pengujian dari setiap video dapat dilihat pada Lampiran 1 - Lampiran 6. Rangkuman hasil pengujian dari setiap skenario dapat dilihat pada Tabel 5.6 dan Tabel 5.7.

Meninjau hasil pengujian dari setiap video, dapat diketahui bahwa *average precision* tertinggi sebesar 90.38% pada pengenalan kerusakan *lateral crack* dengan ukuran *input* sebesar 512 × 512 di video ke-5. Namun jika melihat keseluruhan hasil pengujian, terlihat bahwa pengenalan kerusakan *pothole* memili kinerja yang lebih tinggi jika dibandingkan dengan pengenalan kerusakan lainnya. Dari hasil pengujian juga dapat diketahui bahwa model masih sering menghasilkan *false positive* dan *false negative*.

Gambar 5.5: Contoh dari Hasil Pengenalan Kerusakan Jalan Dengan YOLOv3 Pada Skenario 1

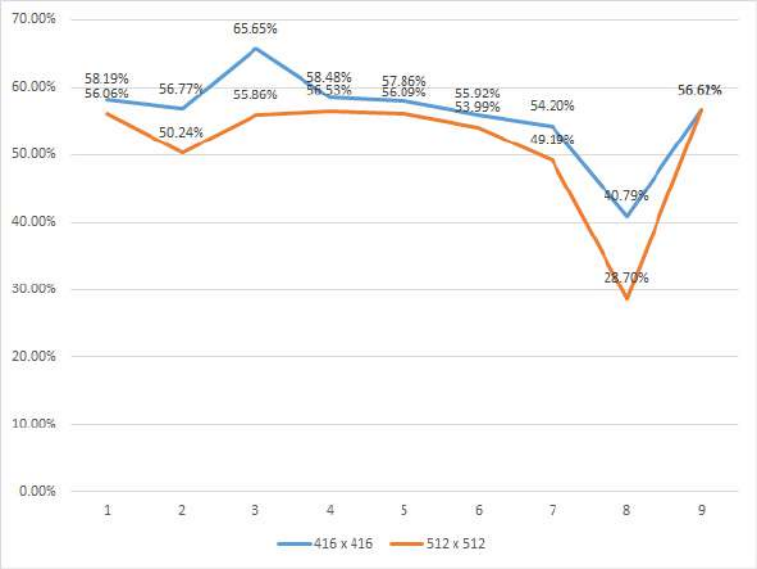
Tabel 5.6: Hasil Pengujian Metode YOLOv3 Dari Setiap Skenario Dengan Ukuran Input 416 × 416

Skenario	Class	416 x 416 Confusion Matrix			Precision	Recall	F1 Score
		TP	FP	FN			
1	Pothole	1255	520	676	70.70%	64.99%	67.73%
	Alligator Crack	491	337	244	59.30%	66.80%	62.83%
	Lateral Crack	51	39	54	56.67%	48.57%	52.31%
	Longitudinal Crack	123	153	736	44.57%	14.32%	21.67%
		1920	**1049**	**1710**	**64.67%**	**52.89%**	**58.19%**
2	Pothole	1322	611	708	68.39%	65.12%	66.72%
	Alligator Crack	470	348	265	57.46%	63.95%	60.53%
	Lateral Crack	30	17	75	63.83%	28.57%	39.47%
	Longitudinal Crack	127	212	732	37.46%	14.78%	21.20%
		1949	**1188**	**1780**	**62.13%**	**52.27%**	**56.77%**
3	Pothole	1182	397	748	74.86%	61.24%	67.37%
	Alligator Crack	468	279	142	62.65%	76.72%	68.98%
	Lateral Crack	59	61	46	49.17%	56.19%	52.44%
	Longitudinal Crack	118	146	93	44.70%	55.92%	49.68%
		1827	**883**	**1029**	**67.42%**	**63.97%**	**65.65%**
4	Pothole	1194	396	736	75.09%	61.87%	67.84%
	Alligator Crack	488	271	247	64.30%	66.39%	65.33%
	Lateral Crack	59	51	46	53.64%	56.19%	54.88%
	Longitudinal Crack	111	135	748	45.12%	12.92%	20.09%
		1852	**853**	**1777**	**68.47%**	**51.03%**	**58.48%**
5	Pothole	1210	449	720	72.94%	62.69%	67.43%
	Alligator Crack	501	316	234	61.32%	68.16%	64.56%
	Lateral Crack	42	46	63	47.73%	40.00%	43.52%
	Longitudinal Crack	113	144	746	43.97%	13.15%	20.25%
		1866	**955**	**1763**	**66.15%**	**51.42%**	**57.86%**
6	Pothole	1105	327	825	77.16%	57.25%	65.73%
	Alligator Crack	446	244	289	64.64%	60.68%	62.60%
	Lateral Crack	66	68	39	49.25%	62.86%	55.23%
	Longitudinal Crack	92	135	767	40.53%	10.71%	16.94%
		1709	**774**	**1920**	**68.83%**	**47.09%**	**55.92%**
7	Pothole	1145	606	785	65.39%	59.33%	62.21%
	Alligator Crack	429	366	186	53.96%	69.76%	60.85%
	Lateral Crack	31	20	74	60.78%	29.52%	39.74%
	Longitudinal Crack	162	252	697	39.13%	18.86%	25.45%
		1767	**1244**	**1742**	**58.68%**	**50.36%**	**54.20%**
8	Pothole	802	417	1218	65.79%	39.70%	49.52%
	Alligator Crack	226	138	509	62.09%	30.75%	41.13%
	Lateral Crack	18	17	87	51.43%	17.14%	25.71%
	Longitudinal Crack	77	92	782	45.56%	8.96%	14.98%
		1123	**664**	**2596**	**62.84%**	**30.20%**	**40.79%**
9	Pothole	1167	455	763	71.95%	60.47%	65.71%
	Alligator Crack	471	323	264	59.32%	64.08%	61.61%
	Lateral Crack	39	44	66	46.99%	37.14%	41.49%
	Longitudinal Crack	182	251	683	42.03%	21.04%	28.04%
		1859	**1073**	**1776**	**63.40%**	**51.14%**	**56.62%**

Tabel 5.7: Hasil Pengujian Metode YOLOv3 Dari Setiap Skenario Dengan Ukuran Input 512 × 512

Skenario	Class	512 x 512 Confusion Matrix TP	FP	FN	Precision	Recall	F1 Score
1	Pothole	1201	448	729	72.83%	62.23%	67.11%
	Alligator Crack	409	347	326	54.10%	55.65%	54.86%
	Lateral Crack	41	51	64	44.57%	39.05%	41.62%
	Longitudinal Crack	158	170	701	48.17%	18.39%	26.62%
		1809	**1016**	**1820**	**64.04%**	**49.85%**	**56.06%**
2	Pothole	1024	435	906	70.19%	53.06%	60.43%
	Alligator Crack	310	185	425	62.63%	42.18%	50.41%
	Lateral Crack	24	21	81	53.33%	22.86%	32.00%
	Longitudinal Crack	133	168	732	44.19%	15.38%	22.81%
		1491	**809**	**2144**	**64.83%**	**41.02%**	**50.24%**
3	Pothole	1181	319	749	78.73%	61.19%	68.86%
	Alligator Crack	371	267	364	58.15%	50.48%	54.04%
	Lateral Crack	63	61	42	50.81%	60.00%	55.02%
	Longitudinal Crack	105	162	754	39.33%	12.22%	18.65%
		1720	**809**	**1909**	**68.01%**	**47.40%**	**55.86%**
4	Pothole	1180	332	750	78.04%	61.14%	68.56%
	Alligator Crack	410	304	325	57.42%	55.78%	56.59%
	Lateral Crack	54	70	51	43.55%	51.43%	47.16%
	Longitudinal Crack	118	137	741	46.27%	13.74%	21.18%
		1762	**843**	**1867**	**67.64%**	**48.55%**	**56.53%**
5	Pothole	1191	416	729	74.11%	62.03%	67.54%
	Alligator Crack	407	331	328	55.15%	55.37%	55.26%
	Lateral Crack	47	45	58	51.09%	44.76%	47.72%
	Longitudinal Crack	141	171	718	45.19%	16.41%	24.08%
		1786	**963**	**1833**	**64.97%**	**49.35%**	**56.09%**
6	Pothole	1073	258	857	80.62%	55.60%	65.81%
	Alligator Crack	392	278	343	58.51%	53.33%	55.80%
	Lateral Crack	52	46	53	53.06%	49.52%	51.23%
	Longitudinal Crack	77	100	782	43.50%	8.96%	14.86%
		1594	**682**	**2035**	**70.04%**	**43.92%**	**53.99%**
7	Pothole	1018	397	942	71.94%	51.94%	60.33%
	Alligator Crack	283	167	452	62.89%	38.50%	47.76%
	Lateral Crack	27	32	78	45.76%	25.71%	32.93%
	Longitudinal Crack	117	175	742	40.07%	13.62%	20.33%
		1445	**771**	**2214**	**65.21%**	**39.49%**	**49.19%**
8	Pothole	564	202	1366	73.63%	29.22%	41.84%
	Alligator Crack	73	12	662	85.88%	9.93%	17.80%
	Lateral Crack	15	32	90	31.91%	14.29%	19.74%
	Longitudinal Crack	1	22	858	4.35%	0.12%	0.23%
		653	**268**	**2976**	**70.90%**	**17.99%**	**28.70%**
9	Pothole	1249	454	681	73.34%	64.72%	68.76%
	Alligator Crack	410	362	325	53.11%	55.78%	54.41%
	Lateral Crack	29	61	76	32.22%	27.62%	29.74%
	Longitudinal Crack	182	231	677	44.07%	21.19%	28.62%
		1870	**1108**	**1759**	**62.79%**	**51.53%**	**56.61%**

Untuk melakukan pengenalan kerusakan jalan diperlukan juga pencahayaan yang baik, hal ini dapat dilihat dari kinerja metode YOLOv3 dalam melakukan pengenalan pada skenario dengan pencahayaan kurang (skenario 7, skenario 8, dan skenario 9) mendapatkan hasil yang rendah jika dibandingkan dengan skenario yang lain. Skenario 3 dengan ukuran input 416×416 memiliki hasil pengujian terbaik jika dibandingkan dengan skenario lainnya berdasarkan nilai *f1-score*.



Gambar 5.6: Grafik nilai *f1-score* dari setiap skenario

Meninjau Gambar 5.6, meningkatkan ukuran input dari 416×416 menjadi 512×512 tidak memperlihatkan perbaikan kinerja jika meninjau pada nilai *f1-score*, hanya beberapa kerusakan pada beberapa skenario yang menunjukkan nilai *f1-score* yang lebih baik.

Setelah mendapatkan hasil pengenalan pada setiap *frame*

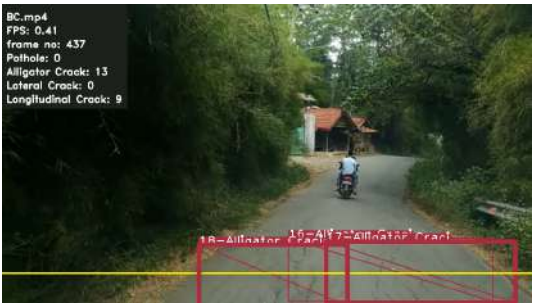
di seluruh data uji, kemudian akan dilakukan pengujian perhitungan kerusakan jalan dan kecepatan pengenalan dari YOLOv3. Berikut merupakan hasil pengujian perhitungan kerusakan jalan dan kecepatan pengenalan pada skenario 1 dapat dilihat pada Tabel 5.8.

Tabel 5.8: Hasil Perhitungan Kerusakan dan Kecepatan Deteksi YOLOv3

No Video	Total Kerusakan		Jumlah Objek Yang Berhasil Dihitung		Kecepatan Pengenalan			
	Class	Jumlah	416 x 416	512 x 512	416 x 416 CPU	416 x 416 GPU	512 x 512 CPU	512 x 512 GPU
1	Pothole	**18**	18	16	0.44	40.40	0.48	39.20
	Alligator Crack	**0**	3	0				
	Lateral Crack	**0**	0	0				
	Longitudinal Crack	**0**	0	0				
2	Pothole	**11**	15	22	0.43	41.90	0.53	41.30
	Alligator Crack	**4**	2	0				
	Lateral Crack	**0**	1	1				
	Longitudinal Crack	**0**	0	0				
3	Pothole	**1**	1	2	0.47	38.40	0.51	39.00
	Alligator Crack	**14**	36	56				
	Lateral Crack	**0**	0	0				
	Longitudinal Crack	**19**	33	52				
4	Pothole	**5**	6	5	0.45	34.50	0.54	35.80
	Alligator Crack	**1**	1	3				
	Lateral Crack	**1**	1	1				
	Longitudinal Crack	**0**	0	0				
5	Pothole	**2**	3	2	0.43	33.00	0.50	32.50
	Alligator Crack	**0**	0	0				
	Lateral Crack	**1**	2	1				
	Longitudinal Crack	**0**	0	0				
6	Pothole	**0**	0	0	0.40	44.00	0.47	37.60
	Alligator Crack	**0**	0	0				
	Lateral Crack	**2**	1	1				
	Longitudinal Crack	**3**	2	2				

Dari hasil tersebut terdapat beberapa nilai yang lebih besar atau lebih kecil dari nilai sesungguhnya, hasil perhitungan yang lebih sedikit dari nilai sesungguhnya dapat terjadi dikarenakan objek yang dikenali tidak memotong garis ROI sehingga objek gagal dihitung. Untuk hasil perhitungan yang lebih tinggi dari nilai sesungguhnya dapat

terjadi dikarenakan YOLOv3 tidak melakukan pengenalan kerusakan jalan dengan stabil sehingga *tracker* mengaggap objek tersebut merupakan sebuah objek yang baru. Salah satu permasalahan lain dalam melakukan pengenalan kerusakan jalan terjadi saat melakukan pengenalan *alligator crack*. Dalam beberapa kasus saat melakukan pengenalan *alligator crack*, kerusakan *longitudinal crack* dan *lateral crack* yang menyusun kerusakan *alligator crack* cenderung ikut dikenali sebagai kerusakan *longitudinal crack* dan *lateral crack*.



Gambar 5.7: Contoh Terjadinya Pengenalan Ganda Terhadap Satu Objek Kerusakan Jalan

Penggunaan GPU dalam melakukan pengenalan kerusakan jalan sangat mempengaruhi kecepatan pengenalan. Pada seluruh video, terlihat bahwa penggunaan GPU dapat memberikan pengingkatan kecepatan yang signifikan jika dibandingkan dengan penggunaan CPU. Ukuran input yang digunakan akan mempengahuri kecepatan pengenalan kerusakan jalan. Semakin besar ukuran input yang digunakan, maka semakin lambat kecepatan pengenalan objeknya. Dari hasil tersebut juga dapat diketahui bahwa kecepatan pengenalan pada metode YOLOv3 dapat mencapai angka 44.40 FPS sehingga penggunaan GPU pada YOLOv3 sangat direkomendasikan.

Gambar 5.8: Contoh Permasalahan Saat Melakukan Pengenalan Kerusakan *Alligator Crack*

Gambar 5.9: Contoh Kerusakan *Lateral Crack* Yang Gagal Dihitung

Halaman ini sengaja dikosongkan

BAB VI
PENUTUP

Pada bab ini berisikan kesimpulan yang diperolah dalam pengerjaan Tugas Akhir. Selain kesimpulan, diberikan juga saran-saran yang diharapkan dapat menjadi masukan untuk pengembangan penelitian selanjutnya.

6.1 Kesimpulan

Berdasarkan proses dan hasil yang didapatkan selama pengerjaan Tugas Akhir, maka dapat diambil kesimpulan sebagai berikut:

1. Metode YOLOv3 dapat melakukan pengenalan kerusakan jalan pada data video dengan diawali melakukan pelatihan untuk mendapatkan *trained weight* dengan menggunakan data kerusakan jalan yang sudah diberikan anotasi. Setelah didapatkan *trained weight*, maka tahapan selanjutnya adalah melakukan pengenalan kerusakan jalan dengan langkah pertama yaitu memberikan *input* data video, kemudian untuk setiap *frame* akan dikenali kerusakan jalan dengan menggunakan YOLOv3 dan dilanjutkan dengan *tracking*.

2. Metode YOLOv3 mampu untuk melakukan pengenalan terhadap kerusakan jalan dengan nilai *mean average precision* tertinggi sebesar 79.06% pada 665 data uji dengan 1437 kerusakan jalan. Kecepatan pengenalan tertinggi yang didapatkan dengan menggunakan metode YOLOv3 dapat mencapai 44.40 FPS.

6.2 Saran

Berdasarkan pengamatan yang dilakukan selama mengerjakan Tugas Akhir, terdapat beberapa saran sebagai masukan untuk penelitian berikutnya:

1. Menambahkan jumlah data dengan jenis kerusakan, sudut pandang, dan pencahayaan yang lebih bervariatif agar kinerja YOLOv3 jauh lebih baik. Penambahan data yang digunakan akan berdampak sangat signifikan terhadap kinerja YOLOv3.

2. Diperlukan spesifikisai perangkat keras yang tinggi untuk mendapatkan kecepatan pengenalan kerusakan jalan yang tinggi.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] "Pavement distresses," Jan 2019. [Online]. Available: https://pavementinteractive.org/reference-desk/pavement-management/pavement-distresses/

[2] R. C. Gonzales and R. E. Woods, *Digital Image Processing Third Edition*. Pearson, 2008.

[3] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *CoRR*, vol. abs/1511.08458, 2015. [Online]. Available: http://arxiv.org/abs/1511.08458

[4] J. Redmon, S. K. Divvala, R. B. Girshick *et al.*, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015. [Online]. Available: http://arxiv.org/abs/1506.02640

[5] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018. [Online]. Available: http://arxiv.org/abs/1804.02767

[6] K. Schwab, *The Global Competitiveness Report 2019*. Geneva: World Economic Forum, 2019.

[7] PUPR, *Buku Informasi Statistik Pekerjaan Umum dan Perumahan Rakyat*. Jakarta: Kementrian PUPR Sekretariat Jenderal Pusat Data dan Teknologi Informasi, 2017.

[8] *Prosedur Pemeliharaan Jalan SOP/UPM/DJBM-12.* Jakarta: Kementrian PUPR Sekretariat Jenderal Pusat Data dan Teknologi Informasi, 2016.

[9] S. Nienaber, M. T. Booysen, and R. Kroon, "Detecting potholes using simple image processing techniques and real-world footage," 07 2015.

[10] J. Walsh, N. O' Mahony, S. Campbell *et al.*, "Deep learning vs. traditional computer vision," 04 2019.

[11] H. Maeda, Y. Sekimoto, T. Seto *et al.*, "Road damage detection and classification using deep neural networks with smartphone images: Road damage detection and classification," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, 06 2018.

[12] Y.-J. Cha, W. Choi, and O. Büyüköztürk, "Deep learning-based crack damage detection using convolutional neural networks," *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 5, pp. 361–378, mar 2017.

[13] S. L. H. Oktavian, "Penerapan metode *Faster Regional Network (Faster R-CNN)* untuk klasifikasi kerusakan jalan," 2020.

[14] S. Dwi Ratna, O. Daniel, P. Ravy Hayu *et al.*, "Pavement distress classification using deep learning method based on digital image," in *Proceedings of the 2nd International Symposium on Transportation Studies in Developing Countries (ISTSDC 2019).* Atlantis Press, 2020, pp. 143–146. [Online]. Available: https://doi.org/10.2991/aer.k.200220.030

[15] S. Sukirman, *Perkerasan Lentur Jalan Raya.* Bandung: Nova, 1999.

[16] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, May 2015. [Online]. Available: https://doi.org/10.1038/nature14539

[17] D. Wang, C. Li, S. Wen *et al.*, "Daedalus: Breaking non-maximum suppression in object detection via adversarial examples," *CoRR*, vol. abs/1902.02067, 2019. [Online]. Available: http://arxiv.org/abs/1902.02067

[18] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 6517–6525.

[19] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[20] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427 – 437, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S030645730900025

[21] M. Everingham, L. Van Gool, C. K. I. Williams *et al.*, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun 2010. [Online]. Available: https://doi.org/10.1007/s11263-009-0275-4

[22] R. Seyed Hamid, T. Nathan, G. JunYoung *et al.*, "Generalized intersection over union: A metric and A loss for bounding box regression," *CoRR*, vol. abs/1902.09630, 2019. [Online]. Available: http://arxiv.org/abs/1902.09630

Lampiran 1: Hasil Pengujian Metode YOLOv3 Pada Video 1

Skenario	Class	416 x 416							512x512						
		Confusion Matrix			Precision	Recall	F1 Score	$AP_{0.5}$	Confusion Matrix			Precision	Recall	Recall	$AP_{0.5}$
		TP	FP	FN					TP	FP	FN				
1	Pothole	562	151	247	78.82%	69.47%	73.85%	76.57%	572	154	237	78.79%	70.70%	74.53%	75.39%
	Alligator Crack	0	42	0	0.00%	-	-	0.00%	0	10	0	0.00%	-	-	0.00%
	Lateral Crack	0	2	0	0.00%	-	-	0.00%	0	1	0	0.00%	-	-	0.00%
	Longitudinal Crack	0	1	0	0.00%	-	-	0.00%	0	0	0	-	-	-	-
2	Pothole	670	149	239	81.81%	73.71%	77.55%	79.31%	560	139	249	80.11%	69.22%	74.27%	72.95%
	Alligator Crack	0	16	0	0.00%	-	-	0.00%	0	0	0	-	-	-	-
	Lateral Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
	Longitudinal Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
3	Pothole	543	118	266	82.15%	67.12%	73.88%	73.79%	550	106	259	83.84%	67.99%	75.09%	76.27%
	Alligator Crack	0	20	0	0.00%	-	-	0.00%	0	15	0	0.00%	-	-	0.00%
	Lateral Crack	0	7	0	0.00%	-	-	0.00%	0	5	0	0.00%	-	-	0.00%
	Longitudinal Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
4	Pothole	533	108	276	83.15%	65.88%	73.52%	74.01%	554	103	255	84.32%	68.48%	75.58%	75.14%
	Alligator Crack	0	32	0	0.00%	-	-	0.00%	0	17	0	0.00%	-	-	0.00%
	Lateral Crack	0	7	0	0.00%	-	-	0.00%	0	4	0	0.00%	-	-	0.00%
	Longitudinal Crack	0	0	0	-	-	-	-	0	1	0	0.00%	-	-	0.00%
5	Pothole	532	124	277	81.10%	65.76%	72.63%	75.06%	556	144	243	79.43%	69.59%	74.18%	74.08%
	Alligator Crack	0	43	0	0.00%	-	-	0.00%	0	12	0	0.00%	-	-	0.00%
	Lateral Crack	0	8	0	0.00%	-	-	0.00%	0	1	0	0.00%	-	-	0.00%
	Longitudinal Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
6	Pothole	518	100	291	83.82%	64.03%	72.60%	71.11%	511	78	298	86.76%	63.16%	73.10%	72.61%
	Alligator Crack	0	10	0	0.00%	-	-	0.00%	0	12	0	0.00%	-	-	0.00%
	Lateral Crack	0	9	0	0.00%	-	-	0.00%	0	4	0	0.00%	-	-	0.00%
	Longitudinal Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
7	Pothole	558	165	251	77.18%	68.97%	72.85%	76.72%	581	143	228	80.25%	71.82%	75.80%	76.17%
	Alligator Crack	0	27	0	0.00%	-	-	0.00%	0	1	0	0.00%	-	-	0.00%
	Lateral Crack	0	1	0	0.00%	-	-	0.00%	0	0	0	-	-	-	-
	Longitudinal Crack	0	0	0	-	-	-	-	0	2	0	0.00%	-	-	0.00%
8	Pothole	591	165	308	78.17%	65.74%	71.42%	68.32%	462	102	347	81.91%	57.11%	67.30%	68.70%
	Alligator Crack	0	39	0	0.00%	-	-	0.00%	0	3	0	0.00%	-	-	0.00%
	Lateral Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
	Longitudinal Crack	0	1	0	0.00%	-	-	0.00%	0	0	0	-	-	-	-
9	Pothole	553	146	256	79.11%	68.36%	73.34%	74.28%	600	157	209	79.26%	74.17%	76.63%	78.83%
	Alligator Crack	0	13	0	0.00%	-	-	0.00%	0	24	0	0.00%	-	-	0.00%
	Lateral Crack	0	1	0	0.00%	-	-	0.00%	0	2	0	0.00%	-	-	0.00%
	Longitudinal Crack	0	1	0	0.00%	-	-	0.00%	0	3	0	0.00%	-	-	0.00%

Lampiran 2: Hasil Pengujian Metode YOLOv3 Pada Video 2

Skenario	Class	416 x 416							512x512						
		Confusion Matrix			Precision	Recall	F1 Score	$AP_{0.5}$	Confusion Matrix			Precision	Recall	F1 Score	$AP_{0.5}$
		TP	FP	FN					TP	FP	FN				
1	Pothole	356	300	316	54.27%	52.98%	53.61%	49.45%	281	186	391	60.17%	41.82%	49.34%	42.82%
	Alligator Crack	34	3	119	91.89%	22.22%	35.79%	40.22%	7	12	146	36.84%	4.58%	8.14%	21.45%
	Lateral Crack	0	1	0	0.00%	-	-	0.00%	0	12	0	0.00%	-	-	0.00%
	Longitudinal Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
2	Pothole	326	334	346	49.39%	48.51%	48.95%	45.92%	190	180	482	51.35%	28.27%	36.47%	33.14%
	Alligator Crack	37	8	116	82.22%	24.18%	37.37%	39.95%	3	6	150	33.33%	1.96%	3.70%	10.16%
	Lateral Crack	0	2	0	0.00%	-	-	0.00%	0	0	0	-	-	-	-
	Longitudinal Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
3	Pothole	321	235	351	57.73%	47.77%	52.28%	49.50%	285	155	387	64.77%	42.41%	51.26%	48.80%
	Alligator Crack	36	6	117	85.71%	23.53%	36.92%	38.01%	10	12	143	45.45%	6.54%	11.43%	34.55%
	Lateral Crack	0	3	0	0.00%	-	-	0.00%	0	18	0	0.00%	-	-	0.00%
	Longitudinal Crack	0	6	0	0.00%	-	-	0.00%	0	0	0	-	-	-	-
4	Pothole	330	242	342	57.69%	49.11%	53.05%	51.20%	284	158	388	64.25%	42.26%	50.99%	48.31%
	Alligator Crack	37	6	116	86.05%	24.18%	37.76%	39.67%	9	13	144	40.91%	5.88%	10.29%	35.57%
	Lateral Crack	0	0	0	-	-	-	-	0	15	0	0.00%	-	-	0.00%
	Longitudinal Crack	0	2	0	0.00%	-	-	0.00%	0	0	0	-	-	-	-
5	Pothole	353	262	319	57.40%	52.53%	54.86%	52.20%	289	177	383	62.02%	43.01%	50.79%	46.32%
	Alligator Crack	46	3	107	93.88%	30.07%	45.54%	43.45%	9	10	144	47.37%	5.88%	10.47%	35.49%
	Lateral Crack	0	1	0	0.00%	-	-	0.00%	0	5	0	0.00%	-	-	0.00%
	Longitudinal Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
6	Pothole	285	195	387	59.38%	42.41%	49.48%	48.01%	248	135	424	64.75%	36.90%	47.01%	45.83%
	Alligator Crack	34	22	119	60.71%	22.22%	32.54%	34.21%	24	19	129	55.81%	15.69%	24.49%	37.78%
	Lateral Crack	0	1	0	0.00%	-	-	0.00%	0	4	0	0.00%	-	-	0.00%
	Longitudinal Crack	0	2	0	0.00%	-	-	0.00%	0	0	0	-	-	-	-
7	Pothole	273	292	399	48.32%	40.63%	44.14%	41.13%	157	155	515	50.32%	23.36%	31.91%	30.03%
	Alligator Crack	20	9	13	68.97%	60.61%	64.52%	23.26%	0	8	153	0.00%	0.00%	-	3.92%
	Lateral Crack	0	1	0	0.00%	-	-	0.00%	0	1	0	0.00%	-	-	0.00%
	Longitudinal Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
8	Pothole	110	193	562	36.30%	16.37%	22.56%	14.41%	60	84	612	41.67%	8.93%	14.71%	9.36%
	Alligator Crack	3	12	150	20.00%	1.96%	3.57%	3.76%	1	4	152	20.00%	0.65%	1.27%	2.36%
	Lateral Crack	0	0	0	-	-	-	-	0	8	0	0.00%	-	-	0.00%
	Longitudinal Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
9	Pothole	283	227	389	55.49%	42.11%	47.88%	48.04%	288	191	384	60.13%	42.86%	50.04%	78.83%
	Alligator Crack	26	2	127	92.86%	16.99%	28.73%	32.68%	5	13	148	27.78%	3.27%	5.85%	0.00%
	Lateral Crack	0	2	0	0.00%	-	-	0.00%	0	25	0	0.00%	-	-	0.00%
	Longitudinal Crack	0	3	0	0.00%	-	-	0.00%	0	1	0	0.00%	-	-	0.00%

Lampiran 3: Hasil Pengujian Metode YOLOv3 Pada Video 3

Skenario	Class	416 x 416							512x512						
		Confusion Matrix			Precision	Recall	F1 Score	$AP_{0.5}$	Confusion Matrix			Precision	Recall	F1 Score	$AP_{0.5}$
		TP	FP	FN					TP	FP	FN				
1	Pothole	96	27	15	78.05%	86.49%	82.05%	79.49%	108	32	3	77.14%	97.30%	86.06%	84.21%
	Alligator Crack	439	282	109	60.89%	80.11%	69.19%	79.72%	388	304	160	56.07%	70.80%	62.58%	67.82%
	Lateral Crack	0	6	0	0.00%	-	-	0.00%	0	12	0	0.00%	-	-	0.00%
	Longitudinal Crack	118	151	717	43.87%	14.13%	21.38%	14.59%	155	170	680	47.69%	18.56%	26.72%	21.45%
2	Pothole	91	40	20	69.47%	81.98%	75.21%	70.79%	101	38	10	72.66%	90.99%	80.80%	78.81%
	Alligator Crack	423	318	125	57.09%	77.19%	65.63%	77.22%	305	170	243	64.21%	55.66%	59.63%	64.81%
	Lateral Crack	0	4	0	0.00%	-	-	0.00%	0	11	0	0.00%	-	-	0.00%
	Longitudinal Crack	122	212	713	36.53%	14.61%	20.87%	10.98%	129	168	706	43.43%	15.45%	22.79%	14.61%
3	Pothole	90	22	21	80.36%	81.08%	80.72%	80.08%	101	18	10	84.87%	90.99%	87.83%	88.75%
	Alligator Crack	410	240	13	63.08%	96.93%	76.42%	79.26%	340	214	208	61.37%	62.04%	61.71%	65.61%
	Lateral Crack	0	9	0	0.00%	-	-	0.00%	0	5	0	0.00%	-	-	0.00%
	Longitudinal Crack	116	139	71	45.49%	62.03%	52.49%	16.06%	104	161	731	39.25%	12.46%	18.91%	11.35%
4	Pothole	90	21	21	81.08%	81.08%	81.08%	79.77%	94	19	17	83.19%	84.68%	83.93%	85.07%
	Alligator Crack	429	222	119	65.90%	78.28%	71.56%	79.33%	379	252	169	60.06%	69.16%	64.29%	70.07%
	Lateral Crack	0	6	0	0.00%	-	-	0.00%	0	12	0	0.00%	-	-	0.00%
	Longitudinal Crack	108	132	727	45.00%	12.93%	20.09%	15.30%	116	134	719	46.40%	13.89%	21.38%	16.17%
5	Pothole	89	24	22	78.76%	80.18%	79.46%	52.20%	101	22	10	82.11%	90.99%	86.32%	84.21%
	Alligator Crack	437	261	111	62.61%	79.74%	70.14%	43.45%	379	285	169	57.08%	69.16%	62.54%	70.12%
	Lateral Crack	0	5	0	0.00%	-	-	0.00%	0	10	0	0.00%	-	-	0.00%
	Longitudinal Crack	106	140	729	43.09%	12.69%	19.61%	13.87%	132	168	703	44.00%	15.81%	23.26%	16.68%
6	Pothole	88	17	23	83.81%	79.28%	81.48%	81.69%	90	13	21	87.38%	81.08%	84.11%	85.74%
	Alligator Crack	389	193	159	66.84%	70.99%	68.85%	76.65%	346	193	202	64.19%	63.14%	63.66%	66.49%
	Lateral Crack	0	10	0	0.00%	-	-	0.00%	0	3	0	0.00%	-	-	0.00%
	Longitudinal Crack	91	129	744	41.36%	10.90%	17.25%	14.24%	76	98	759	43.68%	9.10%	15.06%	11.46%
7	Pothole	81	57	30	58.70%	72.97%	65.06%	61.08%	109	36	2	75.17%	98.20%	85.16%	78.46%
	Alligator Crack	400	320	148	55.56%	72.99%	63.09%	74.62%	281	148	267	65.50%	51.28%	57.52%	58.83%
	Lateral Crack	0	3	0	0.00%	-	-	0.00%	0	7	0	0.00%	-	-	0.00%
	Longitudinal Crack	156	252	679	38.24%	18.68%	25.10%	13.14%	111	173	724	39.08%	13.29%	19.84%	12.46%
8	Pothole	8	3	103	72.73%	7.21%	13.11%	41.11%	3	5	108	37.50%	2.70%	5.04%	9.62%
	Alligator Crack	220	82	328	72.85%	40.15%	51.76%	47,95%	72	4	476	94.74%	13.14%	23.08%	27.45%
	Lateral Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
	Longitudinal Crack	74	89	761	45.40%	8.86%	14.83%	9.62%	1	22	834	4.35%	0.12%	0.23%	2.56%
9	Pothole	94	34	17	73.44%	84.68%	78.66%	76.01%	108	37	3	74.48%	97.30%	84.38%	81.64%
	Alligator Crack	427	294	121	59.22%	77.92%	67.30%	78.84%	390	302	158	56.36%	71.17%	62.90%	67.65%
	Lateral Crack	0	21	0	0.00%	-	-	0.00%	0	10	0	0.00%	-	-	0.00%
	Longitudinal Crack	178	240	657	42.58%	21.32%	28.41%	16.63%	177	224	658	44.14%	21.20%	28.64%	21.42%

Lampiran 4: Hasil Pengujian Metode YOLOv3 Pada Video 4

Skenario	Class	416 x 416							512x512						
		Confusion Matrix			Precision	Recall	F1 Score	$AP_{0.5}$	Confusion Matrix			Precision	Recall	F1 Score	$AP_{0.5}$
		TP	FP	FN					TP	FP	FN				
1	Pothole	61	21	18	74.39%	77.22%	75.78%	81.87%	55	19	23	74.32%	70.51%	72.37%	76.90%
	Alligator Crack	0	2	0	0.00%	-	-	0.00%	0	0	0	-	-	-	-
	Lateral Crack	20	14	25	58.82%	44.44%	50.63%	78.24%	13	3	32	81.25%	28.89%	42.62%	52.59%
	Longitudinal Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
2	Pothole	43	37	35	53.75%	55.13%	54.43%	63.39%	39	31	39	55.71%	50.00%	52.70%	42.74%
	Alligator Crack	0	0	0	-	-	-	0.00%	0	0	0	-	-	-	-
	Lateral Crack	0	2	45	0.00%	0.00%	-	17.93%	0	4	45	0.00%	0.00%	-	1.09%
	Longitudinal Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
3	Pothole	51	13	27	79.69%	65.38%	71.83%	82.65%	46	7	32	86.79%	58.97%	70.23%	84.98%
	Alligator Crack	0	1	0	0.00%	-	-	0.00%	0	0	0	-	-	-	-
	Lateral Crack	31	26	14	54.39%	68.89%	60.78%	81.61%	33	14	12	70.21%	73.33%	71.74%	86.46%
	Longitudinal Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
4	Pothole	60	13	18	82.19%	76.92%	79.47%	86.77%	58	11	20	84.06%	74.36%	78.91%	89.75%
	Alligator Crack	0	2	0	0.00%	-	-	0.00%	0	0	0	-	-	-	-
	Lateral Crack	28	22	17	56.00%	62.22%	58.95%	85.84%	24	17	21	58.54%	53.33%	55.81%	83.37%
	Longitudinal Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
5	Pothole	63	22	15	74.12%	80.77%	77.30%	84.05%	62	23	16	72.94%	79.49%	76.07%	79.56%
	Alligator Crack	0	2	0	0.00%	-	-	0.00%	0	0	0	-	-	-	-
	Lateral Crack	15	13	30	53.57%	33.33%	41.10%	78.15%	17	7	28	70.83%	37.78%	49.28%	68.39%
	Longitudinal Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
6	Pothole	45	9	33	83.33%	57.69%	68.18%	79.99%	36	6	42	85.71%	46.15%	60.00%	65.06%
	Alligator Crack	0	0	0	-	-	-	-	0	24	0	0.00%	-	-	-
	Lateral Crack	36	25	9	59.02%	80.00%	67.92%	79.75%	27	17	18	61.36%	60.00%	60.67%	76.74%
	Longitudinal Crack	0	3	0	0.00%	-	-	0.00%	0	0	0	-	-	-	-
7	Pothole	40	35	38	53.33%	51.28%	52.29%	41.27%	27	32	51	45.76%	34.62%	39.42%	24.33%
	Alligator Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
	Lateral Crack	0	3	45	0.00%	0.00%	-	9.77%	0	4	45	0.00%	0.00%	-	0.00%
	Longitudinal Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
8	Pothole	0	6	78	0.00%	0.00%	-	78.00%	0	4	78	0.00%	0.00%	-	0.00%
	Alligator Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
	Lateral Crack	0	0	45	-	0.00%	-	45.00%	0	0	45	-	0.00%	-	0.00%
	Longitudinal Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
9	Pothole	40	17	38	70.18%	51.28%	59.26%	45.55%	40	12	38	76.92%	51.28%	61.54%	49.45%
	Alligator Crack	0	1	0	0.00%	-	-	0.00%	0	1	0	0.00%	-	-	0.00%
	Lateral Crack	10	6	35	62.50%	22.22%	32.79%	58.44%	0	0	45	-	0.00%	-	20.32%
	Longitudinal Crack	0	4	0	0.00%	-	-	0.00%	0	1	0	0.00%	-	-	0.00%

Lampiran 5: Hasil Pengujian Metode YOLOv3 Pada Video 5

Skenario	Class	416 x 416							512x512						
		Confusion Matrix			Precision	Recall	F1 Score	$AP_{0.5}$	Confusion Matrix			Precision	Recall	F1 Score	$AP_{0.5}$
		TP	FP	FN					TP	FP	FN				
1	Pothole	180	21	80	89.55%	69.23%	78.09%	85.95%	185	57	75	76.45%	71.15%	73.71%	75.94%
	Alligator Crack	18	8	16	69.23%	52.94%	60.00%	53.73%	14	21	20	40.00%	41.18%	40.58%	30.30%
	Lateral Crack	16	16	0	50.00%	100.00%	66.67%	85.18%	15	16	1	48.39%	93.75%	63.83%	90.38%
	Longitudinal Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
2	Pothole	192	51	68	79.01%	73.85%	76.34%	81.65%	134	47	126	74.03%	51.54%	60.77%	74.63%
	Alligator Crack	10	6	24	62.50%	29.41%	40.00%	35.75%	2	9	32	18.18%	5.88%	8.89%	4.41%
	Lateral Crack	11	9	5	55.00%	68.75%	61.11%	82.93%	5	2	11	71.43%	31.25%	43.48%	51.09%
	Longitudinal Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
3	Pothole	177	9	83	95.16%	68.08%	79.37%	85.55%	199	33	61	85.78%	76.54%	80.89%	84.94%
	Alligator Crack	22	12	12	64.71%	64.71%	64.71%	66.48%	21	26	13	44.68%	61.76%	51.85%	62.02%
	Lateral Crack	13	14	3	48.15%	81.25%	60.47%	76.33%	16	14	0	53.33%	100.00%	69.57%	87.13%
	Longitudinal Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
4	Pothole	181	12	79	93.78%	69.62%	79.91%	86.62%	190	41	70	82.25%	73.08%	77.39%	82.10%
	Alligator Crack	22	9	12	70.97%	64.71%	67.69%	67.54%	22	22	12	50.00%	64.71%	56.41%	69.98%
	Lateral Crack	13	14	3	48.15%	81.25%	60.47%	75.09%	15	16	1	48.39%	93.75%	63.83%	86.40%
	Longitudinal Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
5	Pothole	173	17	87	91.05%	66.54%	76.89%	86.59%	183	50	77	78.54%	70.38%	74.24%	75.87%
	Alligator Crack	18	7	16	72.00%	52.94%	61.02%	58.64%	19	24	15	44.19%	55.88%	49.35%	53.32%
	Lateral Crack	12	14	4	46.15%	75.00%	57.14%	63.44%	16	18	0	47.06%	100.00%	64.00%	84.31%
	Longitudinal Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
6	Pothole	169	6	91	96.57%	65.00%	77.70%	84.08%	188	26	72	87.85%	72.31%	79.32%	86.44%
	Alligator Crack	23	19	11	54.76%	67.65%	60.53%	67.89%	22	30	12	42.31%	64.71%	51.16%	64.49%
	Lateral Crack	12	18	4	40.00%	75.00%	52.17%	65.86%	14	14	2	50.00%	87.50%	63.64%	73.23%
	Longitudinal Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
7	Pothole	193	57	67	77.20%	74.23%	75.69%	75.68%	144	31	146	82.29%	49.66%	61.94%	70.75%
	Alligator Crack	9	10	25	47.37%	26.47%	33.96%	26.23%	2	10	32	16.67%	5.88%	8.70%	3.48%
	Lateral Crack	12	12	4	50.00%	75.00%	60.00%	79.99%	7	17	9	29.17%	43.75%	35.00%	48.40%
	Longitudinal Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
8	Pothole	93	50	167	65.03%	35.77%	46.15%	48.34%	39	7	221	84.78%	15.00%	25.49%	24.96%
	Alligator Crack	3	5	31	37.50%	8.82%	14.29%	12.46%	0	1	34	0.00%	0.00%	-	0.00%
	Lateral Crack	2	13	14	13.33%	12.50%	12.90%	15.25%	0	9	16	0.00%	0.00%	-	3.52%
	Longitudinal Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
9	Pothole	197	31	63	86.40%	75.77%	80.74%	88.82%	213	57	47	78.89%	81.92%	80.38%	81.76%
	Alligator Crack	18	13	16	58.06%	52.94%	55.38%	52.10%	15	22	19	40.54%	44.12%	42.25%	41.38%
	Lateral Crack	13	14	3	48.15%	81.25%	60.47%	69.45%	16	19	0	45.71%	100.00%	62.75%	72.69%
	Longitudinal Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-

Lampiran 6: Hasil Pengujian Metode YOLOv3 Pada Video 6

Skenario	Class	416 x 416							512x512						
		Confusion Matrix			Precision	Recall	F1 Score	$AP_{0.5}$	Confusion Matrix			Precision	Recall	F1 Score	$AP_{0.5}$
		TP	FP	FN					TP	FP	FN				
1	Pothole	0	0	0	-	-	-	-	0	0	0	-	-	-	-
	Alligator Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
	Lateral Crack	15	0	29	100.00%	34.09%	50.85%	59.16%	13	7	31	65.00%	29.55%	40.63%	47.13%
	Longitudinal Crack	5	1	19	83.33%	20.83%	33.33%	39.71%	3	0	21	100.00%	12.50%	22.22%	43.10%
2	Pothole	0	0	0	-	-	-	-	0	0	0	-	-	-	-
	Alligator Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
	Lateral Crack	19	0	25	100.00%	43.18%	60.32%	60.92%	19	4	25	82.61%	43.18%	56.72%	53.12%
	Longitudinal Crack	5	0	19	100.00%	20.83%	34.48%	51.51%	4	0	26	100.00%	13.33%	23.53%	51.03%
3	Pothole	0	0	0	-	-	-	-	0	0	0	-	-	-	-
	Alligator Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
	Lateral Crack	15	2	29	88.24%	34.09%	49.18%	58.29%	14	5	30	73.68%	31.82%	44.44%	48.09%
	Longitudinal Crack	2	1	22	66.67%	8.33%	14.81%	26.06%	1	1	23	50.00%	4.17%	7.69%	22.64%
4	Pothole	0	0	0	-	-	-	-	0	0	0	-	-	-	-
	Alligator Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
	Lateral Crack	18	2	26	90.00%	40.91%	56.25%	56.93%	15	6	29	71.43%	34.09%	46.15%	51.06%
	Longitudinal Crack	3	1	21	75.00%	12.50%	21.43%	31.44%	2	2	22	50.00%	8.33%	14.29%	32.06%
5	Pothole	0	0	0	-	-	-	-	0	0	0	-	-	-	-
	Alligator Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
	Lateral Crack	15	5	29	75.00%	34.09%	46.88%	48.87%	14	4	30	77.78%	31.82%	45.16%	55.36%
	Longitudinal Crack	7	4	17	63.64%	29.17%	40.00%	34.56%	9	3	15	75.00%	37.50%	50.00%	58.04%
6	Pothole	0	0	0	-	-	-	-	0	0	0	-	-	-	-
	Alligator Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
	Lateral Crack	18	5	26	78.26%	40.91%	53.73%	56.79%	11	4	33	73.33%	25.00%	37.29%	49.31%
	Longitudinal Crack	1	1	23	50.00%	4.17%	7.69%	20.51%	1	2	23	33.33%	4.17%	7.41%	30.48%
7	Pothole	0	0	0	-	-	-	-	0	0	0	-	-	-	-
	Alligator Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
	Lateral Crack	19	0	25	100.00%	43.18%	60.32%	59.24%	20	3	24	86.96%	45.45%	59.70%	57.97%
	Longitudinal Crack	6	0	18	100.00%	25.00%	40.00%	55.66%	6	0	18	100.00%	25.00%	40.00%	49.79%
8	Pothole	0	0	0	-	-	-	-	0	0	0	-	-	-	-
	Alligator Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
	Lateral Crack	16	4	28	80.00%	36.36%	50.00%	55.13%	15	15	29	50.00%	34.09%	40.54%	33.04%
	Longitudinal Crack	3	2	21	60.00%	12.50%	20.69%	28.99%	0	0	24	-	0.00%	-	7.15%
9	Pothole	0	0	0	-	-	-	-	0	0	0	-	-	-	-
	Alligator Crack	0	0	0	-	-	-	-	0	0	0	-	-	-	-
	Lateral Crack	16	0	28	100.00%	36.36%	53.33%	60.54%	13	5	31	72.22%	29.55%	41.94%	46.66%
	Longitudinal Crack	4	3	26	57.14%	13.33%	21.62%	35.30%	5	2	19	71.43%	20.83%	32.26%	47.88%

BIODATA PENULIS

Nama lengkap penulis Alvaro Basily Supriyanto, lahir di Tangerang pada tanggal 26 April 1998. Penulis sudah menempuh pendidikan formal di SD Al-Azhar 27 Cibinong (2004-2010), SMP Al-Azhar Syifa Budi Cibinong (2010-2013), SMA Pesantren Unggul Al Bayan Sukabumi (2013-2016), dan melanjutkan pendidikan S1 Matematika di Fakultas Sains dan Analitika Data Institut Teknologi Sepuluh Nopember. Selama perkuliahan penulis aktif mengikuti berbagai kegiatan. Diantaranya adalah menjadi *organizing committee* di ITS Mengajar *for Indonesia*, koordinator divisi *Offset Design* di Olimpiade Matematika ITS 2018 (OMITS 2018), menjadi *Head of Media and Information Departemen* di HIMATIKA ITS, peserta pelatihan *Digital Talent Scholarship: Online Academy 2019* yang diadakan oleh Kementerian Komunikasi dan Informatika Republik Indonesia, dan asisten dosen Sistem Basis Data pada tahun 2019. Demikian biodata tentang penulis. Jika terdapat kritik dan saran mengenai Tugas Akhir ini, penulis dapat dihubungi melalui *alvarobasily@gmail.com*.