TESIS – TE185401

DETEKSI KERUSAKAN JALAN MENGGUNAKAN CNN DENGAN ARSITEKTUR YOLO

ERNIN NISWATUL UKHWAH
NRP 07-111-75006-7007

DOSEN PEMBIMBING
Prof. Dr. Ir. Yoyon Kusnendar Suprapto, M.Sc
Dr. Eko Mulyanto Yuniarno, S.T, M.T

PROGRAM MAGISTER
BIDANG KEAHLIAN TELEMATIKA
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2019

TESIS – TE185401

DETEKSI KERUSAKAN JALAN MENGGUNAKAN CNN DENGAN ARSITEKTUR YOLO

ERNIN NISWATUL UKHWAH
NRP 07-111-75006-7007

DOSEN PEMBIMBING
Prof. Dr. Ir. Yoyon Kusnendar Suprapto, M.Sc
Dr. Eko Mulyanto Yuniarno, S.T, M.T

PROGRAM MAGISTER
BIDANG KEAHLIAN TELEMATIKA
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2019

LEMBAR PENGESAHAN

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Teknik (M.T)
di
Institut Teknologi Sepuluh Nopember

oleh:

Ernin Niswatul Ukhwah
NRP. 07-111-75006-7007

Tanggal Ujian : 26 Juni 2019
Periode Wisuda : September 2019

Disetujui oleh:

1. Prof. Dr. Ir. Yoyon Kusnendar Suprapto, M.Sc. (Pembimbing I)
   NIP: 195409251978031001

2. Dr. Eko Mulyanto Yuniarno, S.T, M.T. (Pembimbing II)
   NIP: 196806011995121009

3. Dr. Supeno Mardi Susiki Nugroho, ST.,M.T. (Penguji)
   NIP: 197003131995121001

4. Dr.Ir. Achmad Mauludiyanto, M.T. (Penguji)
   NIP: 196109031989031001

5. Dr. Ir. Wirawan, DEA (Penguji)
   NIP: 196311091989031011

Dekan Fakultas Teknologi Elektro

Dr. Tri Arief Sardjono, S.T., M.T.
NIP. 197002121995121001

Halaman ini sengaja dikosongkan

PERNYATAAN KEASLIAN TESIS

Dengan ini saya menyatakan bahwa isi keseluruhan Tesis saya dengan judul "**DETEKSI KERUSAKAN JALAN MENGGUNAKAN CNN DENGAN ARSITEKTUR YOLO**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, Juni 2019

Ernin Niswatul Ukhwah
NRP. 07-111-75006-7007

Halaman ini sengaja dikosongkan

DETEKSI KERUSAKAN JALAN MENGGUNAKAN CNN DENGAN ARSITEKTUR YOLO

Nama mahasiswa : Ernin Niswatul Ukhwah.
NRP : 07-111-75006-7007
Pembimbing : 1. Prof. Dr. Ir. Yoyon Kusnendar Suprapto, M.Sc.
2. Dr. Eko Mulyanto Yuniarno, S.T, M.T.

ABSTRAK

Penilaian kondisi jalan adalah kegiatan rutin yang dilakukan untuk mengumpulkan data pendukung untuk program penanganan jalan. Metode yang digunakan untuk pengumpulan data dapat menggunakan kendaraan survei khusus untuk mengambil gambar permukaan jalan dan data pendukung lainnya sehingga pengumpulan data dapat dilakukan lebih cepat. Pengambilan data dengan metode ini sudah sangat membantu namun masih memerlukan pengolahan data yang lebih rumit. Panjang ruas jalan nasional, perbedaan konstruksi jalan, kondisi tanah, dan beban kendaraan menyebabkan data yang diperoleh sangat banyak dan beragam. Metode pengolahan data secara otomatis atau semi-manual dinilai cukup mahal, tidak konsisten dan perkembangan regulasi juga menyebabkan data yang dihasilkan belum dapat memenuhi semua kriteria yang ada.

Berdasarkan kelemahan-kelemahan tersebut, pada penelitian ini dilakukan deteksi kerusakan jalan berdasarkan gambar sebagai alternatif dan pelengkap metode yang sudah ada saat ini. Pendekatan *Convolutional Neural Network* dengan deteksi satu tahap menggunakan arsitektur *Yolo* diterapkan untuk melakukan deteksi kerusakan jalan. Hasil penelitian pada satu kelas kerusakan jalan yaitu lubang menggunakan arsitektur *Yolo v3*, *Yolo v3 Tiny* dan *Yolo v3 SPP* memberikan hasil rata-rata akurasi *mAP* terbaik masing-masing sebesar 83,43%, 79,33%, dan 88,93%. Sedangkan perhitungan luas menunjukkan akurasi masing-masing sebesar 63,81%, 51,34%, dan 70,15%. Sedangkan penggunaan tiga kelas kerusakan yaitu lubang, retak dan lainnya menggunakan arsitektur *Yolo v3*, dan *Yolo v3 SPP* memberikan hasil rata-rata akurasi *mAP* terbaik masing-masing sebesar 77,45% dan 80,40%. Sedangkan perhitungan luas menunjukkan akurasi masing-masing sebesar 88,69% dan 91,13%. Dan waktu yang diperlukan untuk melakukan deteksi pada masing-masing gambar adalah 0,04 detik.

Kata kunci: Deteksi Kerusakan Jalan, Deteksi Lubang, YOLO, Visi Komputer, Deteksi Obyek.

Halaman ini sengaja dikosongkan

DISTRESS DETECTION USING CNN BASED ON YOLO ARCHITECTURE

By : Ernin Niswatul Ukhwah.
Student Identity Number : 07-111-75006-7007
Supervisor(s) : 1. Prof. Dr. Ir. Yoyon Kusnendar Suprapto, M.Sc.
2. Dr. Eko Mulyanto Yuniarno, S.T, M.T.

ABSTRACT

Road assessments are executed annually to all national road segments to collect supporting data on road maintenance and management programs. The method of data collection uses a survey vehicle to capture road images and other supporting data. Compared to data collection, data processing is more complicated. The national road section length, the difference of road construction, soil conditions, and vehicle load causes vast and various data. Automatic or semi-manual methods are quite expensive, inconsistent and not enough to meet the regulation requirements.

Based on those drawbacks, we proposed an object detection method based on the image as an alternative and complementary to current road distress detection techniques. The deep learning approach with single stage detection using the YOLO v3 architecture was applied to detect road distress. The results in pothole detection using *Yolo v3*, *Yolo v3 Tiny* dan *Yolo v3 SPP* architecture presents the best *mAP* average of 83.43%, 79.33% and 88.93% respectively. And the area measurement presents the accuracy of 63.81%, 51.34% and 70.15% respectively. While the result in distress detection with three distress classes, i.e potholes, cracks and others using *Yolo v3* and *Yolo v3 SPP* architecture presents the best *mAP* average of 77.45% and 80.40% respectively. While the area measurement presents the accuracy of 88.69% and 91.13% respectively. And it needs 0,04 second to detect each image.

Key words: distress detection, pothole detection, YOLO, computer vision, object detection.

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Segala puji syukur penulis panjatkan kepada Allah SWT atas rahmat, hidayah serta petunjuk-Nya sehingga penulis dapat menyelesaikan Tesis yang berjudul "**DETEKSI KERUSAKAN JALAN MENGGUNAKAN CNN DENGAN ARSITEKTUR YOLO**". Tesis ini disusun untuk memenuhi salah satu syarat kelulusan pada Program Magister Bidang Keahlian Telematika, Departemen Teknik Elektro, Fakultas Teknik Elektro, Institut Teknologi Sepuluh Nopember Surabaya.

Proses penyusunan dan penyelesaian tesis ini tidak terlepas dari bantuan berbagai pihak. Pada kesempatan ini penulis mengucapkan terima kasih dan penghargaan kepada :

1. Bapak Prof. Dr. Ir. Yoyon Kusnendar Suprapto, M. Sc dan Bapak Dr. Eko Mulyanto Yuniarno, S.T, M.T selaku dosen pembimbing, atas waktu dan bimbingannya selama penyusunan tesis.
2. Dewan penguji yang telah memberikan masukan dan saran dalam tesis ini.
3. Para Dosen Program Magister, Bidang Keahlian Telematika, Departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember Surabaya atas motivasi, pengetahuan, semangat, dan inspirasi selama penyelesaian studi.
4. Suami, Anak, Orang Tua, Adik, dan Kakak serta seluruh keluarga besar atas dukungan, semangat, doa dan pengorbanan yang diberikan.
5. Kepala Badan Litbang SDM Kementerian Komunikasi dan Informatika yang telah memberikan beasiswa untuk mengikuti pendidikan Program Magister Bidang Keahlian Telematika, Departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember Surabaya.
6. Keluarga Besar Balai Besar Pelaksanaan Jalan Nasional VIII Kementerian Pekerjaan Umum Dan Perumahan Rakyat atas dukungan selama penyelesaian studi.

7. Para Dosen dan seluruh staf sekretariat Program Magister Departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember Surabaya atas dukungan dan kerjasamanya selama penyelesaian studi.
8. Direktorat Jenderal Bina Marga, Dir. Pengembangan Jariangan Jalan, Subdit Analisa dan Pengembangan Sistem atas kemudahan dan bantuannya dalam mendapatkan data penelitian untuk penyusunan tesis ini.
9. Teman-teman Program Pengelola TIK Pemerintahan 2017.
10. Semua pihak yang telah membantu dalam proses pengerjaan tesis ini.

Penulis menyadari bahwa tesis ini masih jauh dari sempurna. Oleh karena itu masukan, saran dan kritik untuk perbaikan sangat diharapkan. Besar harapan penulis agar tesis ini dapat bermanfaat bagi pembaca dan penelitian-penelitian selanjutnya.

Surabaya, Juni 2019
Penulis

DAFTAR ISI

LEMBAR PENGESAHAN iii
PERNYATAAN KEASLIAN TESIS v
ABSTRAK vii
ABSTRACT ix
KATA PENGANTAR xi
DAFTAR ISI xiii
DAFTAR GAMBAR xvii
DAFTAR TABEL xix
DAFTAR NOMENKLATUR xxi
BAB 1 PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 4
1.3 Tujuan 5
1.4 Batasan Masalah 5
1.5 Kontribusi 5
BAB 2 KAJIAN PUSTAKA 7
2.1 Penelitian Terkait. 7
2.2 Kondisi Jalan 9
2.2.1 Jalan 9
2.2.2 Survei Kondisi Jalan 9
2.2.3 Kategori Kerusakan Jalan. 11
2.3 Teori Dasar. 14
2.3.1 *Neural Network.* 14
2.3.2 *Backpropagation.* 17
2.3.3 *Convolutional Neural Network.* 17
2.3.4 Fungsi Aktivasi 21
2.3.5 *Loss Function.* 22

2.3.6 Deteksi Obyek. 22
2.3.7 *You Only Look Once (YOLO)* 23
2.3.8 *Spatial Pyramid Pooling*. 29
2.3.9 Konfigurasi *Yolo v3*. 30
2.3.10 *Transfer Learning* dan *Fine Tuning*. 33
2.4 Evaluasi. 35
2.4.1 Evaluasi Deteksi. 35
2.4.2 Evaluasi Perhitungan Luas. 38
BAB 3 METODE PENELITIAN 39
3.1 Alur Penelitian. 39
3.2 Data Jalan Raya. 39
3.3 Anotasi dan pelabelan. 42
3.4 Pemodelan Kerusakan Jalan menggunakan *Yolo*. 43
3.5 Deteksi dan Perhitungan Luas. 45
3.6 Evaluasi. 46
3.6.1 Evaluasi Deteksi. 46
3.6.2 Evaluasi Perhitungan Luas. 46
3.7 Spesifikasi Perangkat. 47
BAB 4 HASIL DAN PEMBAHASAN 49
4.1 Percobaan Kelas Lubang. 49
4.1.1 Data Jalan Raya. 49
4.1.2 Pemodelan menggunakan *Yolo*. 50
4.1.3 Evaluasi. 53
4.2 Percobaan Kelas Lubang, Retak dan Lainnya. 63
4.2.1 Data Jalan Raya. 64
4.2.2 Statistik Data Penelitian. 65
4.2.3 Pemodelan menggunakan *Yolo*. 65
4.2.4 Evaluasi. 72
4.3 Hasil deteksi secara visual. 86
4.3.1 Deteksi Sukses. 87
4.3.2 Hasil deteksi gagal. 88
4.3.3 Deteksi pada skema pemodelan *Yolo v3* dan *Yolo v3 SPP*. 88

4.4 Proses identifikasi kerusakan jalan secara manual dan yang dilakukan pada penelitian ini. 90
4.5 Evaluasi waktu komputasi. 92
4.5.1 Evaluasi waktu komputasi berbasis *cloud.* 93
4.5.2 Evaluasi waktu komputasi berbasis *offline* menggunakan *CPU.* 94
BAB 5 PENUTUP 97
5.1 Kesimpulan. 97
5.2 Saran. 98
DAFTAR PUSTAKA 101
LAMPIRAN 105

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Gambar 2.1 Kendaraan Survei *Hawkeye.* 10
Gambar 2.2 Pengukuran Luas Kerusakan Permukaan Jalan menggunakan *software hawkeye processing toolkit.* 10
Gambar 2.3 *Form* Penilaian Kerusakan Permukaan Jalan. 11
Gambar 2.4 Contoh lubang pada perkerasan lentur. 13
Gambar 2.5 Contoh retak pada perkerasan lentur. a. retak kulit buaya. b. retak blok. c. retak refleksi sambungan. d. retak tepi. e. retak memanjang/melintang. f. retak selip. 14
Gambar 2.6 Struktur *Neuron* Pada Otak Manusia. 15
Gambar 2.7 *Artificial Neuron* Sederhana (Cesar dan da Fontoura Costa, 2002). . 15
Gambar 2.8 Contoh sederhana *Artificial Neural Network.* 16
Gambar 2.9 3-*layer neural network.* 18
Gambar 2.10 *Convolutional Neural Network* (Parthy, 2018). 19
Gambar 2.11 Lapisan Jaringan *CNN* (Goodfellow, Bengio dan Courville, 2016). 19
Gambar 2.12 Ilustrasi Operasi konvolusi (Goodfellow, Bengio dan Courville, 2016). 20
Gambar 2.13 Ilustrasi *Pooling layer* (Parthy, 2018). 21
Gambar 2.14 Sistem Deteksi *Yolo* (Redmon *et al.*, 2015). 24
Gambar 2.15 Gambaran Umum *Yolo* (Redmon *et al.*, 2015). 25
Gambar 2.16 Arsitektur *Yolo v1.* 26
Gambar 2.17 *Spatial Pyramid Pooling* pada *Yolo v3* (Huang dan Wang, 2019). . 30
Gambar 2.18 Konfigurasi arsitektur Yolo v3. 31
Gambar 2.19 Konfigurasi arsitektur Yolo v3 SPP. 32
Gambar 2.20 Konfigurasi arsitektur Yolo v3 Tiny. 33
Gambar 2.21 Contoh *transfer learning* di mana dua lapisan *fully connected* yang terakhir dihapus, dan dua lapisan adaptasi baru ditambahkan ke jaringan. Kedua lapisan ini kemudian disesuaikan untuk melakukan tugas baru (Oquab *et al.*, 2014). 34
Gambar 2.22 Ilustrasi *IoU*. a. ilustrasi antara deteksi dan *ground truth*. b. ilustrasi irisan deteksi dan *ground truth*. c. ilustrasi nilai *IoU*. 36
Gambar 3.1 Alur Penelitian. 39
Gambar 3.2 Data citra jalan raya dengan kerusakan. a. Lubang. b. Retak. c. Trotoar. d. Bahu Jalan. 40
Gambar 3.3 Contoh anotasi. a. Anotasi kelas lubang. b. Anotasi kelas retak. c. Anotasi kelas lainnya dan kelas lubang. d. Anotasi kelas lainnya dan kelas retak. 42
Gambar 3.4 Contoh output anotasi dan pelabelan. 43
Gambar 4.1 *Loss Yolo v3, Yolo v3 Tiny*, dan *Yolo v3 SPP*. 52
Gambar 4.2 *mAP Yolo v3*, *Yolo v3 Tiny* dan *Yolo v3 SPP.* 58

Gambar 4.3 Perbandingan akurasi deteksi (*mAP)* dan akurasi luas percobaan kelas lubang. 63
Gambar 4.4 *Loss fine tuning* dan *transfer learning.* 68
Gambar 4.5 *Loss* data latih 516 dan 774. 69
Gambar 4.6 *Loss Yolo v3* dan *Yolo v3 SPP.* 71
Gambar 4.7 Perbandingan *mAP Fine Tuning, Transfer Learning* dengan *Yolo v3* dan *Transfer Learning* dengan *Yolo v3 SPP.* 79
Gambar 4.8 Perbandingan *mAP* berdasarkan data latih yang digunakan pada percobaan kelas lubang, retak dan lainnya. 80
Gambar 4.9 Perbandingan *AP* masing-masing tipe kerusakan jalan. 81
Gambar 4.10 Sebaran akurasi luas percobaan kelas lubang, retak dan lainnya. 85
Gambar 4.11 Perbandingan *mAP* dan Akurasi Luas Percobaan kelas lubang, retak dan lainnya. 86
Gambar 4.12 Hasil deteksi sukses. 87
Gambar 4.13 Hasil deteksi gagal. a. deteksi gagal pada kelas retak. b. deteksi gagal pada kelas lubang. 88
Gambar 4.14 Deteksi gagal pada *Yolo v3* dan sukses pada *Yolo v3 SPP.* 89
Gambar 4.15 Deteksi gagal pada *Yolo v3 SPP* dan sukses pada *Yolo v3.* 90
Gambar 4.16 Identifikasi kerusakan jalan metode manual dengan metode yang diusulkan. 91

DAFTAR TABEL

Tabel 2.1 Perbaikan *Yolo v1* ke *Yolo v2*. 27
Tabel 2.2 Arsitektur *Yolo v2*. 27
Tabel 3.1 Daftar ruas jalan nasional yang digunakan. 41
Tabel 3.2 Akurasi Perhitungan Luas. 47
Tabel 4.1 Distribusi data dan kerusakan kelas lubang. 50
Tabel 4.2 Matriks konfigurasi percobaan kelas lubang. 50
Tabel 4.3 Matriks *pre-trained* percobaan kelas lubang. 51
Tabel 4.4 Rekapitulasi *Loss* Pemodelan Percobaan kelas lubang. 53
Tabel 4.5 *Precision, Recall,* skor *F1, IoU* dan *mAP* percobaan kelas lubang menggunakan arsitektur *Yolo v3*. 55
Tabel 4.6 *Precision, Recall,* skor *F1, IoU* dan *mAP* percobaan kelas lubang menggunakan arsitektur *Yolo v3 Tiny*. 56
Tabel 4.7 *Precision, Recall,* skor *F1, IoU* dan *mAP* percobaan kelas lubang menggunakan arsitektur *Yolo v3 SPP*. 57
Tabel 4.8 Rata-rata *mAP* Percobaan kelas lubang. 59
Tabel 4.9 Akurasi perhitungan luas percobaan kelas lubang menggunakan arsitektur *Yolo v3*. 60
Tabel 4.10 Akurasi perhitungan luas percobaan kelas lubang menggunakan arsitektur *Yolo v3 Tiny*. 61
Tabel 4.11 Akurasi perhitungan luas percobaan kelas lubang menggunakan arsitektur *Yolo v3 SPP*. 62
Tabel 4.12 Kombinasi skema pemodelan dan pengujian. 64
Tabel 4.13 Distribusi Kerusakan Jalan. 65
Tabel 4.14 Matriks konfigurasi percobaan kelas lubang, retak dan lainnya. 66
Tabel 4.15 Matriks *pre-trained* percobaan kelas lubang, retak dan lainnya. 66
Tabel 4.16 Rekapitulasi *Loss* pemodelan skema *fine tuning* dan *transfer learning* dengan 516 dan 774 data latih 70
Tabel 4.17 Rekapitulasi *Loss* Pemodelan dengan *Yolo v3* dan *Yolo v3 SPP*. 72
Tabel 4.18 *Precision, Recall,* dan skor *F1* percobaan kelas lubang, retak dan lainnya menggunakan skema *fine tuning* dan arsitektur *Yolo v3*. 74
Tabel 4.19 *IoU*, Ap dan *mAP* percobaan kelas lubang, retak dan lainnya menggunakan skema *fine tuning* dan arsitektur *Yolo v3*. 74
Tabel 4.20 *Precision, Recall,* dan skor *F1* percobaan kelas lubang, retak dan lainnya menggunakan skema *transfer learning* dan arsitektur *Yolo v3*. 76
Tabel 4.21 *IoU*, Ap dan *mAP* percobaan kelas lubang, retak dan lainnya menggunakan skema *transfer learning* dan arsitektur *Yolo v3*. 76
Tabel 4.22 *Precision, Recall,* dan skor *F1* Percobaan dengan target tiga kelas dan arsitektur *Yolo v3 SPP*. 78
Tabel 4.23 *IoU*, *Ap* dan *mAP* percobaan dengan target tiga kelas dan arsitektur *Yolo v3 SPP*. 78

Tabel 4.24 Akurasi perhitungan luas percobaan kelas lubang, retak dan lainnya skema *Fine Tuning* dan Arsitektur *Yolo v3*. 82
Tabel 4.25 Akurasi perhitungan luas percobaan kelas lubang, retak dan lainnya skema *Transfer Learning* dan Arsitektur *Yolo v3*. 83
Tabel 4.26 Akurasi perhitungan luas percobaan kelas lubang, retak dan lainnya skema *Transfer Learning* dan Arsitektur *Yolo v3 SPP*. 84
Tabel 4.27 Waktu komputasi deteksi kelas lubang pada komputasi berbasis *cloud*. 93
Tabel 4.28 Waktu komputasi deteksi kelas lubang, retak, dan lainnya pada komputasi berbasis *cloud*. 94
Tabel 4.29 Waktu komputasi deteksi kelas lubang pada komputasi berbasis *offline*. 95
Tabel 4.30 Waktu komputasi deteksi kelas lubang, retak, dan lainnya pada komputasi berbasis *offline*. 95

DAFTAR NOMENKLATUR

x_i	: vektor input
y_i	: vektor output
w_{ij}	: matriks bobot
b_j	: bias ke lapisan berikutnya
W	: ukuran input
F	: ukuran *filter*
P	: *padding*
S	: *stride*
TP	: *True Positive*
FP	: *False Positive*
FN	: *False Negative*
Acc	: akurasi luas
L	: Luas *ground truth*
L'	: Luas hasil deteksi
w	: lebar *bounding box*
h	: tinggi *bounding box*
x	: pusat *bounding box* pada sumbu x
h	: pusat *bounding box* pada sumbu y
S	: selisih luas *ground truth* dengan luas prediksi dalam satuan m
S'	: selisih luas ground truth dengan luas prediksi dalam persen

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Jalan merupakan infrastruktur yang penting dalam mendukung perekonomian nasional sebagai prasarana transportasi darat untuk distribusi barang dan jasa (Republik Indonesia, 2004). Keberadaan jalan memberikan kemudahan mobilitas yang akan berdampak pada pertumbuhan ekonomi. Akhir-akhir ini program pemerintah Indonesia diprioritaskan pada pembangunan infrastrukur termasuk pembangunan jalan. Pembangunan tidak hanya berfokus pada penyediaan jalan baru, namun juga pada pemeliharaan jalan. Oleh karena itu, keberadaan data kondisi jalan sangat penting untuk mendukung program penanganan dan pemeliharaan jalan yang efektif dan efisien.

Data kondisi jalan dapat disediakan melalui program penilaian kondisi jalan. Penilaian kondisi jalan minimal dilakukan setiap tahun sekali, sesuai dengan peraturan yang ada. Hasil penilaian kondisi jalan merupakan data awal dalam menentukan program penanganan jalan. Begitu juga pemodelan kerusakan jalan dalam mendukung Sistem Manajemen Perkerasan juga disediakan dari data kondisi jalan. Penialain kondisi jalan terdiri dari tiga tahap, yaitu tahap pengumpulan data, identifikasi dan klasifikasi dan yang terakhir adalah penilaian kondisi jalan (Koch, Jog dan Brilakis, 2013).

Pengumpulan data yang sudah berkembang di Indonesia dan negara berkembang pada umumnya terbagi menjadi dua kelompok (Hoang, 2018). Kelompok pertama yaitu survei manual dengan berjalan kaki atau menggunakan kendaraan dengan kecepatan rendah. Hasil pengamatan dicatat pada *form* survei sesuai dengan ketentuan petunjuk manual pelaksanaan survei kondisi jalan. Kelompok yang kedua yaitu menggunakan kendaraan survei yang dilengkapi dengan kamera untuk mengambil foto permukaan jalan. Cara yang kedua sudah sangat membantu dalam pengumpulan data. Namun untuk identifikasi dan klasifikasi kerusakan jalan masih ada yang harus dilakukan secara manual oleh operator menggunakan *software image processing.* Kompleksitas pengolahan data

berupa identifikasi dan klasifikasi kerusakan jalan tergantung pada perangkat akuisisi yang digunakan.

Dua metode tersebut memerlukan biaya yang tinggi baik dari sisi waktu maupun tenaga (Koch dan Brilakis, 2011; Hidayatullah *et al.*, 2012) dan juga inkonsistensi dalam melakukan identifikasi jenis kerusakan jalan, karena dipengaruhi oleh pengalaman dan jam kerja operator (Coenen dan Golroo, 2017). Dengan banyaknya ruas jalan nasional yang harus diinspeksi seperti di Propinsi Jawa Timur saja terdapat jalan nasional ± 2.400 km (Balai Besar Pelaksanaan Jalan Nasional VIII, 2018). Peraturan penilaian kondisi jalan juga membagi jenis kerusakan jalan lebih rinci, sebagaimana dalam penentuan Indeks Kondisi Perkerasan (IKP) yaitu sebanyak dua puluh kategori kerusakan jalan (Kementerian Pekerjaan Umum dan Perumahan Rakyat, 2016b). Sehingga peralatan yang ada belum tentu dapat mememenuhi kebutuhan tersebut sehingga dari tahun ke tahun menjadi semakin tertinggal. Hal ini mendorong untuk dilakukan perbaikan secara terus menerus untuk melengkapi solusi yang sudah ada agar data penilaian kondisi jalan dapat disediakan secara lebih lengkap.

Dari beberapa kategori kerusakan yang ada, lubang adalah salah satu indikator penting kerusakan jalan (Turner-Fairbank Highway Research Center, 2000), oleh karena itu deteksi dan perhitungan yang tepat memberikan pengaruh dalam penilaian kondisi jalan dan penanganan yang akan dilakukan. Begitu juga dengan retak, dimana dalam IKP retak dirinci menjadi enam kategori yang berbeda (Kementerian Pekerjaan Umum dan Perumahan Rakyat, 2016b). Sehingga retak memberikan prosentasi besar dalam penilaian kondisi jalan. Dua jenis kerusakan ini merupakan sebagian target *output* dari deteksi kerusakan jalan yang diangkat dalam penelitian ini. Disamping dua kategori tersebut juga dimasukkan kategori lainnya dimana yang termasuk kategori lainnya adalah bahu jalan dan trotoar. Hal ini dimotivasi oleh kondisi data yang diperoleh dimana banyak obyek berupa bahu jalan dan trotoar yang juga tertangkap kamera. Tentunya hal ini juga banyak ditemui, karena masih banyak jalan di Indonesia yang mempunyai lebar cukup kecil.

Beberapa penelitian sudah dilakukan dalam bidang deteksi kerusakan jalan, diantaranya berbasis laser (Chang, Chang dan Liu, 2005), sensor getaran

(Mednis *et al.*, 2011; Wang *et al.*, 2015) dan citra baik dua dimensi (Vigneshwar dan Kumar, 2016) maupun tiga dimensi menggunakan kamera stereo (Moazzam *et al.*, 2013; Zhang *et al.*, 2014). Deteksi kerusakan jalan berbasis citra memiliki beberapa keunggulan dibanding metode lainnya, yaitu: teknologi dalam pengolahan citra sudah cukup matang, interpretasi gambar mudah dilakukan, dan tersedianya alat akuisisi data yang terjangkau berupa kamera, merupakan nilai lebih untuk melakukan deteksi kerusakan jalan berbasis citra.

Metode yang berkembang dalam pengolahan citra untuk deteksi kerusakan jalan dapat digolongkan menjadi dua, yaitu dua tahap dan satu tahap. Dalam metode dua tahap, deteksi dilakukan dengan diawali proses ekstraksi fitur, kemudian didefiniskan perbedaan antara jalan yang rusak dengan jalan yang tidak rusak sebagai dasar penentuan klasifikasi kerusakan jalan. Klasifikasi dapat dilakukan dengan memberikan *threshold* tertentu untuk memisahkan antara jalan yang rusak dan tidak (Koch dan Brilakis, 2011). Kelemahan dengan penerapan *threshold* tertentu yaitu berupa ketidak fleksibelan dalam melakukan klasifikasi memunculkan penggunaan metode-metode yang lebih *adaptive* seperti penggunaan *fuzzy* (Ouma dan Hahn, 2017) dan *machine learning* menggunakan *Least Squares Support Machine (LS-SVM)* dan *Artificial Neural Network* (Hoang, 2018).

Pendekatan-pendekatan diatas telah membantu dalam proses deteksi kerusakan jalan, akan tetapi memiliki kelemahan di mana ekstraksi fitur harus dilakukan oleh tenaga ahli untuk menghasilkan kinerja akurasi yang baik dan juga waktu komputasi yang lama selama proses deteksi (LeCun, Bengio dan Hinton, 2015). Kelemahan ini telah di jawab dengan munculnya *deep learning* dalam bidang visi komputer. *Deep Learning* mampu melakukan ekstraksi fitur dan klasifikasi secara berurutan melalui operasi *Convolutional Neural Network (CNN)* sekaligus. Deteksi kerusakan jalan berbasis *deep learning* juga sudah mulai diterapkan (Maeda *et al.*, 2018) dimana deteksi kerusakan jalan menggunakan *single stage detection* dilakukan terhadap data yang diambil menggunakan *smartphone* dengan diletakkan pada bagian depan mobil dengan fokus kamera menghadap kedepan. Sehingga informasi luas kerusakan jalan tidak didapatkan dalam penelitian ini. Meskipun hasil penelitian ini memuaskan namun tidak dapat

diterapkan dalam penilaian kondisi jalan yang ada di Indonesia. Karena diperlukan luasan dalam melakukan identifikasi dan klasifikasi kerusakan jalan.

Oleh karena itu dalam penelitian ini, proses deteksi kerusakan jalan dilakukan terhadap data dengan proses pengambilan gambar menggunakan kamera belakang dan fokus kebawah yang diperoleh dari kendaraan survei khusus. Dengan pengaturan pengambilan gambar seperti ini, maka dapat diperoleh kategori kerusakan jalan dan luas kerusakan jalan. Sehingga dapat memenuhi kebutuhan penilaian kondisi jalan yang ada di Indonesia.

Penelitian ini menggunakan metode *CNN* dalam melakukan deteksi kerusakan jalan dengan arsitektur *YOLO*. *YOLO* merupakan metode *state of the art* dalam bidang deteksi obyek, dimana memiliki nilai akurasi yang memuaskan dan waktu komputasi yang cukup cepat (Redmon dan Farhadi, 2018). Dengan kelebihan ini diharapkan deteksi yang dilakukan dapat diterapkan dalam penilaian kondisi jalan di Indonesia, mengingat jumlah peralatan survei yang ada bersifat terbatas sedangkan panjang jalan nasional yang harus dilakukan penilaian cukup banyak.

1.2 Rumusan Masalah

Penilaian kondisi jalan merupakan bagian penting dalam menentukan rencana penanganan jalan. Metode yang digunakan terutama identifikasi dan klasifikasi kerusakan jalan serta perhitungan luas kerusakan jalan belum semuanya dapat diselesaikan dengan mudah. Perkembangan regulasi, panjang ruas jalan dan masih adanya proses yang dilakukan secara manual menjadikan penilaian kondisi jalan menjadi mahal dan rawan bias. Oleh karena itu, perlu dilakukan upaya perbaikan untuk melakukan deteksi kerusakan jalan, dimana metode yang dikembangkan dapat digunakan sebagai pelengkap metode yang sudah ada dan dapat menyediakan data kondisi jalan sebagai dasar penyusunan rencana penanganan jalan.

1.3 Tujuan

Tujuan penelitian ini adalah melakukan deteksi kerusakan jalan serta mendapatkan luas kerusakan jalan sehingga dapat membantu dalam melakukan identifikasi dan klasifikasi kerusakan jalan pada proses penilaian kondisi jalan.

1.4 Batasan Masalah

Batasan masalah pada penelitian ini adalah:

1. Penelitian ini dilakukan untuk deteksi kerusakan jalan dengan tiga kelas yaitu lubang, retak dan lainnya berdasarkan pengamatan 2 Dimensi.
2. Data yang digunakan dalam penelitian ini adalah data hasil survei permukaan jalan yang diperoleh dari kendaraan survei *Hawkeye* 2000.
3. Deteksi kerusakan jalan hanya pada perkerasan lentur (*flexible pavement*).

1.5 Kontribusi

Kontribusi yang diharapkan dari hasil penelitian ini adalah :

1. Dapat menjadi alat bantu identifikasi dan klasifikasi kerusakan jalan untuk menghindari subyektifitas yang dilakukan oleh *operator* maupun *surveyor*.
2. Identifikasi kerusakan jalan dapat disajikan lebih cepat dan lengkap sehingga informasi yang disampaikan sesuai dengan kondisi lapangan dan pengambilan keputusan terhadap program penanganan jalan dapat dilakukan secara tepat.
3. Hasil identifikasi dapat digunakan sebagai data dasar penilaian kondisi jalan sehingga penentuan program penanganan jalan jangka panjang dapat dilakukan dengan tepat.

Halaman ini sengaja dikosongkan

BAB 2
KAJIAN PUSTAKA

2.1 Penelitian Terkait.

Koch et.al. melakukan deteksi kerusakan jalan dengat target kelas berupa lubang (Koch dan Brilakis, 2011). Langkah yang dilakukan yaitu dengan melakukan segmentasi terlebih dahulu untuk memisahkan bagian jalan yang rusak dengan yang tidak, menggunakan histogram ambang batas berbasis bentuk. Berdasarkan pada sifat-sifat geometris dari daerah lubang, bentuk lubang diperkirakan menggunakan *morphological thinning* dan regresi elips. Selanjutnya, tekstur di dibagian lubang diekstrak dan dibandingkan dengan tekstur disekitar lubang untuk menentukan bagian tersebut adalah lubang. Hasil penelitian ini memberikan akurasi yang cukup memuaskan. Namun ada keterbatasan dalam waktu komputasi karena dilakukan terhadap citra tunggal dan kelemahan dalam menentukan tingkat keparahan lubang. Untuk melengkapi dan meningkatkan metode tersebut, Koch et al. menggunakan video sebagai obyek penelitian (Koch, Jog dan Brilakis, 2013). Dengan melakukan perbaikan dalam menentukan tekstur yang *representative* dalam satu area perkerasan dan memanfaatkan pelacakan dan penghitungan lubang untuk mengurangi waktu komputasi, meningkatkan kemampuan deteksi dan menghitung lubang secara efisien.

Yashon O. Ouma et.al melakukan deteksi lubang pada permukaan jalan menggunakan pendekatan dua dimensi berbiaya rendah (Ouma dan Hahn, 2017). Pendekatan yang dilakukan dibagi menjadi tiga tahap, tahap pertama yaitu melakukan *filtering* gambar 2 Dimensi menggunakan transfomasi *wavelet* multiskala, tahap kedua yaitu klasifikasi antara citra lubang dan bukan lubang menggunakan *Fuzzy C-Means Clustering*, dan langkah terakhir dilakukan pengisisan daerah yang dianggap lubang menggunakan operasi morfologi dilasi. Data yang digunakan pada penelitian ini diambil menggunakan *smartphone* yang ditempatkan pada kaca depan dengan pengambilan data sejauh 3 km dan periode pengambilan gambar sebanyak 5 *frame* per menit. Data yang digunakan untuk percobaan sebanyak 75 gambar.

L. Huidrom et al. juga melakukan deteksi kerusakan jalan dengan tiga kelas kerusakan, yaitu lubang, retak dan tambalan. Langkah yang dilakukan yaitu dengan memisahkan *frame* yang mengandung kerusakan dan tidak, menggunakan algoritma DFS (Huidrom, Das dan Sud, 2013). Langkah-langkah utama dalam algoritma ini adalah segmentasi piksel kerusakan dari *background* menggunakan teknik *thresholding* adaptif dan logika pemisahan yang ditentukan pengguna untuk mengklasifikan kerusakan jalan. Kemudian dilanjutkan dengan algoritma *CDDMC* menggunakan fitur tekstur, bentuk dan dimensi untuk melakukan klasifikasi kerusakan jalan.

Ketiga metode diatas merupakan metode-metode deteksi kerusakan jalan menggunakan pendekatan *image processing* yang dilanjutkan dengan klasifikasi kerusakan jalan menggunakan *threshold* tertentu. Metode-metode tersebut dinilai kurang adaptif sehingga perlu dilakukan pendekatan berbasis *machine learning* yang lebih adaptif dalam melakukan deteksi keruskan jalan (Huidrom, Das dan Sud, 2013).

Nhat-Duc Hoang mengembangkan model *machine learning* yang lebih adaptif untuk melakukan deteksi lubang. Pengembangan model kecerdasan buatan menggunakan metode ekstraksi fitur *Gaussian filter, steerable filter*, dan *integral projection* diterapkan (Hoang, 2018). Kemudian dilakukan klasifikasi kerusakan jalan menggunakan *Least Squares Support Machine* (*LS-SVM*) dan *Artificial Neural Network*. Hasil yang diperoleh menunjukkan metode *LS-SVM* memberikan akurasi yang lebih baik dibandingkan dengan metode *Artificial Neural Network*.

H. Maeda et al. melakukan deteksi kerusakan jalan menggunakan *Convolutional Neural Network (CNN)* dengan pendekatan *Single Stage Detection* (Maeda *et al.*, 2018), Zhang et al. juga melakukan deteksi retak untuk membandingkan tiga metode yaitu *Deep Convolutional Neural Network, Boosting*, dan *SVM* (Zhang *et al.*, 2016). Masing-masing metode tersebut dilatih dan diuji menggunakan 500 gambar dengan ukuran 3264×2448 piksel yang diambil menggunakan *smartphone*. Hasil penelitian menunjukkan bahwa metode *Deep Convolutional Neural Network* memberikan akurasi paling baik dibanding dua metode lainnya.

2.2 Kondisi Jalan.

2.2.1 Jalan

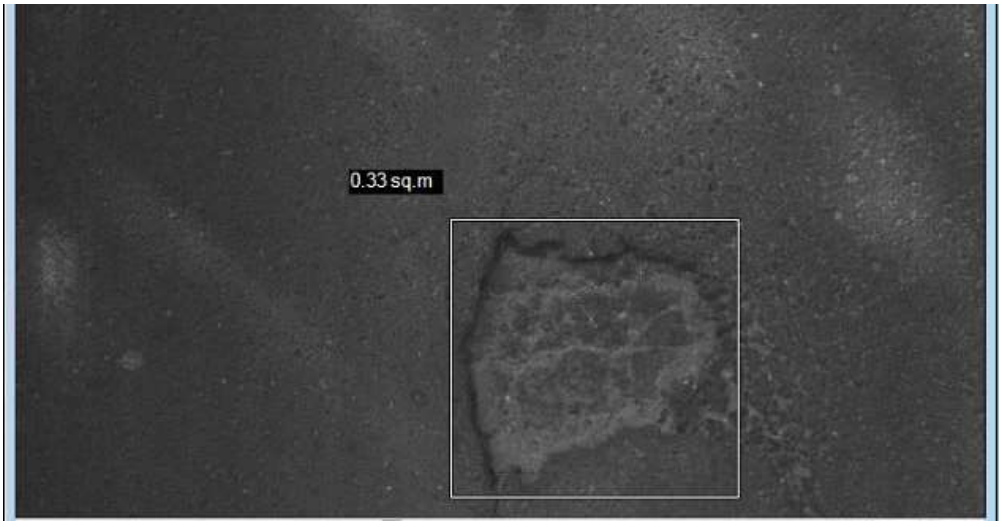
Menurut Undang - Undang Nomor 38 tahun 2004 tentang Jalan, Jalan adalah prasarana transportasi darat yang meliputi segala bagian jalan, termasuk bangunan pelengkap dan perlengkapannya yang diperuntukkan bagi lalu lintas, yang berada pada permukaan tanah, di atas permukaan tanah, di bawah permukaan tanah dan/atau air, serta di atas permukaan air, kecuali jalan kereta api, jalan lori, dan jalan kabel. Jalan sebagai bagian prasaranan transportasi mempunyai peran penting dalam bidang ekonomi, sosial budaya, lingkungan hidup, politik, pertahanan dan keamanan serta dipergunakan untuk sebesar-besar kemakmuran rakyat. Asas penyelenggaraan jalan salah satunya yaitu keselamatan, keselamatan ini berkenaan dengan kondisi permukaan jalan (Republik Indonesia, 2004). Oleh karena itu pemerintah sebagai penyelenggara jalan yang memegang asas keselamatan berusaha untuk menyediakan prasarana jalan yang baik dengan melakukan penilaian secara berkala terhadap kondisi jalan.

2.2.2 Survei Kondisi Jalan

Survei Kondisi Jalan (SKJ) bertujuan untuk menentukan kondisi jalan pada satu waktu tertentu. SKJ adalah bagian dari analisis fungsional jalan secara langsung dengan mendata kondisi bagian jalan yang mudah berubah baik untuk jalan beraspal dan jalan kerikil/tanah. Salah satu output SKJ adalah data kerusakan permukaan jalan. Adapun metode SKJ yang dapat digunakan meliputi metode manual dengan cara berjalan dengan jarak tertentu misalnya perseratus meter dengan menilai setiap jenis kondisi dengan estimasi ukuran, kedalaman dan persentasi luas. Hasil pengukuran diisikan pada *form* penilaian kondisi jalan. Seiring berkembangnya teknologi, metode SKJ dapat dilakukan dengan metode otomatis. Pengumpulan data dilakukan menggunakan kendaraan survei seperti *Hawkeye* sebagaimana ditunjukkan pada Gambar 2.1. Alat ini dilengkapi komponen *Digital Laser Profiler* untuk mendapatkan data *IRI* dan *rutting* dan komponen *Pavement View Camera* untuk mendapatkan data permukanaan jalan. Untuk melengkapi peralatan *Hawkeye* dalam mengolah data hasil survei lebih lanjut, digunakan perangkat piranti lunak (*software*) yang merupakan bagian dari paket peralatan yaitu *Hawkeye Processing Toolkit*.

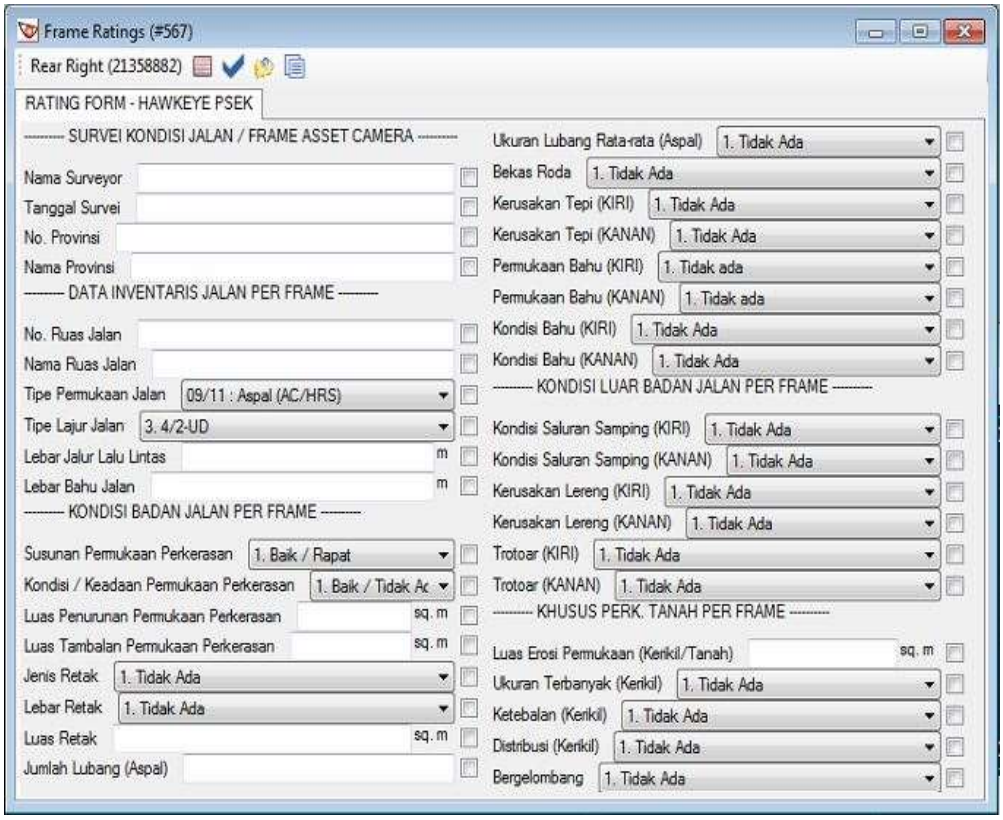
Gambar 2.1 Kendaraan Survei *Hawkeye.*

Identifikasi dan klasifikasi kerusakan jalan dilakukan dengan cara menganalisa data menggunakan *software Hawkeye Processing Toolkit* dari data foto yang dikumpulkan menggunakan komponen *Pavement View Camera. Pavement View Camera* adalah kamera yang terpasang pada sisi belakang kendaraan sebanyak dua buah, Ilustrasi pemasangan kamera ditunjukkan pada kotak warna merah pada Gambar 2.1. Hasil video dari *pavement view camera* dirubah menjadi *frame* citra foto. Citra foto ini dilakukan pengukuran luas kerusakan dengan melakukan penandaan dimasing-masing kondisi kerusakan sebagaimana ditunjukkan pada Gambar 2.2 (Kementerian Pekerjaan Umum dan Perumahan Rakyat, 2016a).



Gambar 2.2 Pengukuran Luas Kerusakan Permukaan Jalan menggunakan *software hawkeye processing toolkit.*

Pengukuran luas diinputkan kedalam *form* yang ditunjukkan pada Gambar 2.3 (Kementerian Pekerjaan Umum dan Perumahan Rakyat, 2016a) yang nantinya dilakukan rekapitulasi untuk menentukan indeks kondisi perkerasan jalan. Proses ini rawan bias (Coenen dan Golroo, 2017), mahal, perlu banyak tenaga dan waktu yang lama (Koch dan Brilakis, 2011; Hidayatullah *et al.*, 2012) untuk memperoleh hasil yang diharapkan. Karena operator harus melihat satu persatu kerusakan pada masing-masing *frame* untuk identifikasi dan klasifikasi kerusakan jalan.



Gambar 2.3 *Form* Penilaian Kerusakan Permukaan Jalan.

2.2.3 Kategori Kerusakan Jalan.

Berdasarkan Surat Edaran Menteri Pekerjaan Umum dan Perumahan Rakyat No. 19/SE/M/2016 tentang pedoman penentuan Indeks Kondisi Perkerasan (IKP) (Kementerian Pekerjaan Umum dan Perumahan Rakyat, 2016b), kategori kerusakan jalan pada perkerasan beton aspal dikelompokkan menjadi:

1. Retak kulit buaya (retak lelah)
2. Kegemukan
3. Retak blok

4. Jembul dan lekukan *(bumps and sags)*
5. Keriting (*corrugation*)
6. Ambles/depresi (*depression*)
7. Retak tepi (*edge cracking*)
8. Retak refleksi sambungan (*joint reflection cracking*)
9. Penurunan lajur/bahu (*lane/shoulder drop off*)
10. Retak memanjang dan melintang (bukan retak refleksi)
11. Tambalan dan tambalan galian utilitas
12. Pengausan agregat (*polished aggregate*)
13. Lubang
14. Persilangan rel kereta api (*railroad crossing*)
15. Alur (*rutting*)
16. Sungkur (*shoving*)
17. Retak selip (*slippage cracking*)
18. Pemuaian (*swell*)
19. Pelepasan butir (*ravelling*)
20. Pelapukan (*surface wear*).

Dari total dua puluh jenis kerusakan jalan tersebut, satu kategori kerusakan jalan yaitu alur *(rutting)* sudah dapat diidentifikasi secara otomatis oleh *Hawkeye 2000.* Untuk kategori lainnya masih perlu dilakukan identifikasi dan klasifikasi serta perhitungan luas kerusakan jalan menggunakan *form rating* secara manual. Oleh karena itu, pada penelitian ini dipilih obyek kerusakan jalan berupa lubang, retak, dan lainnya sebagai fokus penelitian untuk dilakukan deteksi obyek kerusakan secara otomatis sehingga diperoleh kategori kerusakan secara otomatis dilengkapi dengan luas kerusakannya. Meskipun pada prakteknya, tidak semua kategori kerusakan perlu diperoleh luas kerusakan, bisa jadi hanya berupa panjang kerusakan, akan tetapi dalam penelitian ini dilakukan *generalisasi* dalam bentuk ukuran luas.

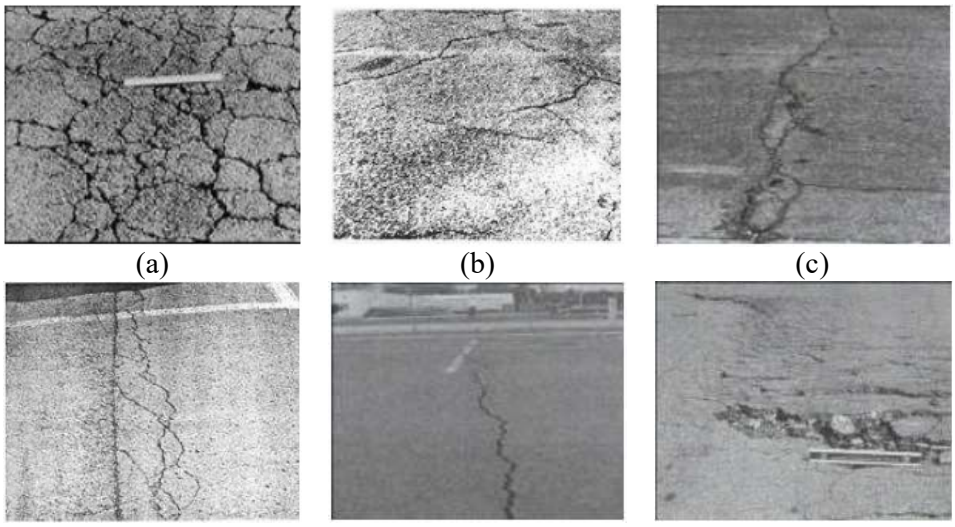
Pemilihan lubang sebagai salah satu obyek dalam penelitian ini dikarenakan, lubang merupakan salah satu indikator penting kerusakan jalan (Turner-Fairbank Highway Research Center, 2000). Oleh karena itu deteksi lubang penting dilakukan untuk mendapatkan data kondisi jalan agar dapat dilakukan

penanganan jalan yang tepat. Sedangkan retak merupakan jenis kategori kerusakan jalan yang dalam IKP dikelompokkan menjadi enam kategori, dan dalam penelitian ini dikelompokkan menjadi satu kategori retak. Hal ini menunjukkan bahwa retak memberikan prosentase yang besar dalam penilaian kondisi jalan. Adapun kategori lainnya berupa bahu jalan dan trotoar dipilih sebagai obyek penelitian karena kategori tersebut banyak ditemui pada data penelitian yang ada. Dan kondisi ini tentunya juga banyak ditemui diwiliyah Indonesia, karena masih banyak jalan yang memiliki lebar cukup kecil. Sehingga selain permukaan jalan, obyek disekitarnya berupa bahu jalan dan trotoar tertangkap kamera.

Masing-masing tipe kerusakan mempunyai definisi yang berbeda-beda sesuai dengan IKP, lubang didefiniskan sebagai cekungan pada permukaan perkerasan yang mempunyai diameter kecil, biasanya kurang dari 750 mm (30 in). Lubang umumnya mempunyai sudut yang tajam dan dinding bagian atas yang tegak. Contoh lubang pada perkerasan lentur ditunjukkan pada Gambar 2.4.

Gambar 2.4 Contoh lubang pada perkerasan lentur.

Sedangkan definisi retak beragam sesuai dengan kategori retak yang terdiri dari enam kategori yang lebih rinci yaitu retak kulit buaya, retak blok, retak tepi retak refleksi sambungan, retak memanjang dan melintang, dan retak selip. Retak dapat berupa retak tungal seperti pada retak memanjang dan melintang, ataupun kumpulan retak yang tersambung membentuk blok seperti retak blok, dan retak buaya yang memiliki kerapatan blok retak lebih kecil. Contoh masing-masing dari keenam kategori retak tersebut ditunjukkan pada Gambar 2.5.



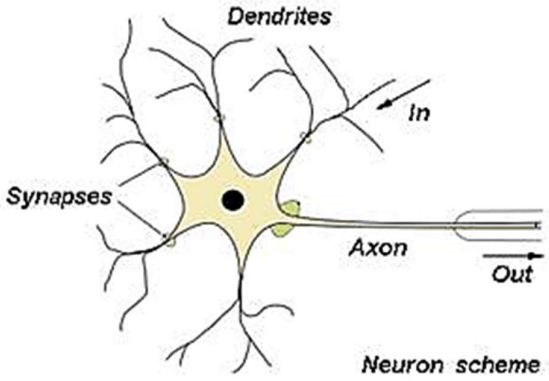
Gambar 2.5 Contoh retak pada perkerasan lentur. a. retak kulit buaya. b. retak blok. c. retak refleksi sambungan. d. retak tepi. e. retak memanjang/melintang. f. retak selip.

2.3 Teori Dasar.

2.3.1 *Neural Network.*

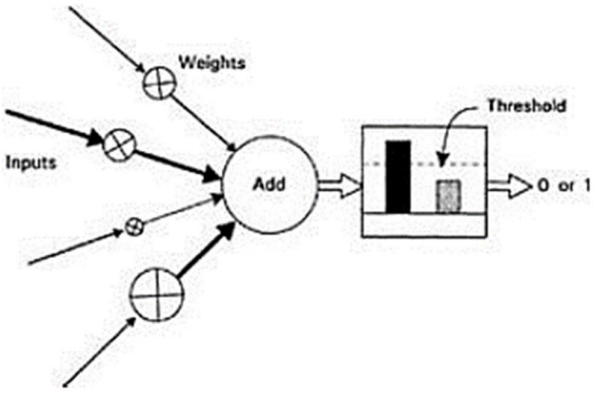
Neural network (*NN*) adalah jaringan dari sekumpulan elemen, unit atau node pemroses yang saling berhubungan, yang fungsinya didasarkan pada sistem saraf (*neuron)* manusia. Kemampuan pemrosesan jaringan disimpan dalam kekuatan koneksi interunit, atau bobot *(weight)*, yang diperoleh dengan proses belajar dari serangkaian pola pelatihan (Cesar dan da Fontoura Costa, 2002). Ide dasar *Neural Network* dimulai dari otak manusia, dimana proses yang terjadi pada otak manusia adalah sebuah neuron menerima impuls dari *neuron* lain melalui dendrit dan mengirimkan sinyal yang dihasilkan oleh badan sel melalui akson. Akson dari sel syaraf ini bercabang-cabang dan berhubungan dengan dendrit dari sel syaraf lain dengan cara mengirimkan impuls melalui sinapsis. Sinapsis adalah unit fungsional antara 2 buah sel syaraf, misal A dan B, dimana yang satu adalah serabut akson dari neuron A dan satunya lagi adalah dendrit dari neuron B. Kekuatan sinapsis bisa menurun/meningkat tergantung seberapa besar tingkat propagasi (penyiaran) sinyal yang diterimanya. Impuls-impuls sinyal (informasi) akan diterima oleh neuron lain jika memenuhi batasan tertentu, yang sering disebut

dengan nilai ambang (*threshold*). Struktur neuron pada otak manusia di tunjukkan pada Gambar 2.6.



Gambar 2.6 Struktur *Neuron* Pada Otak Manusia.

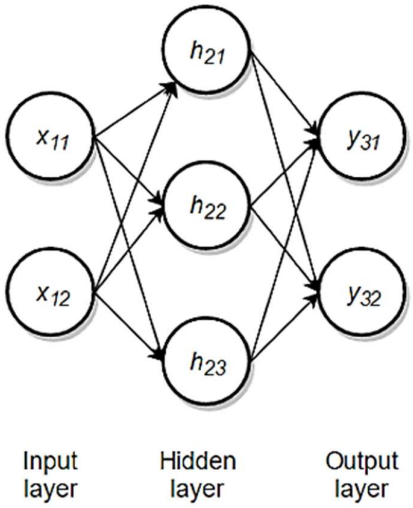
Dari struktur *neuron* pada otak manusia, dan proses kerja yang dijelaskan di atas, maka konsep dasar pembangunan neural network buatan (*Artificial Neural Network*) terbentuk. Sinapsis dimodelkan dengan angka tunggal *(weight)* sehingga setiap input dikalikan dengan bobot menggunakan fungsi aritmatik penjumlahan sederhana kemudian dikirim ke node aktifasi. Dari node aktifasi ini dilakukan perbandingan dengan *threshold* tertentu, jika melebihi *threshold* maka input tersebut menghasilkan nilai yang tinggi (mendekati 1) jika dibawah *threshold* maka input tersebut menghasilkan nilai yang rendah (mendekati 0), sebagaimana diilustrasikan pada Gambar 2.7.



Gambar 2.7 *Artificial Neuron* Sederhana (Cesar dan da Fontoura Costa, 2002).

Gambar 2.7 adalah jaringan tunggal yang akan mengirimkan nilai output ke semua *neuron* yang berhubungan dengannya. Proses ini akan terus berulang pada

input-input selanjutnya sehingga membentuk suatu jaringan *neuron* yang lebih kompleks sebagaimana ditunjukkan pada Gambar 2.8.



Gambar 2.8 Contoh sederhana *Artificial Neural Network.*

Neural network pada umumnya memiliki beberapa *layer*: *layer* masukan (*input layer)*, *layer tersembunyi (hidden layer),* dan *layer* keluaran *(output layer)*. Setiap neuron adalah unit komputasi yang mengambil nilai *input* dari lapisan sebelumnya dan mengeluarkan nilai ke lapisan berikutnya kecuali neuron pada *layer* masukan. Jika dari ilustrasi pada Gambar 2.6 dan Gambar 2.7 diberikan skenario sebuah proses pemodelan dimana *input* dinotasikan sebagai x_i dan y_i adalah target *output* yang dikehendaki, maka pemodelan adalah proses mempelajari pola data dari *input x* untuk menghasilkan *output* label *y*.

Proses pemodelan dapat diwakili sebagai fungsi nonlinier $fw(x)$, di mana x adalah fitur *input* dan w adalah matriks bobot. Pada Gambar 2.8 dapat kita lihat bahwa jaringan memiliki dua neuron *input*, dilambangkan x_{11} dan x_{12}, dan dua neuron *output* y_{31} dan y_{32}. *Output* dari jaringan dapat dilambangkan sebagai $(y_{31}, y_{32}) = fw\,(x_{11}, x_{12})$. Jika dalam gambar tersebut *hidden layer* h_j merupakan *layer* lanjutan setelah *input*, maka neuron h_j adalah kombinasi liniar dari *input* x_i dikalikan dengan bobot w_{ij} dan bias b_j. Sehingga fungsi *output* secara umum dapat diperoleh dengan persamaan 2.1.

$$O_j = \sum_{i=1}^{n} w_{ij} x_i + b_j \quad (2.1)$$

Dimana:

b_j adalah bias ke $hidden\ layer\ h$.

w_{ij} adalah bobot dari *layer* input x_i ke $hidden\ layer\ h_j$.

Langkah terakhir untuk memperoleh nilai h_j adalah memasukkan O_j ke fungsi aktivasi nonlinier f. Sehingga h_j dapat diperoleh dari persamaan 2.2.

$$h_j = f(O_j) = f\left(\sum_{i=1}^{n} w_{ij} x_i + b_j\right) \quad (2.2)$$

Karena setiap neuron hanya tergantung pada perhitungan lapisan sebelumnya, maka perhitungan aktivasi dilakukan lapisan demi lapisan hingga lapisan *output*. Sehingga diperoleh hasil akhir dari neuron keluaran $y = fw\,(x)$.

2.3.2 *Backpropagation.*

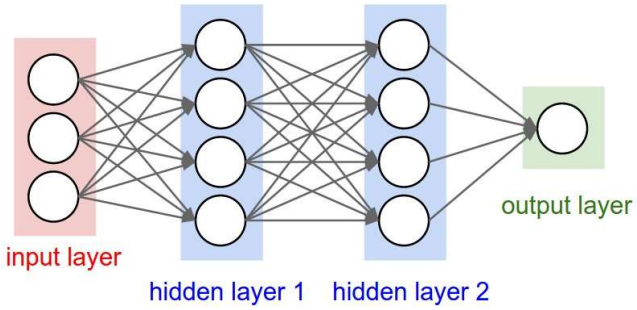
Backpropagation adalah proses perambatan balik yang dilakukan secara berulang-ulang untuk memperkecil *loss* pada proses pemodelan. Pada proses pemodelan *input* dilewatkan ke depan melalui *layer-layer* jaringan saraf dan ditransformasikan melalui fungsi aktivasi. Begitu input mencapai *layer* terakhir, hasil prediksi dibandingkan dengan *ground truth*. Dengan membandingkan prediksi dengan *ground truth*, maka *error* dapat dihitung dengan fungsi *loss*. *Loss* yang dihitung kemudian digunakan untuk memperbarui bobot beberapa *layer* jaringan untuk meminimalkan total *loss*. Proses memperbarui bobot ini dilakukan dengan mempropagasikan *loss* dari *layer* terakhir sampai dengan *layer input*.

2.3.3 *Convolutional Neural Network.*

Convolutional Neural Network (CNN) pertama kali diperkenalkan pada 1989 untuk melakukan pengenalan tulisan tangan dan menunjukkan hasil yang memuaskan (LeCun *et al.*, 1989). Keberhasilan *CNN* ini meningkatkan minat para peneliti untuk mengembangkannya, dan terbukti *CNN* mampu melakukan tugas-tugas pengenalan pola pada gambar secara baik dan melebihi kesuksesan metode *machine learning* yang lain. *CNN* mampu melakukan ekstraksi fitur dan mengkombinasikan fitur lokal untuk menghasilkan fitur yang lebih abstrak pada level yang lebih tinggi. Adapun definisi *CNN* secara sederhana adalah *Neural*

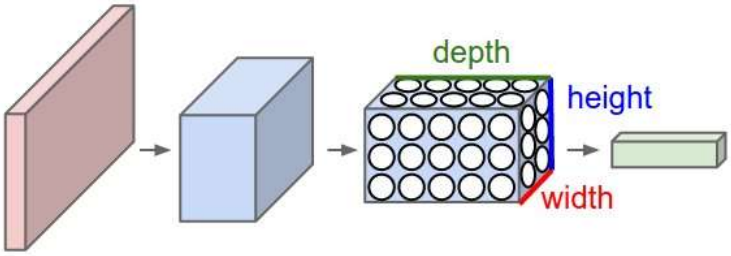
Network yang menggunakan operasi konvolusi menggantikan operasi perkalian matriks minimal pada satu dari *layer*-nya.

Pada Gambar 2.9 merupakan arsitektur *neural network* biasa dimana terdapat satu *input layer* dengan masukan berupa tiga *input neuron*, dua *hidden layer* dan satu *output layer*. Arsitektur tersebut menerima *input* data satu dimensi dan mempropagasikan data tersebut pada jaringan hingga menghasilkan *output*. Setiap hubungan antar *neuron* pada dua *layer* yang bersebelahan memiliki parameter bobot satu dimensi, disetiap data *input* pada *layer* dilakukan operasi linier dengan nilai bobot yang ada, kemudian hasil komputasi akan ditransformasi menggunakan operasi non liniar yang disebut sebagai fungsi aktivasi.



Gambar 2.9 3-*layer neural network.*

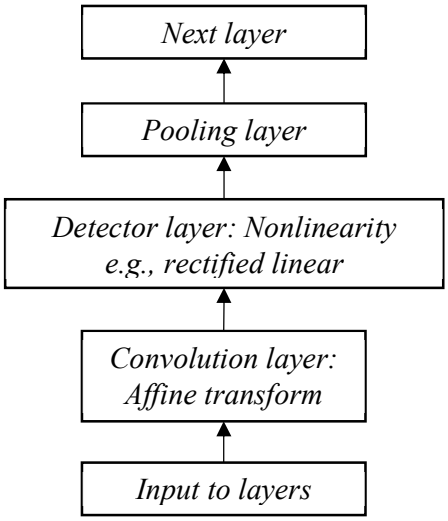
Pada *CNN*, data yang dipropagasikan pada jaringan adalah data dua dimensi, sehingga operasi liniar dan parameter bobot pada *CNN* berbeda. Pada *CNN* operasi liniar menggunakan operasi konvolusi, sedangkan bobot tidak lagi satu dimensi saja, namun berbentuk tiga dimensi yang merupakan kumpulan kernel konvolusi seperti pada Gambar 2.10. *CNN* mengatur neuronnya dalam tiga dimensi (lebar, tinggi, kedalaman), seperti yang divisualisasikan di salah satu lapisan. Setiap lapisan *CNN* mengubah volume *input* 3D menjadi volume *output* 3D dari aktivasi neuron. Dalam contoh ini, lapisan input merah menampung gambar, sehingga lebar dan tingginya akan menjadi dimensi gambar, dan kedalamannya adalah 3 *channel* (Merah, Hijau, Biru). Dari input ini akan dilakukan operasi konvolusi sehingga pada akhir *layer* akan menghasilkan *output* $x \times x \times d$ dimana d adalah kedalaman dimensi *output layer*.



Gambar 2.10 *Convolutional Neural Network* (Parthy, 2018).

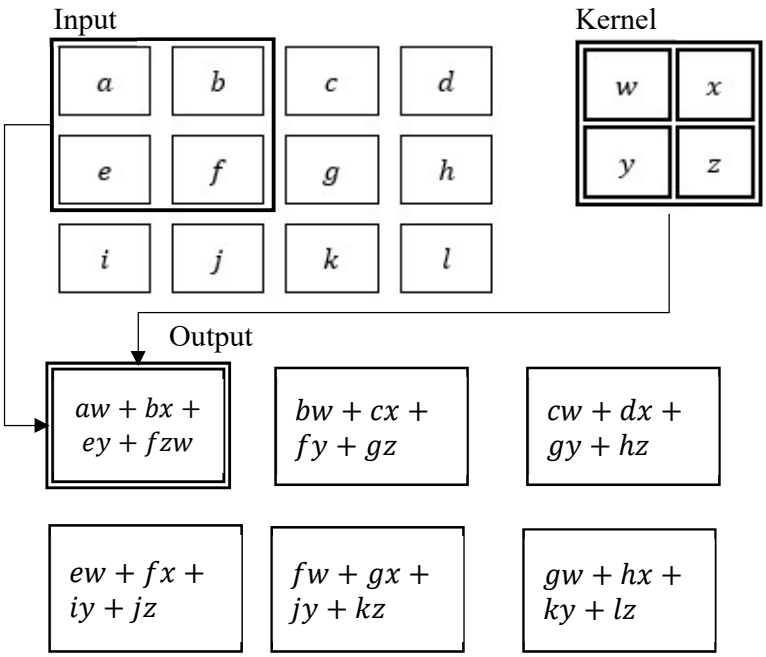
2.3.3.1 Arsitektur Convolutional Neural Network.

Lapisan jaringan konvolusi secara umum terdiri dari tiga tahap, yaitu yaitu *convolution layer, activation* dan *pooling layer* sebagaimana ditunjukkan pada Gambar 2.11. Pada *convolution layer*, lapisan melakukan operasi konvolusi secara paralel untuk menghasilkan satu set aktivasi linier. Pada tahap *activation* setiap aktivasi linier dijalankan melalui fungsi aktivasi nonlinier, seperti *rectified linear*, atau juga disebut *detector*, pada *pooling layer* fungsi *pooling* digunakan untuk memodifikasi output lapisan.



Gambar 2.11 Lapisan Jaringan *CNN* (Goodfellow, Bengio dan Courville, 2016).

Convolutional Layer terdiri dari sekumpulan *filter* berupa matriks bobot yang dikonvolusikan terhadap input pada *layer* sebelumnya. Operasi konvolusi dilakukan dengan melakukan perkalian dan penjumlahan antara *filter* dengan input dari *layer* sebelumnya untuk menghasilkan *output* berupa *feature map* sebagaimana diilustrasikan pada Gambar 2.12.



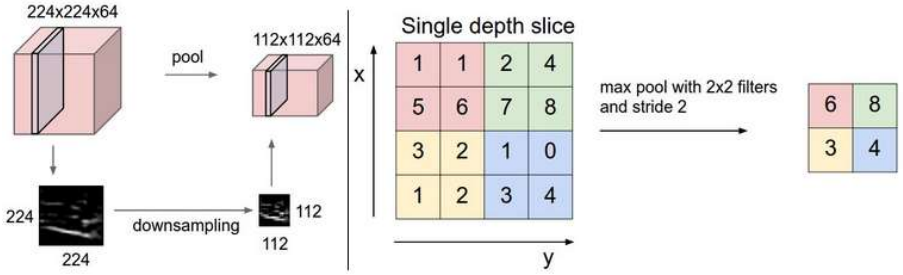
Gambar 2.12 Ilustrasi Operasi konvolusi (Goodfellow, Bengio dan Courville, 2016).

Operasi konvolusi yang diterapkan pada data *input* secara langsung akan menghasilkan fitur yang cukup kompleks karena semua fitur yang ada pada gambar dipetakan. Sehingga *CNN* dapat melakukan tugas-tugas pengenalan pola dengan baik. Hal ini merupakan kelebihan *CNN* dimana proses *pre-processing* seperti ekstraksi fitur yang biasanya harus dilakukan oleh tenaga ahli agar memperoleh hasil yang baik (LeCun, Bengio dan Hinton, 2015) dapat dilakukan oleh *CNN* dan memberikan hasil yang memuaskan. Namun dengan *CNN* komputasi menjadi lebih mahal karena proses ekstraksi fitur dilakukan terhadap data mentah dan biasanya harus diterapkan dalam jumlah banyak untuk memperoleh hasil pengenalan pola yang memuaskan.

Untuk menghitung jumlah neuron yang merupakan *output* dari operasi konvolusi maka perlu memperhatikan tiga parameter penting yaitu *depth* (kedalaman *output*) atau jumlah *filter*, *stride* (pergeseran operasi konvolusi), dan *padding* (penambahan nilai 0 pada tepi input). Jika W adalah ukuran *input*, F adalah ukuran *filter*, P adalah *padding,* dan S adalah *stride* maka ukuran *output* dapat diperoleh dengan persamaan 2.3.

$$Ukuran\ Output = \frac{(W - F + 2P)}{S} + 1 \quad (2.3)$$

Pooling layer berada setelah ***conv layer*** dan berfungsi untuk mereduksi ukuran jaringan serta menghindari *overfitting*. *Pooling layer* beroperasi terhadap keseluruhan input dan mengubah ukuran menggunakan operasi seperti *max pooling.* Paremeter yang perlu diperhatikan dalam *pooling layer* adalah *filter* yang biasanya berukuran 2×2 dan *stride.* Contoh proses *pooling* ditunjukkan pada Gambar 2.13. Volume *input*-an berukuran $224 \times 224 \times 64$ dilakukan operasi *pooling* dengan *filter* berukuran 2×2, *stride* 2 sehingga menghasilkan *output* berukuran $112 \times 112 \times 64$. Ukuran *input* direduksi sedangkan dimensi kedalaman masih tetap, yaitu 64. Pada Gambar 2.13 juga ditunjukkan operasi *downsampling* yang paling umum yaitu *max pooling*, operasi *max pooling* dilakukan dengan cara memilih nilai *max* dari setiap *filter* berukuran 2×2 dengan *stride* 2 sehingga mereduksi *input layer* dari ukuran 4×4 menjadi 2×2.



Gambar 2.13 Ilustrasi *Pooling layer* (Parthy, 2018).

2.3.4 **Fungsi Aktivasi**.

Fungsi aktivasi adalah bagian penting dari jaringan saraf karena mendefinisikan *output* dari sebuah *node* yang diberikan satu set *input*. Pada *CNN* fungsi aktivasi yang biasa digunakan *adalah Rectified Linear Units (ReLU). Rectified Linear Units (ReLU)* diusulkan sebagai fungsi aktivasi alternatif yang telah terbukti mengkonversi lebih cepat daripada nonlinier jenuh (Nair dan Hinton, 2010) dan dijelaskan dalam persamaan 2.4 sebagai fungsi maksimum 0 dan nilai yang diberikan.

$$g(z) = \max\{0, z\} \quad (2.4)$$

Varian *ReLU* yang juga sering digunakan adalah *Leaky ReLU* sebagaimana yang digunakan pada *Yolo v3*. *Leaky ReLU* muncul karena pada fungsi aktivasi

ReLU akan mengkonversi semua input yang bernilai kurang dari 0 menjadi 0 hal ini menyebabkan neuron menjadi tidak aktif. Hal ini menjadikan *ReLU* belum tentu sesuai dengan semua data set. Sehingga *Leaky ReLU* dapat diperoleh dengan persamaan 2.5.

$$g(z) = \max\{az, z\} \tag{2.5}$$

Dimana $a \leq 1$ adalah koefisien *leaky*. Dengan koefisien ini maka memungkinkan gradien non-nol kecil ketika neuron tidak aktif. Kerjanya hampir identik dengan *ReLU* normal, tetapi ditampilkan untuk *convergen* lebih cepat.

2.3.5 *Loss Function*.

Loss adalah tingkat *error* antara prediksi dengan *groundtruth* pada proses pemodelan/pelatihan. Semakin kecil *loss* yang dihasilkan maka model yang dihasilkan semakin peka dalam melakukan prediksi, klasifikasi maupun tugas pengenalan pola lainnya. Untuk mengurangi kesalahan prediksi/klasifikasi dilakukan dengan cara memodifikasi parameter menggunakan fungsi *loss*. Fungsi ini memberikan ukuran seberapa tidak akuratnya *output* prediksi dibandingkan dengan kelas data pemodelan yang diinginkan. Biasanya fungsi *loss* dinormalisasi ke skor antara 0 dan 1 untuk mewakili seberapa besar kemungkinan jaringan akan melakukan kesalahan klasifikasi sampel pelatihan.

Fungsi *loss* yang biasanya digunakan pada masalah regresi adalah *Sum Square Error (SSE)* sebagaimana dapat diperoleh dari persamaan 2.6 dan untuk masalah klasifikasi adalah fungsi *cross entropy loss* (Nasr, Badr dan Joun, 2002) sebagaimana dapat diperoleh dari persamaan 2.7.

$$E(W) = \sum_{i=1}^{N} ||t_i - y_i||^2 \tag{2.6}$$

$$E(W) = -\sum_{i=1}^{N} [t_i log y_i + (1 - t_i) log(1 - y)] \tag{2.7}$$

Dimana Nadalah jumlah keseluruhan data pemodelan, y_i adalah hasil prediksi sedangkan t_i adalah *groundtruth (*kelas yang dikehendaki).

2.3.6 Deteksi Obyek.

Deteksi obyek adalah proses menentukan klasifikasi obyek dan menentukan lokasi obyek tersebut pada citra (Zhao *et al.*, 2019). Pada proses

klasifikasi, model yang dikembangkan bertujuan untuk mengkategorikan sebuah gambar kedalam kelas tertentu. Meskipun dalam gambar tersebut memiliki lebih dari satu obyek maka klasifikasi hanya mampu mengkategorikan kedalam satu kelas. Atau obyek yang ada dianggap sebagai obyek tunggal. Salah satu contoh klasifikasi adalah mengkategorikan gambar hewan kedalam kelas kucing atau anjing. Contoh lain penggunaan kasus klasifikasi, yang relevan dengan tesis ini adalah menentukan apakah kerusakan jalan terdapat pada citra atau tidak.

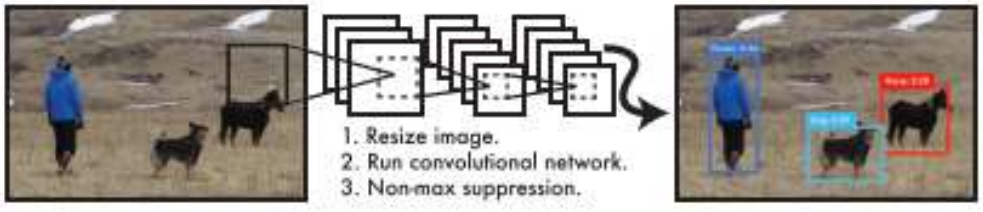
Meskipun klasifikasi mampu mengelompokkan gambar dengan baik, namun informasi mengenai lokasi obyek tidak dapat dilakukan oleh proses klasifikasi. Sedangkan pada prakteknya posisi obyek (*object localization)* diperlukan untuk melakukan analisa lebih lanjut. Sebagaimana dalam penelitian ini, lokasi obyek lebih ditujukan untuk memperoleh luasan obyek yang terdapat pada citra. Disisi lain, Deteksi objek juga memungkinkan deteksi beberapa objek dalam suatu gambar dan juga memberikan informasi lokasi obyek tersebut.

2.3.7 *You Only Look Once (YOLO)*

You Only Look Once (*YOLO*) adalah salah satu metode deteksi obyek dengan pendekatan *single stage*. Dalam melakukan deteksi, *Yolo* melakukan operasi konvolusi *CNN* untuk menentukan lokasi dan kelas obyek sekaligus dari keseluruhan *image*. Secara sederhana langkah-langkah *Yolo* dalam melakukan deteksi adalah (Redmon *et al.*, 2015):

1. Gambar yang diinputkan akan di-*rezise* menjadi ukuran tertentu sesuai arsitektur yang sudah ditentukan. Misalnya menggunakan *input* berukuran $416 \times 416 \times 3$.
2. Kemudian dilakukan operasi *CNN* untuk menentukan fitur dan kemudian melakukan klasifikasi dan lokalisasi.
3. Hasil deteksi terdiri dari banyak *bounding box*, oleh karena itu perlu dilakukan operasi *Non-Max Suppression* untuk menentukan *bounding box* dengan *IoU* paling tinggi sebagai hasil deteksi yang dipilih.

Ilustrasi langkah-langkah deteksi menggunakan *Yolo* ditunjukkan pada Gambar 2.14.



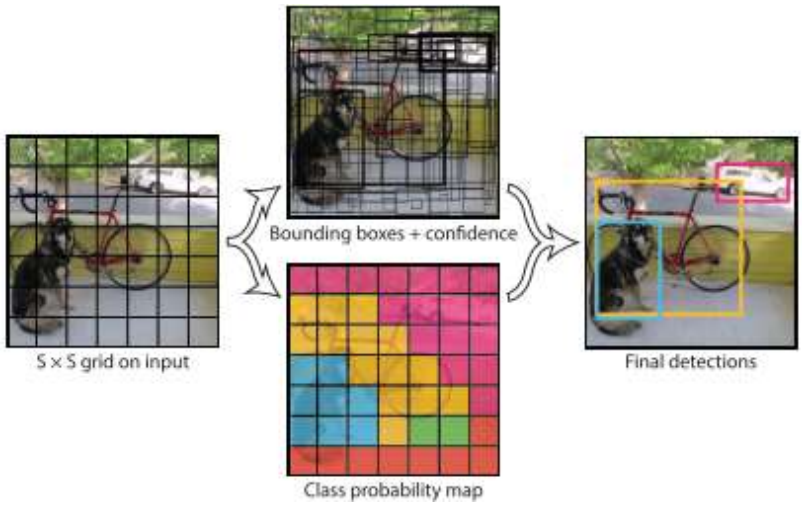
Gambar 2.14 Sistem Deteksi *Yolo* (Redmon *et al.*, 2015).

2.3.7.1 Yolo v1.

Cara kerja *Yolo* secara rinci dimulai dari data citra yang diinputkan akan dibagi menjadi $S \times S$ *grid*. Masing-masing *grid cell* akan melakukan prediksi:

- Satu obyek,
- *B bounding box*, tiap-tiap *bounding box* terdiri dari lima elemen, yaitu *x, y, w, h* dan *box confidence score*. *Box confidence score* adalah seberapa *confidence* model menganggap *bounding box* tersebut terdapat obyek dan seberapa akurat *bounding box* tersebut
- *C Conditional Class probabilities*, yaitu peluang *grid cell* tersebut terdapat obyek dan termasuk kelas ke-i. Setiap *grid cell* hanya menetapkan satu *conditional class probability* meskipun *cell* tersebut melakukan deteksi sebanyak B *bounding box*.

Dari *bounding box* dan *confidence score* sebanyak *B* dan *class probability* sebanyak *C* maka pada akhir prediksi akan terbentuk $S \times S \times (B \times 5 + C)$ *output* hasil deteksi dan $S \times S \times B$ *bounding box*. Dari semua *bounding box* tersebut perlu dilakukan penentuan *bounding box* yang benar-benar terdapat obyek menggunakan operasi *non-max suppression*, sehingga pada akhir prediksi hanya diperoleh beberapa *bounding box* yang mempunyai probabilitas paling tinggi dan berjumlah satu untuk masing-masing obyek. Gambaran umum cara kerja *Yolo* ditunjukkan pada Gambar 2.15.



Gambar 2.15 Gambaran Umum *Yolo* (Redmon *et al.*, 2015).

2.3.7.1.1 *Non-Max Suppression*.

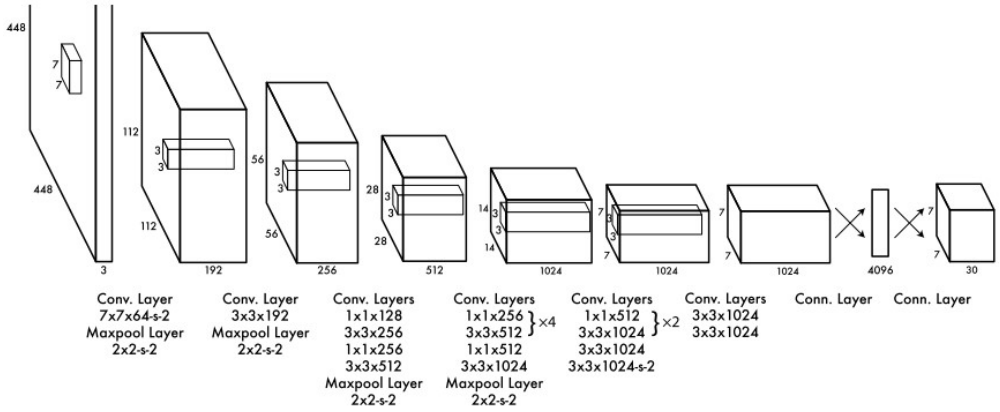
Output deteksi setelah dilakukan operasi konvolusi terdiri dari banyak *bounding box*. Jumlah ini bergantung pada jumlah *detector* yang digunakan dan jumlah *grid cell* yang sudah ditentukan. Oleh karena itu perlu dilakukan pemilihan *bounding box* hasil deteksi yang paling tepat. Pemilihan ini dapat dilakukan dengan operasi *non-max suppression* untuk mengeliminasi *bounding box* yang tidak diharapkan. Adapun langkah-langkah operasi *non-max suppression* adalah:

1. Urutkan hasil deteksi berdasarkan skor probabilitas *class*.
2. Eliminiasi *bounding box* pada *grid cell* yang memiliki probabilitas kecil, misalnya *bounding box* yang memiliki *class conditional probability* $< 0{,}5$. Sehingga hanya menyisakan *bounding box* yang memiliki *class conditional probability* $> 0{,}5$.
3. Mulai dari skor teratas, pilih satu *bounding box* dan eliminasi *bounding box* lain yang memiliki misalnya IoU>0,5 dengan *bounding box* yang dipilih dan kelas yang sama. Periksa *bounding* box yang tersisa pada *grid cell* tersebut jika memiliki kelas yang sama maka eliminasi.
4. Ulangi langkah 3 sampai semua *bounding box* diperiksa.

2.3.7.1.2 Arsitektur *Yolo v1*.

Arsitektur jaringan *Yolo v1* terinspirasi oleh *GoogleNet* yang terdiri dari 24 lapisan konvolusi dan diikuti oleh dua lapisan *fully-connected*. Dua puluh lapisan pertama diperoleh dari *pre-trained* pada *ImageNet* dan kemudian dikonversi

untuk melakukan deteksi dengan menambahkan empat lapisan konvolusi dan dua lapisan *fully connected* dengan bobot yang diinisialisasi secara acak sebagaimana ditunjukkan secara rinci pada Gambar 2.16. Pada lapisan terakhir, kelas dan koordinat *bounding box* dinormalisasi untuk memprediksi lebar dan tinggi antara 0 dan 1. Lapisan terakhir menggunakan fungsi aktivasi linier sedangkan semua lapisan lainnya menggunakan *leaky ReLU* dengan koefisien *leakage* α= 0,1. Sedangkan fungsi *loss* yang digunakan pada *Yolo v1* adalah *Sum Square Error (SSE)* karena *Yolo* merupakan pendekatan penyelesaian permasalahan regresi.



Gambar 2.16 Arsitektur *Yolo v1*.

Untuk mengurangi *overfitting*, *dropout* dan *augmentasi* digunakan. Setelah lapisan *fully connected* yang pertama, lapisan *dropout* dimasukkan secara acak untuk "menjatuhkan" *input node* dengan probabilitas p = 0,5. Untuk data augmentasi digunakan *random scalling* dan *translation* sampai dengan 20% dari ukuran *image* asli. *Exposure* dan saturasi juga dilakukan dengan faktor 1,5 pada ruang warna *HSV*.

2.3.7.2 Yolo v2

Yolo versi 2 (Redmon dan Farhadi, 2017), juga disebut YOLO9000, adalah versi penyempurnaan dari *Yolo v1*. Jika dibandingkan dengan metode deteksi obyek yang sejenis yang berbasis *region proposal* seperti *Fast R-CNN*, *Yolo v1* menunjukkan sebagian besar kesalahan terdapat pada lokalisasi dan memiliki *recall* yang relatif rendah. Dengan demikian perbaikan *YOLO* difokuskan pada peningkatan *recall* dan lokalisasi dengan tetap mempertahankan akurasi klasifikasi. Ringkasan perbaikan dan hasil yang diberikan ditunjukkan pada Tabel 2.1.

Tabel 2.1 Perbaikan *Yolo v1* ke *Yolo v2.*

Perbaikan	*Yolo*							*Yolo v2*
Batch normalization		X	X	X	X	X	X	X
High-res classifier			X	X	X	X	X	X
Anchor boxes				X				
BB-dimension priors					X	X	X	X
Passthrough layer						X	X	X
Multi-scale training							X	X
Highest-res detector								X
VOC2007 *mAP*	63,4	65,8	69,5	69,2	74,4	75,4	76,8	78,6

2.3.7.2.1 Arsitektur *Yolo v2.*

Yolo v2 menggunakan arsitektur ekstraksi fitur yang disebut dengan Darknet-19, dimana Darknet-19 menggunakan 19 lapisan konvolusi dan 5 lapisan *maxpooling*. Struktur jaringan sebagian besar terdiri dari *filter* berukuran 3×3 dan jumlah *filter* dilipatgandakan setiap setelah operasi *pooling*. Untuk melakukan prediksi, *global average pooling* diterapkan menggunakan *filter* 1×1 untuk memampatkan representasi fitur diantara konvolusi 3×3. Secara lebih rinci arsitektur darknet-*19* ditunjukkan pada Tabel 2.2.

Tabel 2.2 Arsitektur *Yolo v2.*

	Tipe	*Filter*	Ukuran/*Stride*	*Output*
	conv	32	$3 \times 3/1$	224×224
	maxpool		$2 \times 2/2$	112×112
	conv	64	$3 \times 3/1$	112×112
	maxpool		$2 \times 2/2$	56×56
	conv	128	$3 \times 3/1$	56×56
	conv	64	$1 \times 1/1$	56×56
	conv	128	$3 \times 3/1$	56×56
	maxpool		$2 \times 2/2$	28×28
	conv	256	$3 \times 3/1$	28×28
	conv	128	$1 \times 1/1$	28×28
	conv	256	$3 \times 3/1$	28×28
	maxpool		$2 \times 2/2$	14×14
$3 \times$	conv	512	$3 \times 3/1$	14×14
	conv	256	$1 \times 1/1$	14×14
	maxpool		$2 \times 2/2$	7×7
$2 \times$	conv	1024	$3 \times 3/1$	7×7
	conv	512	$1 \times 1/1$	7×7
	conv	1024	$3 \times 3/1$	7×7
	conv	1000	$1 \times 1/1$	7×7
	Avgpool		Global	1000
	softmax			

2.3.7.3 Yolo v3.

Salah satu fokus perbaikan yang dilakukan pada *Yolo v3* adalah upaya untuk melakukan deteksi terhadap obyek yang berukuran kecil. Karena pada *Yolo v2* banyak terjadi *error* ketika *Yolo v2* diimplementasikan untuk melakukan deteksi pada obyek berukuran kecil. Poin-poin yang menjadi perbaikan pada *Yolo v3* adalah:

1. Prediksi *Bounding Box*.

Prediksi *bounding box Yolo v3* sama dengan *YOLO v2*. Selama pelatihan *YOLO v3* menggunakan *Sum Square Error Loss*. *YOLO v3* memprediksi *objectness score* untuk setiap *bounding box* menggunakan *logistic regression*. *Yolo v3* hanya menetapkan satu *bounding box anchor box* untuk setiap objek *ground truth*. Jika *bounding box anchor box* tidak ditugaskan ke objek *ground truth*, maka tidak ada *loss* untuk prediksi koordinat atau kelas, hanya *objectness*.

2. Prediksi Kelas.

Yolo v3 mengganti fungsi *softmax* dengan *classifier logistic independent* untuk menghitung kemungkinan *input* termasuk pada label tertentu. Dibanding menggunakan *mean square error* dalam menghitung *loss* klasifikasi, *Yolo v3* menggunakan *cross-entropy loss biner* untuk setiap label. Hal ini bertujuan untuk mengurangi kompleksitas komputasi dengan tidak menggunakan fungsi *softmax*.

3. Prediksi Lintas Skala.

Prediksi lintas skala bertujuan untuk menyempurnakan kelemahan yang ada pada *Yolo v2*, dimana *Yolo v2* kurang peka terhadap obyek yang memiliki ukuran kecil. Prediksi lintas skala terdiri dari tiga ukuran yaitu:

- Pada layer *feature map* terakhir dengan ukuran *output feature map* 13×13.
- Kemudian dilakukan operasi *upsample* dengan faktor 2 sehingga *Yolo v3* menghasilkan *feature map* dengan resolusi lebih tinggi dengan ukuran *output feature map* 26×26..
- Kemudian dilakukan operasi *upsample* kembali dengan faktor 2 sehingga *Yolo v3* menghasilkan *feature map* dengan resolusi lebih tinggi dengan ukuran *output feature map* 52×52..

4. *Anchor box.*

Untuk menentukan *anchor box*, *Yolo v3* menggunakan *k-means clustering*, kemudian dipilih 9 *cluster*. Untuk COCO, lebar dan tinggi *anchor box* adalah (10 × 13), (16 × 30), (33 × 23), (30 × 61), (62 × 45), (59 × 119), (59 × 119), (116 × 90), (156 × 198), (373 × 326). 9 *anchor box* ini dikelompokkan ke dalam 3 grup yang berbeda. Setiap grup ditugaskan ke *feature map* tertentu di atas dalam mendeteksi objek.

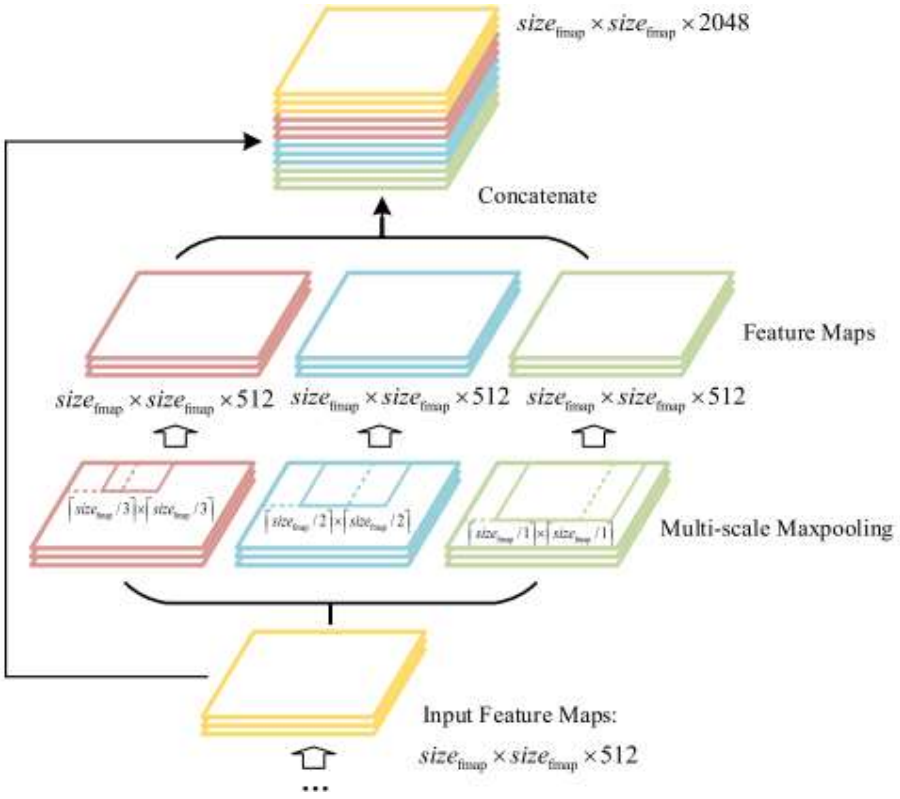
2.3.7.3.1 Arsitektur *Yolo v3.*

Yolo v3 menggunakan *feature extractor* 53-layer, *darknet-53* yang digunakan untuk menggantikan *darknet-19*. *Darknet-53* terdiri dari *filter* 3 × 3 dan 1 × 1 dengan *skip connection* seperti jaringan residual di ResNet. *Darknet-53* memiliki *BFLOP* lebih sedikit daripada ResNet-152, tetapi mencapai akurasi klasifikasi yang sama pada kecepatan 2x lebih cepat. Total keseluruhan layer yang digunakan pada arsitektur *Yolo v3* untuk ekstraksi fitur, klasifikasi dan deteksi sebanyak 106 layer. Secara lebih rinci konfigurasi arsitektur *Yolo v3* akan dijelaskan pada sub bab 2.3.9.

2.3.8 *Spatial Pyramid Pooling.*

Prediksi multi-skala *Yolo v3* berfokus pada penggabungan global fitur lapisan konvolusi multi-skala dan mengabaikan perpaduan fitur multi-skala pada lapisan konvolusi yang sama. Dengan penambahan *Spatial Pyramid Pooling (SPP)*, maka fitur multi-skala global dan lokal digunakan bersama-sama untuk meningkatkan akurasi deteksi objek. Ilustrasi penambahan *SPP* pada *Yolo v3* ditunjukkan pada Gambar 2.17.

Penambahkan *SPP* pada layer sebelum deteksi obyek terdiri dari tiga *max-pooling* layer, filter dengan ukuran $5 \times 5, 9 \times 9$, dan 13×13 sehingga diperoleh tiga *feature map* dengan ukuran $13 \times 13 \times 512$. Kemudian tiga *feature map* lokal dan satu *input feature map* global digabung untuk memperoleh output *feature map* dengan ukuran $13 \times 13 \times 2048$ yang mengekstraksi dan menyatukan fitur multi-skala sebagai *output* untuk deteksi objek.



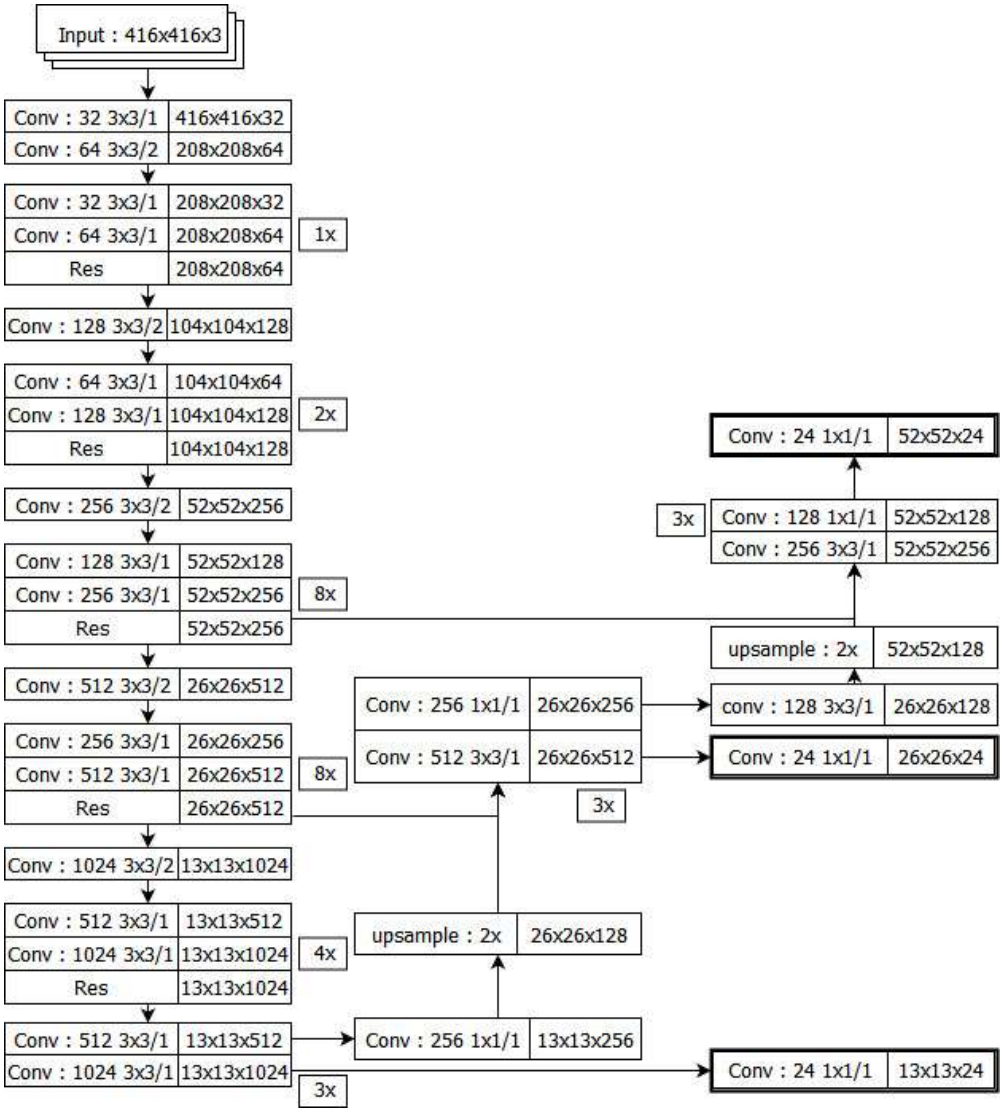
Gambar 2.17 *Spatial Pyramid Pooling* pada *Yolo v3* (Huang dan Wang, 2019).

2.3.9 Konfigurasi *Yolo v3*.

Pada penelitian ini akan dilakukan deteksi kerusakan jalan menggunakan tiga varian arsitektur *Yolo v3* yaitu *Yolo v3, Yolo v3 Tiny* dan *Yolo v3 SPP*. Oleh karena itu konfigurasi lebih detail dari masing-masing arsitektur akan dijelaskan pada sub bab ini.

2.3.9.1 Konfigurasi Yolo v3.

Yolo v3 pada proses ekstraksi fitur menggunakan kombinasi operasi konvolusi dengan *filter* berukuran 3×3, 1×1, dan *stride* 1 untuk lapisan konvolusi dan *stride* 2 sebagai lapisan *pooling*. *Darknet 53* adalah *backbone* yang digunakan untuk proses ekstraksi fitur, yaitu terdiri dari 53 lapisan *convolutional*, lima lapisan *pooling* dan satu lapisan keluaran berupa *feature map* dengan ukuran $3 \times 13 \times 13 \times (5 + C)$. Setelah diperoleh *feture map* dengan ukuran tersebut maka dilakukan operasi *upsampling* dengan faktor 2 sebanyak dua kali. Proses *upsampling* bertujuan untuk membuat *Yolo v3* lebih peka terhadap berbagai ukuran gambar. Konfigurasi arsitektur *YOLO v3* ditunjukkan pada Gambar 2.18.

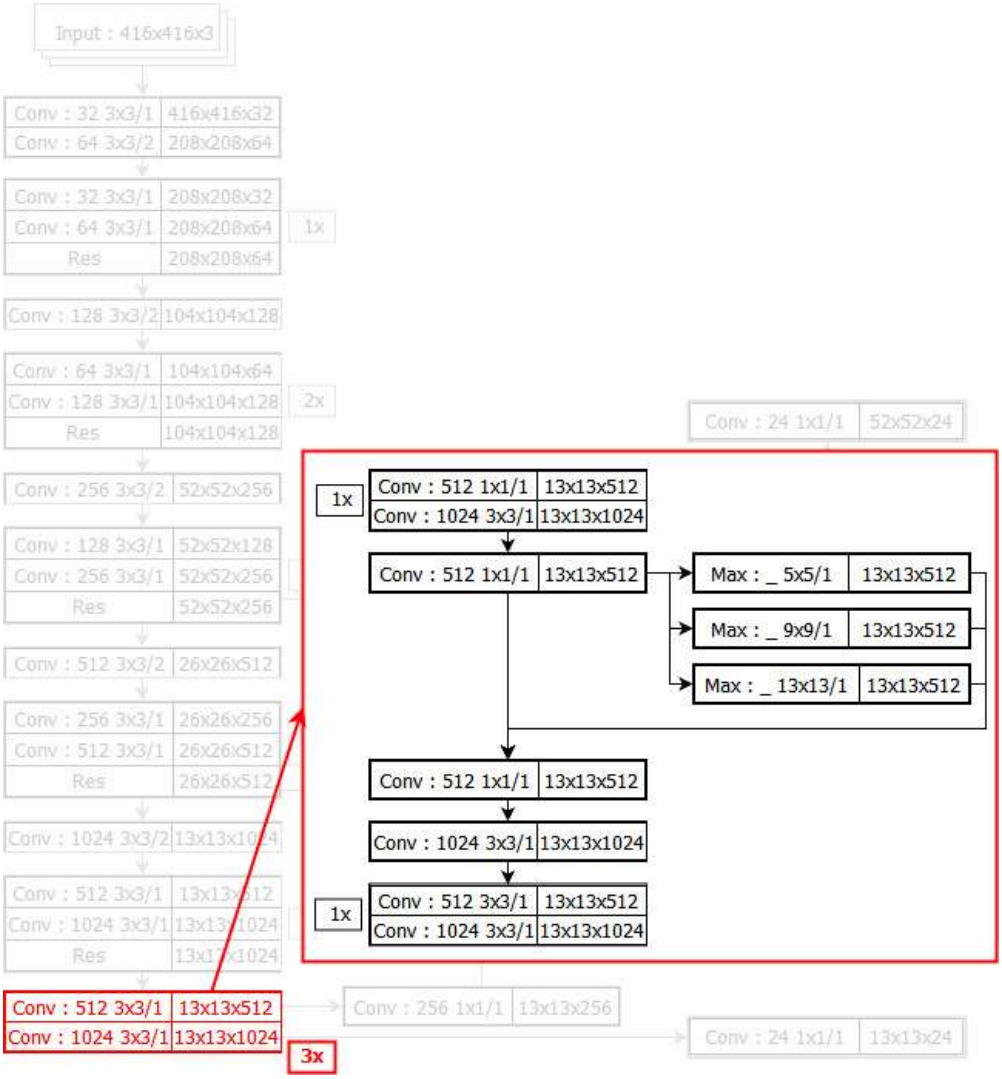


Gambar 2.18 Konfigurasi arsitektur Yolo v3.

2.3.9.2 Konfigurasi Yolo v3 SPP.

Yolo v3 SPP menggunakan ekstraksi fitur yang hampir sama dengan *Yolo v3*, namun ditambahkan *SPP*. Penambahan *SPP* dilakukan setelah lapisan konvolusi ke-50, dengan menambahkan tiga operasi *pooling* menggunakan *filter* berukuran $5 \times 5, 9 \times 9$, dan 13×13 untuk mendapatkan fitur lokal. Dari *output* konvolusi pada lapisan ke-50 dan tiga fitur lokal tersebut dilakukan penggabungan. Penggabungan dari *output* fitur global dan 3 fitur lokal masing-masing dengan dimensi 512 sehingga menghasilkan input dengan dimensi 2048 dan dilakukan operasi konvolusi sehingga menjadi dimensi 512 kembali dan dilanjutkan untuk

dilakukan proses ekstraksi fitur sampai diperoleh *output feature map* yang sama dengan *Yolo* v3, yaitu *feature map* dengan ukuran $3 \times 13 \times 13 \times (5 + C)$. Kemudian dilakukan operasi deteksi selanjutnya menggunakan operasi konvolusi yang sama sebagaimana dilakukan pada *Yolo v3.* Adapun konfigurasi *Yolo v3 SPP* secara rinci ditunjukkan pada Gambar 2.19.

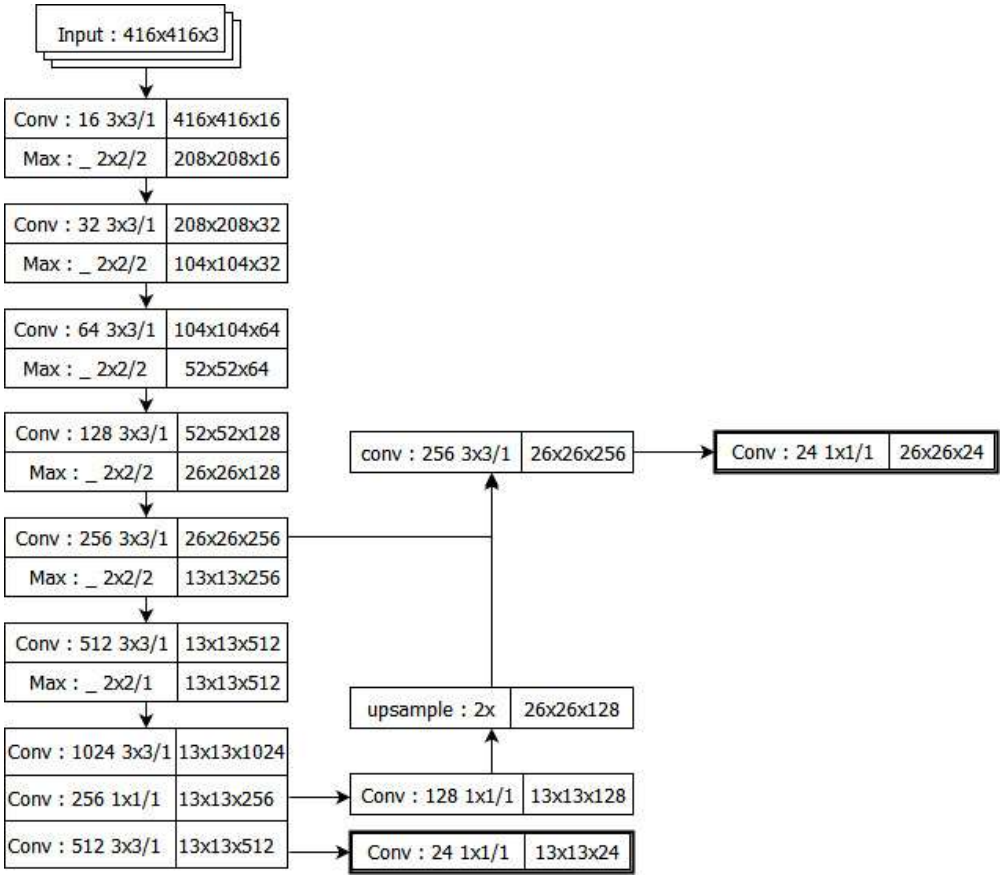


Gambar 2.19 Konfigurasi arsitektur Yolo v3 SPP.

2.3.9.3 Konfigurasi Yolo v3 Tiny.

Yolo v3 Tiny menggunakan operasi konvolusi yang lebih sederhana yang bertujuan agar proses komputasi baik pada proses pemodelan dan pengujian berjalan lebih cepat dari versi lainnya. Namun karena jaringan *Yolo v3 Tiny* lebih

sederhana maka pasti ada *trade off* antara waktu komputasi dengan akurasi. *Yolo v3 Tiny* menggunakan ekstraksi fitur hanya dengan Sembilan lapisan konvolusi dengan ukuran *filter* 3×3 dan *stride* 1 serta lapisan *pooling* dengan ukuran *filter* 2×2 dan *stride* 2. Untuk proses *upsampling* dengan faktor 2, pada *Yolo v3 Tiny* hanya dilakukan sekali. Konfigurasi arsitektur *Yolo v3 Tiny* ditunjukkan pada Gambar 2.20.

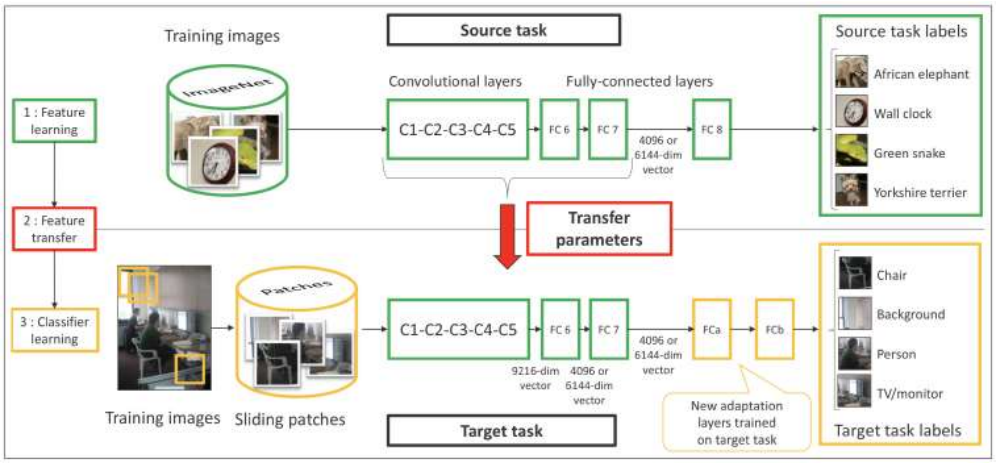


Gambar 2.20 Konfigurasi arsitektur Yolo v3 Tiny.

2.3.10 *Transfer Learning* dan *Fine Tuning*.

Tranfer learning bertujuan untuk men-*transfer* pengetahuan antara sumber terkait dan domain target (Oquab *et al.*, 2014). Keterbatasan data dan mahalnya biaya komputasi, apalagi pada komputasi *CNN,* maka jarang sekali proses pemodelan/pelatihan dilakukan dari awal, namun menggunakan skema *Transfer learning* menggunakan *pre-trained network* yang sudah ada untuk diadaptasi sebagai inisialisasi atau *feature extractor* untuk tugas baru yang dikehendaki.

Proses *transfer learning* dilakukan dengan cara melakukan *training* hanya pada beberapa *layer* terakhir sebuah *network.* Sebagai contoh ketika menggunakan bobot *pre-trained* sebagai *feature extractor*, maka yang dapat dilakukan adalah dengan melakukan *training* pada beberapa lapisan terakhir saja. Sehingga lapisan awal sampai dengan sebelum lapisan terakhir bobot yang ada masih tetap, dan perubahan bobot hanya terjadi pada misalnya dua lapisan terakhir. *Transfer Learning* dapat dilakukan sesuai dengan kebutuhan, jika data yang dimiliki berjumlah sangat besar maka *transfer learning* dapat diterapkan pada lebih banyak layer terakhir. Sebagai contoh jika terdapat arsitektur *CNN* dengan lima lapisan *convolutional* dan dua lapisan *fully connected* dan akan dilakukan proses *transfer learning,* salah satu cara yang dilakukan yaitu dengan melakukan pemodelan pada dua *fully connected layer* yang ada. Sehingga akan diperoleh bobot baru sesuai dengan data baru yang digunakan. Perubahan hanya terjadi pada dua lapisan terakhir sedangkan bobot pada lima lapisan *convolutional* masih mempunyai bobot yang lama. Ilustrasi proses *transfer learning* dapat ditunjukkan pada Gambar 2.21.



Gambar 2.21 Contoh *transfer learning* di mana dua lapisan *fully connected* yang terakhir dihapus, dan dua lapisan adaptasi baru ditambahkan ke jaringan. Kedua lapisan ini kemudian disesuaikan untuk melakukan tugas baru (Oquab *et al.*, 2014).

Pada *Yolo*, *transfer learning* dilakukan untuk melatih jaringan yang awalnya berfungsi sebagai *classifier* dan *detector* dipotong sebanyak tujuh puluh tiga lapisan untuk memperoleh bobot dari *pre-trained*-nya. Tujuh puluh tiga lapisan ini berfungsi sebagai *classifier* saja, sehingga pada proses *training* akan dilakukan

tuning untuk mendapatkan bobot *classifier* sekaligus *detector* yang baru yang dikombinasikan dengan bobot ekstraksi fitur *classifier* yang lama (dari *pre-trained*).

Alternatif *transfer learning* adalah *fine tuning,* dimana *fine tuning* memberi keuntungan dari sisi waktu *training* dimana waktu *training* pada *CNN* cenderung lama dan *fine tuning* dapat diproses lebih cepat. Namun ada *trade-off* dengan akurasi yang dihasilkan. Pada *Yolo, fine-tuning* dilakukan untuk melatih jaringan yang awalnya berfungsi sebagai *classifier* dibekukan dan hanya jaringan yang berfungsi sebagai *detector* yang akan di-*tuning*. Sehingga pada proses *training* akan diperoleh bobot *classifier* yang sama dengan bobot *pre-trained*-nya dan bobot *detector* yang baru (AlexeyAB, 2019).

2.4 Evaluasi.

2.4.1 Evaluasi Deteksi.

Sebagaimana dijelaskan pada subbab 2.3.6, tugas yang diselesaikan oleh deteksi obyek terdiri dari klasifikasi dan lokalisai maka evaluasi diterapkan pada klasifikasi dan juga lokalisasi. Untuk menentukan akurasi klasifikasi dapat dilakukan dengan menganalisis *precision, recall*, dan skor *F1* dari suatu model. *Precision* dihitung sebagai jumlah prediksi yang benar dari semua prediksi kelas tertentu sebagaimana ditunjukkan pada persamaan (2.8).

$$Precision = \frac{TP}{TP + FP} \tag{2.8}$$

Dimana:

TP adalah klasifikasi yang benar terhadap obyek.

FP adalah klasifikasi yang salah terhadap obyek.

Recall dihitung sebagai jumlah prediksi yang benar dari semua kasus kelas tertentu sebagaimana ditunjukkan pada persamaan (2.9).

$$Recall = \frac{TP}{TP + FN} \tag{2.9}$$

Dimana:

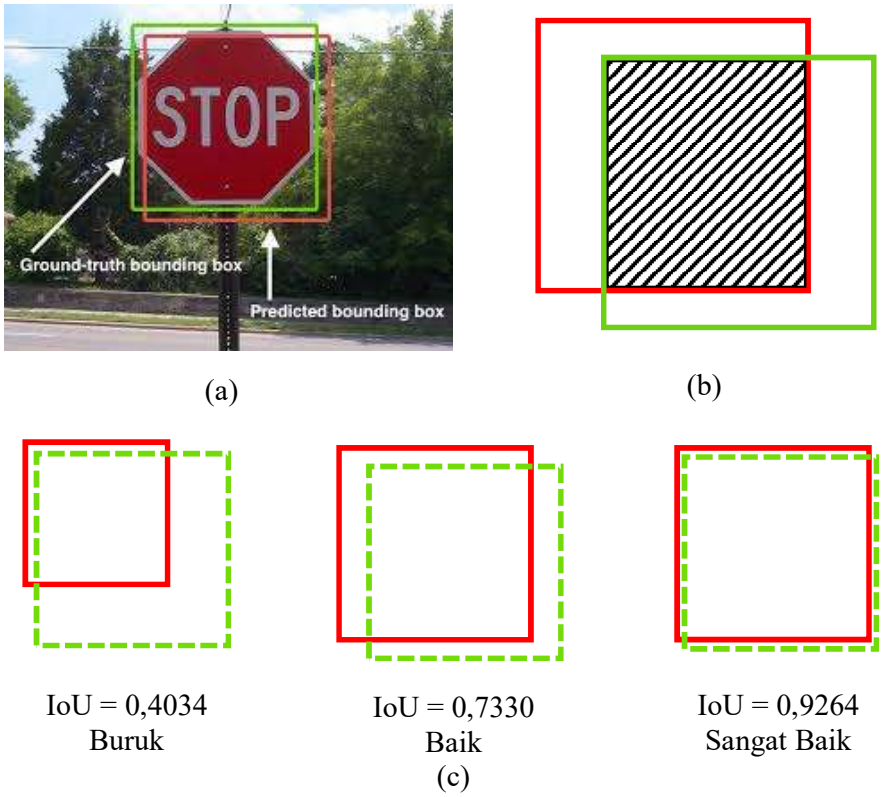
TP adalah klasifikasi yang benar terhadap obyek.

FN adalah klasifikasi yang seharusnya diklasifikasi pada kelas tertentu namun tidak diklasifikasi dengan tepat.

Sedangkan skor *F1* adalah rata-rata harmonik dari *precision* dan *recall* sebagaimana ditunjukkan pada persamaan (2.10).

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \qquad 2.10$$

Untuk menentukan akurasi lokalisasi dapat digunakan *IoU* yaitu prosentase area irisan antara deteksi dengan *ground truth* dengan keseluruhan obyek baik hasil deteksi maupun *ground truth*. Seperti diilustrasikan pada Gambar 2.22 a, dimana terdapat obyek berupa rambu-rambu lalu lintas dimana terdapat kotak berwarna hijau adalah *ground truth* dan kotak berwarna merah adalah hasil deteksi.



Gambar 2.22 Ilustrasi *IoU*. a. ilustrasi antara deteksi dan *ground truth*. b. ilustrasi irisan deteksi dan *ground truth*. c. ilustrasi nilai *IoU*.

Dari kedua kotak tersebut dapat dilakukan perhitungan dengan membandingkan seberapa besar irisan dari kedua kotak tersebut (bagian yang diarsir) dengan keseluruhan area baik *ground truth* maupun hasil deteksi sebagaimana diilustrasikan pada Gambar 2.22 b. Nilai *IoU* berkisar antara 0-1, semakin besar mendekati satu maka akurasi semakin baik, dapat diambil pembagian

misalnya skor *IoU* sekitar 0,4 dianggap buruk, sedangkan 0,7 dianggap baik dan 0,9 sangat baik sebagaimana diilustrasikan pada Gambar 2.22 c. Tidak ada angka pasti yang merupakan skor *IoU* yang baik, akan tetapi pada penelitian ini menggunakan ambang batas *IoU* sebesar 0.5 sesuai dengan matrik evaluasi yang diterapkan pada COCO dan pada pengujian *Yolo v3* (Redmon dan Farhadi, 2018). Dari ilustrasi pada Gambar 2.22, jika area *ground truth* didefiniskan sebagai GT dan area deteksi didefinisikan sebagai P, maka besaran IoU dapat diperoleh dengan persamaan (2.11).

$$IoU = \frac{GT \cap P}{GT \cup P} \tag{2.11}$$

Pengukuran deteksi objek penting lainnya adalah *Average Precision (AP)*. *AP* adalah nilai angka tunggal yang merepresentasikan *recall* dan *precission* yang biasa digunakan dalam penilaian kinerja deteksi obyek. *AP* disebut juga sebagai *Area Under the Curve (AUC)* yaitu perpaduan nilai dibawah kurva *precission* dan *recall*. Dimana *recall* direpresentasikan dalam sumbu x dan *precision* dalam sumbu y, sehingga terbentuk area dibawah titik-titik antara sumbu x dan sumbu y yang disebut dengan *average precision* (Boyd, Eng dan Page, 2013). *AP* dihitung dengan mengambil beberapa set peringkat deteksi obyek dengan *confidence* tertinggi. Masing-masing peringkat tersebut dihitung nilai *precision* dan *recall*-nya, dan yang terakhir dilakukan perhitungan nilai rata-rata *precision* atau dikenal dengan *AP* sebagaimana dapat dinotasikan dengan persamaan (2.12).

$$Average\ Precision\ (AP) = \frac{1}{k} \sum_{j=1}^{k} P_j(r) \tag{2.12}$$

Dimana k adalah jumlah set yang dipilih dan $P_j(r)$ adalah nilai *precision* pada titik ke j dengan *recall* r.

AP merupakan nilai rata-rata *precision* untuk masing-masing kategori, sehingga jika terdapat beberapa kategori dalam suatu sistem deteksi obyek maka rata-rata *AP* atau disebut *mean Average Precision (mAP)* dapat diperoleh dengan menjumlahkan semua nilai *AP* dari masing-masing kategori dan membaginya sebanyak *n* kategori sesuai dengan persamaan (2.13) (Beitzel, Jensen dan Frieder, 2009).

$$MAP = \frac{1}{n} \sum_{n} AP_n \quad (2.13)$$

2.4.2 Evaluasi Perhitungan Luas.

Output pada penelitian ini terdiri dari kelas kerusakan jalan dan luas area kerusakan, sehingga perlu dilakukan perhitungan akurasi luas kerusakan jalan. Secara umum, perhitungan luas dapat ditulis sebagai $L = p \times l$ dengan p merupakan variabel panjang obyek dan l adalah variabel lebar obyek. Perhitungan luas diterapkan baik pada obyek *ground trouth* maupun pada obyek hasil deteksi.

Evaluasi perhitungan luas dapat diperoleh dengan menghitung selisih luas *ground truth* dengan nilai luas hasil deteksi. Dimana luas *ground truth* didefinisikan sebagai L dan luas hasil deteksi didefinisikan sebagai L'. Dari kedua luas tersebut maka dapat diperoleh selisih luas total dalam satuan meter untuk masing-masing kerusakan sebagai S, yang dinotasikan dengan persamaan (2.14). S' sebagai selisih/*error* dalam prosentase, yang dinotasikan dengan persamaan (2.15). Dan akurasi perhitungan luas dalam prosentase sebagai Acc, yang dinotasikan dengan persamaan (2.16).

$$S = |L - L'| \quad (2.14)$$

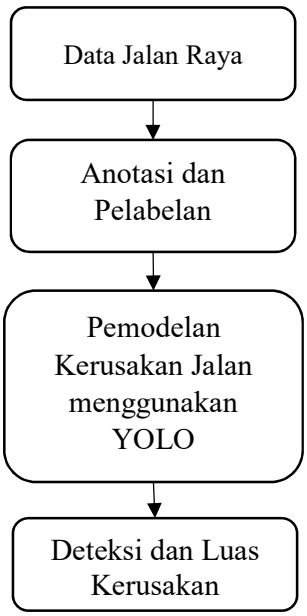
$$S' = \frac{S}{L} \times 100\% \quad (2.15)$$

$$Acc = 100 - S'\% \quad (2.16)$$

BAB 3
METODE PENELITIAN

3.1 Alur Penelitian.

Secara garis besar alur yang dilakukan pada penelitian ini adalah persiapan data jalan raya sebagai bahan penelitian, anotasi dan pelabelan, pemodelan kerusakan jalan menggunakan *Yolo*, serta deteksi dan perhitungan luas kerusakan jalan sebagaimana di tunjukkan pada Gambar 3.1.



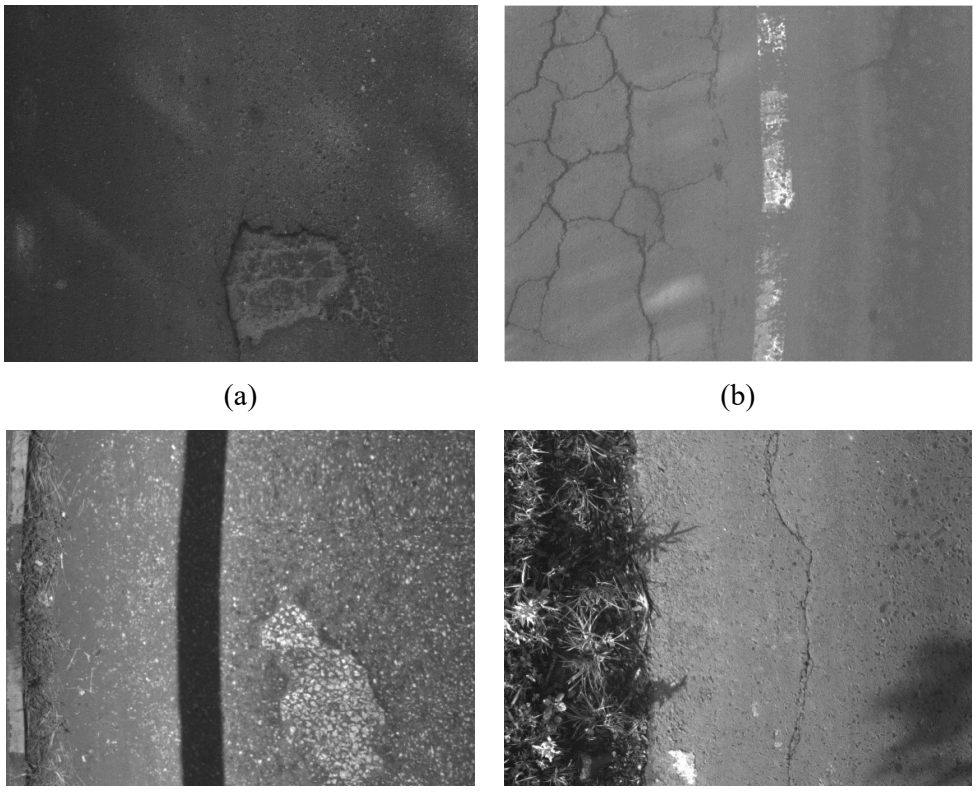
Gambar 3.1 Alur Penelitian.

3.2 Data Jalan Raya.

Data Jalan Raya sebagai bahan penelitian diperoleh dari data citra yang dihasilkan oleh kamera *pavement view* pada kendaraan survei *Hawkeye 2000.* Kamera *pavement view* dipasang pada bagian belakang kendaraan survei dan menghadap kebawah sebagaimana di tunjukkan pada Gambar 2.1. Proses pengambilan gambar dilakukan pada pagi, siang dan sore hari serta dalam kondisi tidak terjadi hujan. Oleh karena itu citra yang dihasilkan memiliki ragam pencahayaan yang bermacam-macam. Kecepatan kendaraan pada saat pengambilan gambar minimal 20 km/jam. Hal ini dikarenakan menyesuaikan dengan sensor lain

yang tidak dapat berfungsi jika dijalankan pada kecepatan dibawah 20 km/jam. Adapun proses pengambilan gambar dilakukan di Propinsi Jawa Timur pada tahun 2016.

File yang dihasilkan dari kamera *pavement view* adalah *video* hitam putih dengan *format .AVI*. Agar dapat digunakan sebagai materi pada penelitian ini, format *video* dikonversi menjadi data citra menggunakan *software* pelengkap *Hawkeye Processing Toolkit.* Proses konversi menghasilkan berbagai ukuran citra yang dapat dipilih yaitu: $1280 \times 960, 1624 \times 1234, 2048 \times 1536$ dan 512×384 piksel dimana ukuran tersebut mewakili $2{,}061 \times 1{,}391$ meter dengan format *.JPG*.



(a) (b)

(c) (d)

Gambar 3.2 Data citra jalan raya dengan kerusakan. a. Lubang. b. Retak. c. Trotoar. d. Bahu Jalan.

Proses konversi menghasilkan banyak data citra jalan, namun tidak semua data tersebut digunakan akan tetapi hanya data dengan kerusakan dan jenis perkerasan aspal yang digunakan sebagai bahan penelitian. Sehingga pada tahap ini

perlu dilakukan pemilihan citra yang akan digunakan sebagai sampel penelitian. Pengamatan visual dilakukan pada tahap ini terhadap semua citra jalan. Sebagai langkah awal pemilihan ruas yang menjadi target dapat dilihat dari laporan lain yang dihasilkan oleh kendaraan survei. Sehingga diharapkan hanya ruas-ruas yang memiliki nilai tidak bagus yang diamati dan jumlahnya tidak sebanyak data mentah sebelumnya. Dari ruas-ruas tersebut dihasilkan puluhan ribu citra, dan akhirnya diperoleh sebagian citra dari sepuluh kelompok ruas jalan nasional yang ada di Provinsi Jawa Timur. Adapun contoh data citra kerusakan jalan ditunjukkan pada Gambar 3.2 dan secara lebih rinci ruas jalan nasional yang digunakan pada penelitian ini ditunjukkan pada Tabel 3.1.

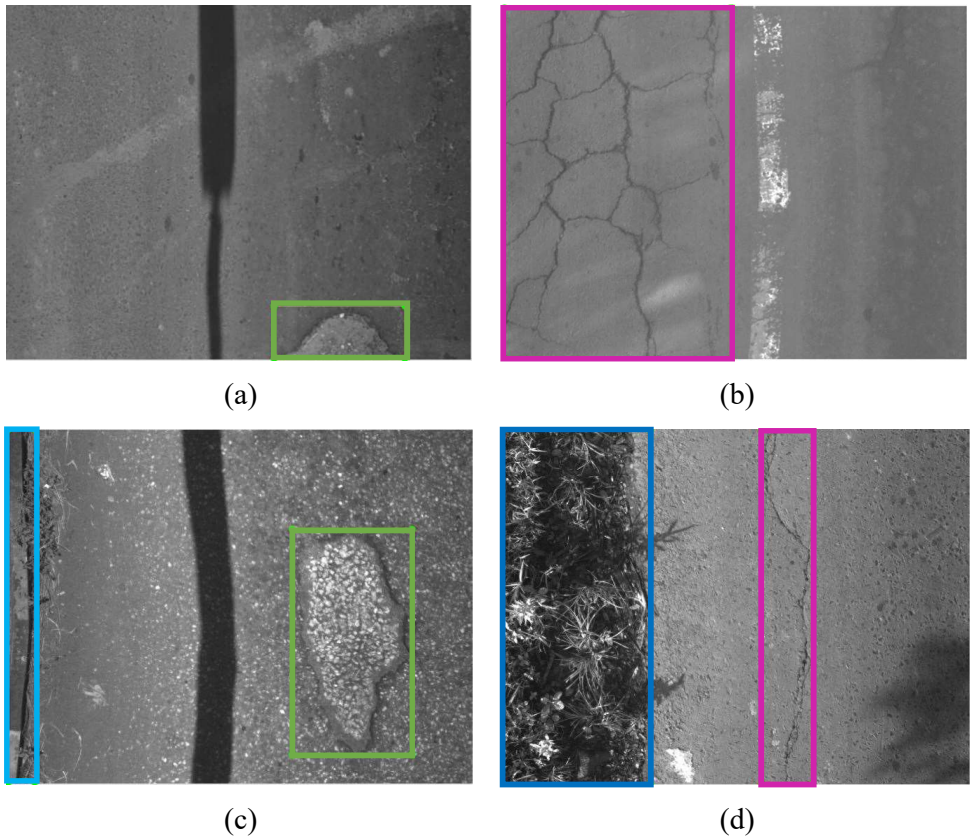
Tabel 3.1 Daftar ruas jalan nasional yang digunakan.

No.	Nama Ruas	Panjang Jalan (km)
1	Pelabuhan Tj. Bumi – Bts. Kab. Bangkalan	55,176
2	Widang – Bts. Kota Lamongan – Pakah Temangkar	49,302
3	Lohgung - Sadang	76,719
4	Cepu - Padangan	
5	Bts. Kota Probolinggo - Paiton	43,302
7	Taman – Waru / Bts. Kab. Jombang - Gemekan	51,690
8	Talok – Druju – Sendang Biru	41,705
9	Jln. Sudirman (Jombang) / Bts. Kab Madiun - Nganjuk	53,739
10	Jl. Raya Madiun Ponorogo	31,167
11	Bts. Kota Jombang – Bts. Kab. Mojokerto / Basuki Rahmat (Jombang)	21,126
	Total	42,884

Pada penelitian ini dilakukan percobaan dengan dua target keluaran. Pada percobaan pertama menggunakan lubang sebagai target keluaran sedangkan pada percobaan kedua menggunakan tiga kelas keluaran yaitu lubang, retak dan lainnya (bahu jalan dan trotoar). Pemilihan tiga kelas ini didasarkan pada karakteristik data jalan raya yang ditemui. Beberapa citra jalan yang sudah dikumpulkan selain menampilkan obyek jalan raya juga menampilkan obyek bahu jalan dan trotoar dikarenakan lebar jalan cukup kecil dan tentunya kondisi ini juga banyak ditemui pada jalan nasional di Indonesia.

3.3 Anotasi dan pelabelan.

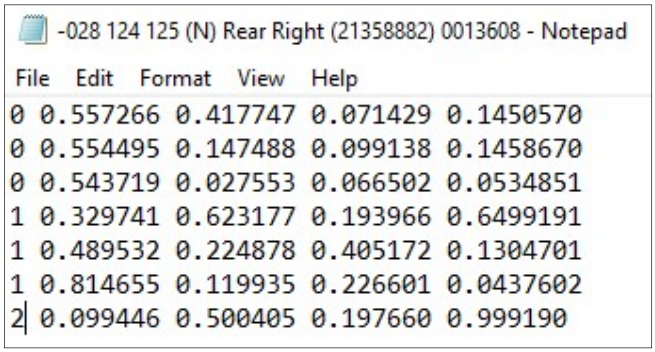
Anotasi dan pelabelan dilakukan dengan cara melakukan *tagging* berupa *bounding box* pada bagian citra yang terdapat lubang, retak dan obyek lainnya (kerusakan lain, trotoar, atau bahu jalan) dan memberikan label jenis kerusakannya. *Tagging* berupa *bounding box* dengan sisi-sisi tepat berada dibagian terluar kerusakan. Proses ini dilakukan satu persatu terhadap semua data latih dan data uji. Pada proses ini *tools LabelImg* (Tzutalin, 2018) digunakan untuk menghasilkan pola inputan sesuai arsitektur *Yolo*. Proses pelabelan mengacu pada petunjuk survei Indeks Kerusakan Jalan (Kementerian Pekerjaan Umum dan Perumahan Rakyat, 2016b). Contoh anotasi dan pelabelan ditunjukkan pada Gambar 3.3.



Gambar 3.3 Contoh anotasi. a. Anotasi kelas lubang. b. Anotasi kelas retak. c. Anotasi kelas lainnya dan kelas lubang. d. Anotasi kelas lainnya dan kelas retak.

Informasi yang disimpan pada tahap ini terdiri dari lima variabel, yaitu id kelas, koordinat pusat *bounding box* (x dan y), dimensi panjang dan lebar *bounding box* (w dan h). Id kelas berupa nilai *integer* yang dimulai dari 0, sedangkan pusat,

panjang, dan lebar *bounding box* berupa angka desimal dalam rentang 0-1. *File* hasil anotasi disimpan dalam format *.txt* dengan nama *file* sama dengan nama *file* citra. Jadi setiap satu *file* citra dengan *format .jpg* akan mempunyai satu *file* anotasi dengan format *.jpg.* Contoh *output* anotasi ditunjukkan pada Gambar 3.4.



Gambar 3.4 Contoh output anotasi dan pelabelan.

Setelah proses anotasi selesai dilakukan, *file* citra dan *file* anotasi dipisahkan dalam *folder* yang berbeda sesuai dengan pola masukan *framework darknet* yang digunakan sebagai *backbone* arsitektur *Yolo*. *Darknet* adalah *framework neural network* yang ditulis dalam Bahasa C dan *CUDA*. Dengan *Framework* ini, komputasi dapat dilakukan lebih cepat karena mendukung komputasi baik *CPU* maupun *GPU*.

3.4 Pemodelan Kerusakan Jalan menggunakan *Yolo*.

Pada penelitian ini pemodelan kerusakan jalan dibangun menggunakan arsitektur *Yolo v3*. Pendekatan *single stage* digunakan sekaligus untuk menentukan klasifikasi dan lokasi kerusakan jalan. Citra yang diinputkan akan dibagi menjadi $S \times S$ *grid* yang disebut dengan *grid cell.* Setiap *grid cell* akan digunakan untuk memprediksi *bounding box* yang memuat informasi pusat, panjang dan tinggi *bounding box* serta nilai probabilitas hasil pengenalan. Setiap *grid cell* hanya melakukan prediksi satu obyek saja. Sebagai inisisalisasi awal bentuk *bounding box*, inisial *anchor* yang digunakan merujuk pada distribusi ukuran obyek COCO dan VOC2007 yaitu sebanyak sembilan inisial *anchor* (Redmon dan Farhadi, 2017).

Proses pemodelan kerusakan jalan dibangun menggunakan *framework darknet* dengan bahasa *C* dan *CUDA* yang dijalankan melalui perantara *Google*

Colaboratory berbasis *Jupyter notebook* (*Colaboratory: Frequently Asked Questions*, 2019). Pemodelan dilakukan dengan skema *transfer learning* menggunakan model *pre-trained* yang sudah dilatih berdasarkan dataset *Imagenet. Output* pemodelan adalah model atau juga dikenal dengan bobot yang akan digunakan untuk melakukan deteksi.

Pemodelan merupakan langkah awal dalam melakukan deteksi kerusakan jalan. *Input*-an data citra akan dilakukan operasi konvolusi sesuai arsitektur *Yolo v3* untuk menghasilkan model sesuai dengan target *feature map* yang sudah ditentukan.

Adapun langkah-langkah yang dilakukan pada proses pemodelan kerusakan jalan menggunakan *Yolo v3* yaitu:

1. Tentukan arsitektur yang akan digunakan, dimana dalam penelitian ini menggunakan tiga varian arsitektur *Yolo v3* yaitu *Yolo v3, Yolo v3 Tiny* dan *Yolo v3 SPP*. Konfigurasi arsitektur berupa *file* yang memuat susunan arsitektur dan parameter-parameter yang digunakan dalam melakukan pemodelan.
2. Tentukan model *pre-trained* yang digunakan.
3. Tentukan parameter-parameter yang digunakan dalam pemodelan, yaitu: penentuan jumlah *batch* dalam setiap iterasi, tentukan ukuran *input*, dan target *feature map*. *Input* dengan ukuran $m \times n \times 3$ dimana $m \times n$ berupa bilangan dengan kelipatan 32 dan 3 adalah 3 *channel* inputan yang merepresentasikan 3 warna *r, g* dan *b*. Sedangkan target *feature map* adalah $3 \times (5 + C)$ dimana C adalah target kelas deteksi.
4. Lakukan konversi model *pre-trained* dengan konfigurasi arsitektur yang digunakan. Konfigurasi pada *file* no.1 harus sesuai dengan model *pre-trained* no.2.
5. Jalankan proses pemodelan.
6. Simpan model/bobot dalam setiap iterasi tertentu untuk memudahkan dalam memeriksa kinerja model yang dibangun. Pada penelitian ini dilakukan penyimpanan model/bobot setiap 100 iterasi. Hal ini bertujuan untuk mempermudah proses pemodelan dan juga evaluasi kinerja model yang dikembangkan. Karena proses pemodelan memerlukan waktu yang lama dan

rawan terjadi kegagalan, maka skenario ini dilakukan agar pemodelan dapat dilanjutkan dari iterasi terakhir yang tersimpan.

Pada penelitian ini dilakukan percobaan menggunakan beberapa varian arsitektur *Yolo v3* untuk melihat kinerja yang dapat dilakukan dari model yang dibangun. Beberapa varian arsitektur *Yolo v3* yang digunakan adalah *Yolo v3*, *Yolo v3 Tiny* dan *Yolo v3 SPP*. Pada proses pembangunan model ada dua tahap yang dijalankan menggunakan pendekatan *convolutional neural network,* tahap pertama yaitu tahap ekstraksi fitur sedangkan tahap berikutnya yaitu tahap klasifikasi dan lokalisasi.

3.5 Deteksi dan Perhitungan Luas.

Pada tahap ini dilakukan deteksi obyek menggunakan model/bobot yang dihasilkan pada proses pemodelan. Model yang dihasilkan dievaluasi terlebih dahulu untuk menentukan model yang memberikan akurasi terbaik. Model dengan akurasi terbaik akan digunakan untuk melakukan deteksi. Output deteksi berupa lima variabel untuk masing-masing citra yang diproses, yaitu id kelas, koordinat pusat *bounding box* (*x* dan *y*), dimensi panjang dan lebar *bounding box* (*w* dan *h*). *Output* tersebut digunakan untuk melakukan perhitungan luas untuk masing-masing kerusakan.

Dimana dalam perhitungan luas diperlukan nilai panjang p dan lebar l, maka sesuai dengan istilah yang digunakan dalam penelitian ini maka *p* dapat didefinisikan sebagai *w* dan *l* didefinisikan sebagai *h.* Dengan nilai *w* dan *h* berada pada rentang nilai antara 0-1 maka untuk memperoleh luas sesungguhnya perlu dilakukan perkalian dengan koefisien pengali panjang dan lebar sesungguhnya dari setiap *frame* citra yang digunakan dari *software hawkeye processing toolkit* yaitu sebesar $2{,}061 \times 1{,}391$ meter, sehingga di peroleh luas *ground truth* L sesuai dengan persamaan (3.1).

$$L = w \times h \times 2{,}061 \times 1{,}391\, m^2 \qquad 3.1$$

Dimana:

L adalah luas *ground truth.*

w adalah lebar obyek atau panjang obyek *ground dtruth.*

h adalah tinggi obyek atau lebar obyek *ground dtruth.*

Dan luas hasil deteksi L' diperoleh dengan persamaan (3.2).

$$L' = w' \times h' \times 2{,}061 \times 1{,}391\, m^2 \qquad 3.2$$

Dimana:

L' adalah luas hasil deteksi.

w' adalah lebar obyek atau panjang obyek hasil deteksi.

h' adalah tinggi obyek atau lebar obyek hasil deteksi.

3.6 Evaluasi.

Evaluasi dilakukan dua kali, yang pertama evaluasi terhadap hasil deteksi, dan yang kedua evaluasi terhadap hasil perhitungan luas kerusakan.

3.6.1 Evaluasi Deteksi.

Evaluasi yang digunakan dalam penelitian ini yaitu *Precission, Recall,* skor *F1, IoU* dan *mAP*. Penentuan *True Positif* pada deteksi obyek mengacu pada besaran *IoU*, dimana *IoU* (*Intersection Over Union*) adalah irisan antara *bounding box* hasil deteksi dengan *bounding box ground truth*. Sebagaimana yang umum dilakukan, *IoU* yang dipilih pada penelitian ini adalah 0.5. Sehingga obyek dengan *IoU* lebih dari 0.5 dinilai sebagai *true positif.*

3.6.2 Evaluasi Perhitungan Luas.

Langkah yang dilakukan pada evaluasi perhitungan luas yaitu:

1. Hitung luas masing-masing obyek kerusakan jalan dengan melakukan perkalian *w* dan *h* baik dari proses deteksi maupun *ground truth.*
2. Lakukan perkalian dengan faktor pengali panjang dan lebar sesungguhnya dari masing-masing *frame.*
3. Hitung total masing-masing luas kerusakan jalan baik prediksi maupun *ground truth*.
4. Hitung selisih antara deteksi dan *ground truth,* sehingga diperoleh nilai *error* hasil deteksi dan akurasi hasil deteksi luas kerusakan jalan.

Secara keseluruhan proses perhitungan akurasi luas dapat diilustrasikan dengan Tabel 3.2.

Tabel 3.2 Akurasi Perhitungan Luas.

No	Luas *ground truth* (m^2)	Luas Deteksi (m^2)	Selisih		Akurasi (%)
			(m^2)	(%)	
1	$L_1 = w_1 \times h_1$	$L_1' = w_1' \times h_1'$	$S_1 = \lvert L_1 - L_1' \rvert$	$S_1' = \left(\frac{S_1}{L_1}\right) \times 100$	$Acc_1 = 100 - S_1'$
2	$L_2 = w_2 \times h_2$	$L_2' = w_2' \times h_2'$	$S_2 = \lvert L_2 - L_2' \rvert$	$S_2' = \left(\frac{S_2}{L_2}\right) \times 100$	$Acc_2 = 100 - S_2'$
:	:	:	:	:	:
n	$L_n = w_n \times h_n$	$L_n' = w_n' \times h_n'$	$S_n = \lvert L_n - L_n' \rvert$	$S_n' = \left(\frac{S_n}{L_n}\right) \times 100$	$Acc_n = 100 - S_n'$
	$\sum L$	$\sum L'$	$\sum S$	$\sum S'$	$Acc = 100 - \sum Sn'$

3.7 Spesifikasi Perangkat.

Pada penelitian ini, pendekatan yang dipilih adalah pendekatan berbasis *deep learning* dengan arsitektur yang kompleks. Sehingga komputasi menjadi hal yang perlu diperhatikan baik dari sisi sumber daya maupun waktu yang diperlukan. Proses paling inti dari penelitian ini terkait dengan komputasi adalah proses pemodelan dan proses deteksi. Pada proses pemodelan dilakukan komputasi *cloud* menggunakan *GPU* melalui perantara *Google Colaboratory* berbasis *Jupyter notebook* (*Colaboratory: Frequently Asked Questions*, 2019). Sedangkan pada proses deteksi dilakukan komputasi berbasis *cloud* menggunakan *GPU* dan *offline* menggunakan *CPU.* Adapun spesifikasi perangkat komputasi berbasis *cloud* yang digunakan adalah:

1. GPU : Tesla T4, Cuda Core 2560, 16 GB GDDR6 300 GB/sec
2. CPU : Intel(R) Xeon(R) CPU @ 2,30GHz
3. Memori : 12 GB
4. Hardisk : 311 GB
5. Sistem Operasi : Ubuntu 18.04.

Sedangkan spesifikasi perangkat komputasi berbasis *offline* yang digunakan adalah:

1. GPU : -
2. CPU : Intel(R) Core(TM) i5-7200U CPU @ 2,50GHz
3. Memori : 4 GB
4. Hardisk : 1 TB
5. Sistem Operasi : Windows 10 Pro.

Halaman ini sengaja dikosongkan

BAB 4
HASIL DAN PEMBAHASAN

Bab ini menjelaskan proses deteksi kerusakan jalan menggunakan arsitektur *Yolo v3*. Berbagai konfigurasi dilakukan untuk memperoleh hasil kinerja arsitektur yang dipilih untuk melakukan deteksi kerusakan jalan pada citra jalan raya. Sebagaimana sudah disebutkan pada sub bab sebelumnya, penelitian ini dilakukan dua kali dengan dua target keluaran. Pada percobaan pertama target keluaran adalah lubang sedang pada percobaan kedua target keluaran adalah lubang, retak dan lainnya.

4.1 Percobaan Kelas Lubang.

Pada percobaan pertama dilakukan deteksi kerusakan jalan dengan target keluaran berupa kelas lubang. Masing-masing hasil metodologi penelitian yang dilakukan mulai dari penyiapan data penelitian sampai dengan evaluasi akan diuraikan secara rinci pada subbab ini. Pada percobaan lubang akan dilakukan satu kali pemodelan dan enam kali pengujian. Pengujian dilakukan untuk melihat kinerja model yang dikembangkan dengan prosentase data uji yang berbeda-beda.

4.1.1 Data Jalan Raya.

Data jalan raya dengan kerusakan lubang yang dipilih sebanyak 448 data dan dibagi menjadi data latih dan data uji. Prosentase pembagian data antara data latih dan data uji yaitu 50%:50%, 60%:40%, 70%:30%, 78%:22%, 80%:20% dan 90%:10%. Antar pengujian yang lebih awal dengan pengujian berikutnya mempunyai penurunan prosentase jumlah data sebesar 10%. Sehingga diperoleh pembagian data latih sebanyak 224 dan data uji berturut-turut sebanyak 224, 150, 96, 64, 56 dan 25 data. Sedangkan jumlah distribusi data kerusakan mempunyai prosentase mendekati prosentase data citra, yaitu 49%:51%, 58%:42%, 69%:31%, 79%:21%, 81%:19%, dan 91%:9%. Adapun distribusi data secara rinci ditunjukkan pada Tabel 4.1.

Tabel 4.1 Distribusi data dan kerusakan kelas lubang.

No.	Data Latih		Data Uji		Total	
	Jumlah Citra	Jumlah Kerusakan	Jumlah Citra	Jumlah Kerusakan	Jumlah Citra	Jumlah Kerusakan
1	224	279	224	296	448	575
2	224	279	150	199	374	478
3	224	279	96	124	320	403
4	224	279	64	73	288	352
5	224	279	56	64	280	343
6	224	279	25	27	249	306

4.1.2 Pemodelan menggunakan *Yolo*.

Pemodelan kerusakan jalan dengan target keluaran kelas lubang menggunakan *Yolo* berbasis operasi *neural network*. Setiap kali iterasi dalam membangun model menghasilkan *loss* yang menunjukkan kinerja deteksi terhadap data latih. Semakin banyak iterasi yang dilakukan maka *loss* yang dihasilkan juga cenderung semakin kecil dan mendekati 0. Dari pemodelan kelas lubang dilakukan pengamatan terhadap *loss* yang dihasilkan oleh masing-masing konfigurasi arsitektur yang dipilih. Pemodelan dilakukan dengan iterasi sebanyak 10.000, *learning rate* 0.001 dan model/bobot disimpan dalam setiap seratus iterasi. Model/bobot akan digunakan untuk memeriksa perubahan kinerja dan hasil deteksi kerusakan jalan.

4.1.2.1 Konfigurasi Pemodelan.

Pemodelan dilakukan menggunakan arsitektur *Yolo v3* dengan tiga varian, yaitu *Yolo v3, Yolo v3-tiny,* dan *Yolo v3-SPP*. Masing-masing percobaan menggunakan konfigurasi input $416 \times 416 \times 3$ dan target *feature map* $13 \times 13 \times 18$. Dalam proses pelatihan digunakan *batch* sebanyak 64 dan *subdivision* 16 dalam sekali proses perbaikan bobot dalam setiap iterasi. Secara rinci matriks konfigurasi percobaan kelas lubang ditunjukkan pada Tabel 4.2.

Tabel 4.2 Matriks konfigurasi percobaan kelas lubang.

Arsitektur	Input	Bacth	Subdivision	Maksimum Iterasi
Yolo v3	$416 \times 416 \times 3$	64	16	10.000
Yolo v3 Tiny	$416 \times 416 \times 3$	64	16	10.000
Yolo v3 SPP	$416 \times 416 \times 3$	64	16	10.000

Pemodelan menggunakan skema *transfer learning,* sehingga diperlukan model *pre-trained* untuk membangun model. Karena pemodelan menggunakan tiga arsitektur, maka diperlukan tiga model *pre-trained* yang berbeda, yaitu *yolov3.weights, yolov3-tiny.weights* dan *yolov3-spp.weights.* Masing-masing akan diambil beberapa lapisan untuk dilatih sesuai dengan kelas yang diinginkan. Secara rinci model *pre-trained* yang digunakan pada percobaan kelas lubang ditunjukkan pada Tabel 4.3.

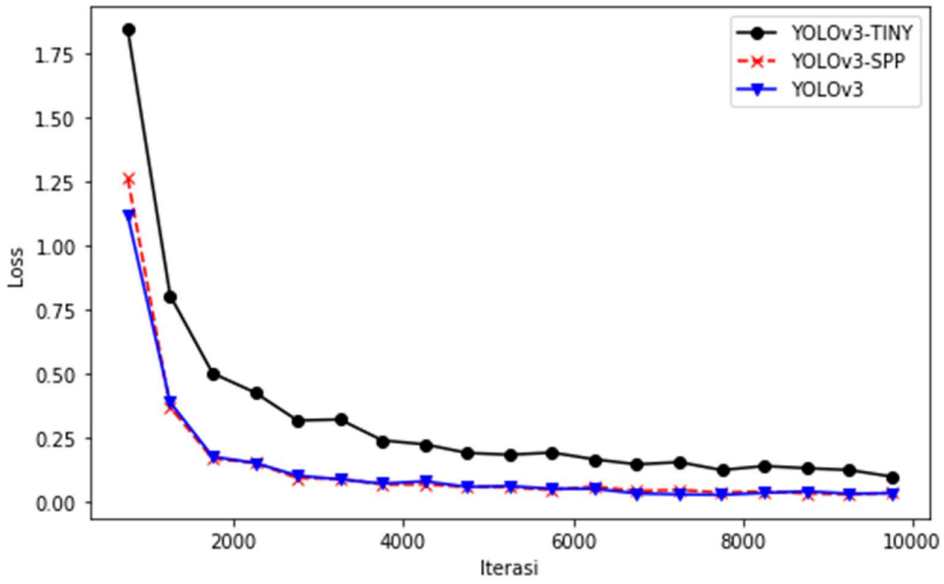
Tabel 4.3 Matriks *pre-trained* percobaan kelas lubang.

Arsitektur	*Pre-trained*	Lapisan yang digunakan
Yolo v3	yolov3.weights	darknet53.conv.74
Yolo v3 Tiny	yolov3-tiny.weights	yolov3-tiny.conv.15
Yolo v3 SPP	yolov3-spp.weights	yolov3-spp.conv.85

4.1.2.2 Proses Pemodelan.

Pada percobaan kelas lubang digunakan tiga varian arsitektur *Yolo v3*, yaitu *Yolo v3*, *Yolo v3 Tiny* dan *Yolo v3 SPP*. *Yolo v3 Tiny* menggunakan model ekstraksi fitur dan arsitektur *Yolo* yang lebih sederhana jika dibandingkan dengan dua varian *Yolo v3* yang lain. Begitu juga untuk *Yolo v3 SPP* menggunakan konfigurasi ekstraksi fitur yang hampir sama dengan *Yolo* v3 namun ditambahkan *Spatial Pyramid Pooling* sehingga menjadikan arsitektur yang digunakan menjadi lebih kompleks. Dari ketiga varian tersebut kita tampilkan perbedaan *loss* yang dihasilkan oleh masing-masing arsitektur sebagaimana di tunjukkan pada Gambar 4.1.

Pada Gambar 4.1 ditunjukkan bahwa *Yolo v3 tiny* memberikan *loss* yang lebih besar dibanding dua arsitektur yang lain. Begitu juga *Yolo v3* memberikan *loss* yang lebih kecil dibandingkan dengan dua arsitektur yang lain. Dimana secara keseluruhan semakin banyak iterasi yang dilakukan maka *loss* yang dihasilkan semakin kecil menuju nilai 0. Pada awal iterasi semua arsitektur memberikan *loss* yang cukup besar, dimulai dengan *loss* diatas 500 kemudian naik turun mencapai nilai diatas 1500. Namun setelah 2000 iterasi pertama penurunan *loss* cenderung stabil dan nilai *loss* juga lebih kecil yaitu berada pada angka dibawah satu.



Gambar 4.1 *Loss Yolo v3, Yolo v3 Tiny*, dan *Yolo v3 SPP*.

Berdasarkan pada Gambar 4.1 dapat kita lihat rata-rata *loss* yang dihasilkan oleh *Yolo v3 Tiny* cenderung diatas dua arsitektur lainnya. Namun jika kita sajikan dari keseluruahn data sesuai 4 maka rata-rata *loss Yolo v3 tiny* lebih kecil dibanding dua arsitektur lainnya. Hal ini terjadi karena perubahan *loss* pada *Yolo v3 Tiny* cenderung lebih kecil tidak seperti pada kedua arsitektur lainnya. Namun setelah iterasi ke-2000 dua arsitektur *Yolo v3* dan *Yolo v3 SPP* memberikan nilai *loss* yang lebih kecil dan lebih stabil dibandingkan *Yolo v3 Tiny*.

Rekapitulasi *loss* percobaan kelas lubang disajikan pada 4, dengan merangkum menjadi beberapa poin yang dapat menjadi perhatian ketika pemodelan dilakukan. Poin yang disajikan meliputi *loss* pada awal dan akhir iterasi, *loss* minimum dan maksimum, rata-rata keseluruhan *loss,* rata-rata *loss* pada 500 dan 2000 iterasi pertama, dan rata-rata *loss* pada 100 iterasi terakhir. Ditampilkan juga rata-rata perubahan *loss* pada 100 iterasi terakhir yang dapat digunakan sebagai pertimbangan dalam melakukan iterasi.

Pada 4 dapat dilihat bahwa *loss Yolo v3* dan *Yolo v3 SPP* cenderung memberikan nilai yang mirip. Sebagai contoh *loss* minimum yang dapat dihasilkan dari dua arsitektur tersebut yaitu 0,01, rata-rata *loss* pada 100 iterasi terakhir yaitu 0,018 dan 0,019 dimana angka ini jauh lebih rendah dari *loss* yang dihasilkan ketika

pemodelan dilakukan menggunakan arsitektur *Yolo v3 tiny*. Begitu juga pada 100 iterasi terakhir ditampilkan juga fluktuasi perubahan *loss* berada pada nilai 0.0049 pada arsitektur *Yolo v3* dan *Yolo v3 SPP* dan angka yang lebih besar dengan nilai 0,0147 dihasilkan oleh *Yolo v3 Tiny*.

Tabel 4.4 Rekapitulasi *Loss* Pemodelan Percobaan kelas lubang.

Keterangan	*Yolo v3 Tiny*		*Yolo v3*		*Yolo v3 SPP*	
	Iterasi Ke-	*Loss*	Iterasi Ke-	*Loss*	Iterasi Ke-	*Loss*
Awal Iterasi	1	703,5779	1	574,4297	1	588,4802
Rata-rata 500 Iterasi Pertama		111,3681		193,0673		228,9331
Rata-rata 2000 Iterasi Pertama		28,6280		48,6864		57,6824
Maksimum	51	780,8470	31	1695,6206	26	1741,6223
Minimum	9.980	0,0455	9.912	0,0100	9.416	0,0108
Rata-rata		5,8856		9,7842		11,5841
Iterasi terakhir	10.000	0,0644	10.000	0,0201	10.000	0,0184
Rata-rata 100 Iterasi Terakhir		0,0735		0,0188		0,0195
Rata-rata Perubahan 100 Iterasi Terakhir		0,0147		0,0049		0,0049

Perubahan *loss* dari iterasi ke iterasi menunjukkan tingkat kesalahan dalam melakukan prediksi data latih terhadap target label yang sudah ditentukan. Semakin kecil *loss* yang dihasilkan maka semakin kecil juga kesalahan yang dilakukan oleh model yang dihasilkan. Penentuan iterasi sebanyak 10.000 dengan memperhatikan rata-rata perubahan iterasi pada 100 iterasi terakhir yaitu 0,0049 pada *Yolo v3* dan *Yolo v3-SPP* dan 0,0147 pada *Yolo v3 Tiny*. Dan juga *loss* terakhir masing-masing sebesar 0,0201, 0,01840, dan 0644.

4.1.3 Evaluasi.

Pada penelitian ini dilakukan evaluasi kinerja model deteksi lubang sebanyak dua evaluasi. Evaluasi pertama yaitu evaluasi deteksi untuk mengukur kinerja model yang dibangun menggunakan *precision, recall,* skor *f-1* dan *mAP*.

Evaluasi deteksi dilakukan terhadap semua model/bobot yang sudah disimpan pada tahap pemodelan kerusakan lubang menggunakan *Yolo*. Dari hasil evaluasi tersebut diambil nilai *mAP* tertinggi sebagai model/bobot yang akan digunakan untuk melakukan deteksi kerusakan lubang. Hasil deteksi kerusakan lubang berupa ukuran *bounding box* dan kelas kerusakan yang akan digunakan untuk menghitung luas hasil deteksi. Luas hasil deteksi ini lah yang digunakan sebagai bahan evaluasi yang kedua yaitu evaluasi luas untuk memperoleh nilai akurasi perhitungan luas. Evaluasi luas dilakukan dengan membandingkan luas *ground truth* dengan luas hasil deteksi.

4.1.3.1 Evaluasi Deteksi.

Pada percobaan kelas lubang dilakukan evaluasi terhadap semua model yang sudah disimpan pada tahap pemodelan. Sesuai dengan penjelasan sebelumnya bahwa pada percobaan kelas lubang dilakukan pemodelan dengan menggunakan tiga arsitektur *Yolo v3* yaitu *Yolo v3, Yolo v3 Tiny* dan *Yolo v3 SPP* dan masing-masing hasil pemodelan dilakukan pengujian dengan enam jumlah data latih yang berbeda.

4.1.3.1.1 Evaluasi Deteksi percobaan kelas lubang menggunakan Arsitektur *Yolo v3*.

Hasil pengujian menggunakan model yang dibangun berdasarkan arsitektur *Yolo v3* dengan enam jumlah data uji yang berbeda yaitu 224, 150, 96, 64, 56 dan 25 menunjukkan nilai akurasi yang berbeda. Pengurangan prosentase data uji terhadap data latih memberikan hasil akurasi yang cenderung lebih baik. Semakin sedikit data uji yang digunakan maka semakin besar akurasi yang dihasilkan. Namun hal ini tidak berlaku mutlak, karena pada tiga pengujian pertama, akurasi terbaik dihasilkan oleh data uji yang paling besar yaitu prosentase data latih dan data uji sebesar 50%:50%. Begitu juga pada pengujian yang terakhir dengan data uji paling sedikit yaitu sebanyak 25 data citra dengan prosentase data latih dan data uji sebesar 90%:10%, hasil akurasi untuk semua penilaian menghasilkan akurasi yang lebih kecil jika dibandingkan dengan dua pengujian sebelumnya.

Meskipun data uji yang digunakan sebagian menggunakan data uji yang sama namun ada perbedaan jumlah yang digunakan, hasil *mAP* tertinggi tidak

terjadi pada iterasi yang sama. Hanya tiga pengujian yang memiliki *mAP* tertinggi pada iterasi yang sama yaitu pada iterasi ke-4700. Adapun semua pengujian, dari pengujian pertama sampai pengujian ke enam nilai *mAP* tertinggi masing-masing terjadi pada iterasi ke-4400, 4300, 4700, 4700, 4500, dan 4700. Secara keseluruhan nilai akurasi yang dihasilkan dari pemodelan menggunakan arsitektur *Yolo v3* memberikan rata-rata nilai *precision, recall,* skor *F1, IoU* dan *mAP* masing-masing sebesar 0,92, 0,75, 0,82, 71,60 dan 83,43. Secara rinci hasil pengujian pada pemodelan kerusakan lubang menggunakan arsitektur *Yolo v3* ditunjukkan pada Tabel 4.5.

Tabel 4.5 *Precision, Recall,* skor *F1, IoU* dan *mAP* percobaan kelas lubang menggunakan arsitektur *Yolo v3*.

Uji	Iterasi	*TP*	*FP*	*FN*	*Precision*	*Recall*	*F1*	*IoU*	*mAP*
1	4400	220	28	76	0,89	0,74	0,81	68,43	81,00
2	4300	147	21	52	0,88	0,74	0,80	68,09	78,77
3	4700	87	8	37	0,92	0,70	0,79	72,29	80,46
4	4700	54	4	19	0,93	0,74	0,82	74,04	86,90
5	4500	52	1	12	0,98	0,81	0,89	75,03	86,92
6	4700	20	2	7	0,91	0,74	0,82	71,69	86,52
Rata-rata					0,92	0,75	0,82	71,60	83,43

4.1.3.1.2 Evaluasi Deteksi percobaan kelas lubang menggunakan Arsitektur *Yolo v3 Tiny*.

Hasil pengujian menggunakan model yang dibangun berdasarkan arsitektur *Yolo v3 Tiny* dengan enam jumlah data uji yang berbeda yaitu 224, 150, 96, 64, 56 dan 25 menunjukkan nilai akurasi yang berbeda. Pengurangan prosentase data uji terhadap data latih memberikan hasil akurasi yang cenderung lebih baik. Semakin sedikit data uji yang digunakan maka semakin besar akurasi yang dihasilkan. Namun hal ini tidak berlaku mutlak, karena pada tiga pengujian pertama, akurasi terbaik dihasilkan oleh data uji yang paling besar yaitu prosentase data latih dan data uji sebesar 50%:50%. Begitu juga pada pengujian yang terakhir dengan data uji paling sedikit yaitu sebanyak 25 data citra dengan prosentase data latih dan data uji sebesar 90%:10%, hasil akurasi untuk semua penilaian kecuali *recall* menghasilkan akurasi yang lebih kecil jika dibandingkan dengan dua pengujian sebelumnya.

Meskipun data uji yang digunakan sebagian menggunakan data uji yang sama namun ada perbedaan jumlah yang digunakan, hasil *mAP* tertinggi tidak terjadi pada iterasi yang sama. Dua pengujian memiliki *mAP* tertinggi pada iterasi yang sama yaitu pada iterasi ke-5900 pada pengujian ke-1 dan ke-2 dan pada iterasi ke-7200 pada pengujian ke-4 dan ke-5. Sedangkan dua pengujian yang lain yaitu pengujian ke-3 dan ke-4 menunjukkan nilai *mAP* tertinggi masing-masing pada iterasi ke-6400 dan iterasi ke-10.000. Secara keseluruhan nilai akurasi yang dihasilkan dari pemodelan menggunakan arsitektur *Yolo v3 Tiny* memberikan rata-rata nilai *precision, recall,* skor *F1, IoU* dan *mAP* masing-masing sebesar 0,94, 0,64, 0,76, 71,07 dan 79,33. Secara rinci hasil pengujian pada pemodelan kerusakan lubang menggunakan arsitektur *Yolo v3* ditunjukkan pada Tabel 4.6.

Tabel 4.6 *Precision, Recall,* skor *F1, IoU* dan *mAP* percobaan kelas lubang menggunakan arsitektur *Yolo v3 Tiny*.

Uji	Iterasi	*TP*	*FP*	*FN*	*Precision*	*Recall*	*F1*	*IoU*	*mAP*
1	5900	182	23	114	0,89	0,61	0,73	67,21	76,16
2	5900	121	13	78	0,90	0,61	0,73	69,03	75,38
3	6400	74	5	50	0,94	0,6	0,73	71,69	75,55
4	7200	48	2	25	0,96	0,66	0,78	73,41	82,89
5	7200	42	1	22	0,98	0,66	0,79	75,14	83,41
6	10000	18	1	9	0,95	0,67	0,78	69,94	82,59
Rata-rata					0,94	0,64	0,76	71,07	79,33

4.1.3.1.3 Evaluasi Deteksi percobaan kelas lubang menggunakan Arsitektur *Yolo v3 SPP*.

Hasil pengujian menggunakan model yang dibangun berdasarkan arsitektur *Yolo v3 SPP* dengan enam jumlah data uji yang berbeda yaitu 224, 150, 96, 64, 56 dan 25 menunjukkan nilai akurasi yang berbeda. Pengurangan prosentase data uji terhadap data latih memberikan hasil akurasi yang cenderung lebih baik. Semakin sedikit data uji yang digunakan maka semakin besar akurasi yang dihasilkan. Namun hal ini tidak berlaku mutlak, karena pada tiga pengujian pertama, akurasi terbaik dihasilkan oleh data uji yang paling besar yaitu prosentase data latih dan data uji sebesar 50%:50% disusul dengan prosentase data latih dan data uji sebesar 70%:30% dan yang terakhir kombinasi data latih dan data uji 60%:40%. Begitu juga pada pengujian yang ke-lima dengan data uji sebanyak 56

data citra dengan prosentase data latih dan data uji sebesar 80%:20%, hasil akurasi untuk semua penilaian menghasilkan akurasi yang lebih kecil jika dibandingkan dengan pengujian sebelumnya.

Meskipun data uji yang digunakan sebagian menggunakan data uji yang sama namun ada perbedaan jumlah yang digunakan, hasil *mAP* tertinggi tidak terjadi pada iterasi yang sama. Terdapat empat pengujian yang memiliki *mAP* tertinggi pada iterasi yang sama yaitu pada iterasi ke-3200. Adapun semua pengujian, dari pengujian pertama sampai pengujian ke enam nilai *mAP* tertinggi masing-masing terjadi pada iterasi ke-3200, 3300, 3200, 3200, 3200, dan 8400. Secara keseluruhan nilai akurasi yang dihasilkan dari pemodelan menggunakan arsitektur *Yolo v3 SPP* memberikan rata-rata nilai *precision, recall,* skor *F1, IoU* dan *mAP* masing-masing sebesar 0,94, 0,82, 0,87, 72,59 dan 88,93. Secara rinci hasil pengujian pada pemodelan kerusakan lubang menggunakan arsitektur *Yolo v3 SPP* ditunjukkan pada Tabel 4.7.

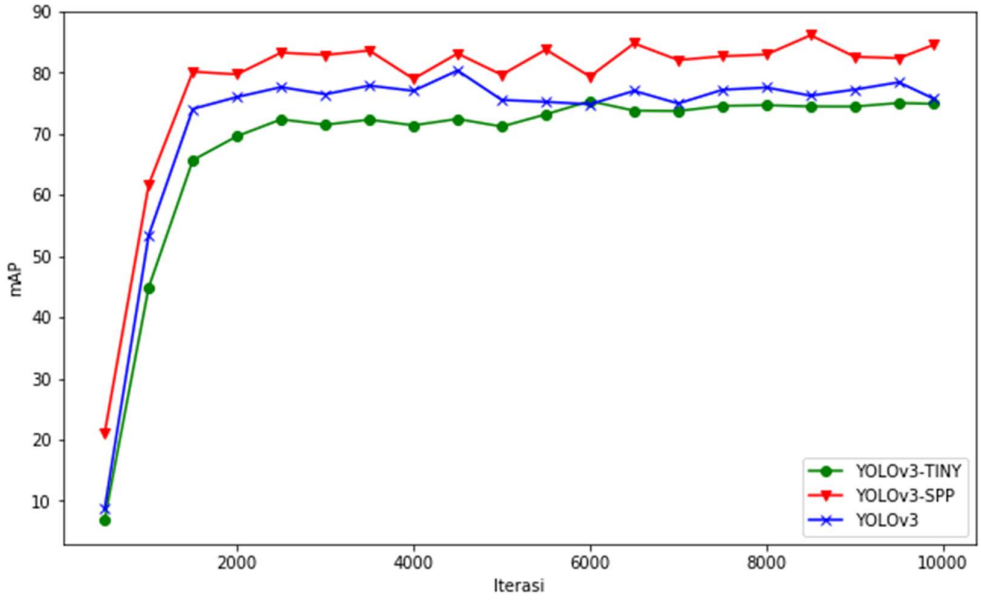
Tabel 4.7 *Precision, Recall,* skor *F1, IoU* dan *mAP* percobaan kelas lubang menggunakan arsitektur *Yolo v3 SPP*.

Uji	Iterasi	*TP*	*FP*	*FN*	*Precision*	*Recall*	*F1*	*IoU*	*mAP*
1	3200	244	21	52	0,92	0,82	0,87	70,12	86,93
2	3300	155	16	44	0,91	0,78	0,84	69,27	84,32
3	3200	101	8	23	0,93	0,81	0,87	70,75	85,75
4	3200	60	3	13	0,95	0,82	0,88	73,32	91,02
5	3200	52	3	12	0,95	0,81	0,87	73,89	90,28
6	8400	23	1	4	0,96	0,85	0,9	78,21	95,27
Rata-rata					0,94	0,82	0,87	72,59	88,93

4.1.3.1.4 Pengaruh arsitektur yang digunakan terhadap *mAP*.

Hasil pengujian tiga arsitektur yang digunakan menunjukkan rata-rata *mAP* paling tinggi dihasilkan oleh arsitektur *Yolo v3 SPP* dan yang paling kecil dihasilkan oleh arsitektur *Yolo v3 Tiny*. Dari awal sampai akhir iterasi *mAP* yang dihasilkan cenderung sama dimana *Yolo v3 SPP* menunjukkan *mAP* paling tinggi, kemudian *Yolo v3* menghasilkan *mAP* berada diantara *Yolo v3 SPP* dan *Yolo v3 Tiny*. Sedangkan *Yolo v3 Tiny* menghasilkan rata-rata *mAP* paling kecil. Pada awal iterasi *mAP* cenderung kecil dan stabil pada iterasi diatas 2000. Ilustrasi perubahan

mAP dari awal iterasi sampai dengan iterasi ke-10.000 ditunjukkan pada Gambar 4.2.



Gambar 4.2 *mAP Yolo v3*, *Yolo v3 Tiny* dan *Yolo v3 SPP*.

Dari hasil pengujian, dirangkum rata-rata *mAP* kedalam tiga kelompok, yaitu:

1. Rata-rata *mAP* pada iterasi ke-2000 sampai dengan iterasi ke-10.000. Iterasi ke-2000 dipilih karena hasil pengujian menunjukkan perubahan nilai *mAP* yang cukup stabil pada sekitar iterasi ke-2000. Rata-rata pada iterasi ini, nilai *mAP Yolo v3 Tiny* memberikan penurunan nilai *mAP* sebesar 3,51% jika dibandingkan dengan nilai *mAP Yolo v3.* Sedangkan penambahan *SPP* pada arsitektur *Yolo v3* memberikan peningkatan *mAP* sebesar 5,75%.
2. Rata-rata *mAP* pada iterasi ke-300 sampai dengan iterasi ke-10.000. Iterasi ke-300 dipilih sebagai awal perhitungan rata-rata karena hasil pengujian pada ke-100 biasanya menunjukkan *mAP* yang cukup kecil dan waktu deteksi yang dibutuhkan jauh lebih lama. Sehingga model pada iterasi ke-100 dan ke-200 tidak mungkin dapat memberikan akurasi dan waktu komputasi yang dapat diterima. Rata-rata pada iterasi ini, nilai *mAP Yolo v3 Tiny* memberikan penurunan nilai *mAP* sebesar 4,02% jika dibandingkan dengan nilai *mAP Yolo*

v3. Sedangkan penambahan *SPP* pada arsitektur *Yolo v3* memberikan peningkatan *mAP* sebesar 6,15%.

3. Rata-rata *mAP* terbaik dari keseluruhan iterasi yang dilakukan. Nilai *mAP* terbaik dipilih karena waktu pemodelan yang dilakukan pada pendekatan *CNN* umumnya membutuhkan waktu yang lama. Sehingga dapat digunakan strategi pemodelan dengan menguji pada tiap beberapa iterasi untuk memperoleh nilai *mAP* yang dirasa sudah cukup baik dalam melakukan deteksi. Rata-rata pada *mAP* terbaik, nilai *mAP Yolo v3 Tiny* memberikan penurunan nilai *mAP* sebesar 4,1% jika dibandingkan dengan nilai *mAP Yolo v3.* Sedangkan penambahan *SPP* pada arsitektur *Yolo v3* memberikan peningkatan *mAP* sebesar 5,5%.

Dari ketiga rekapitulasi rata-rata *mAP* yang diperoleh masing-masing rata-rata *mAP* sebesar 77,60, 72,73 dan 83,90. Secara rinci rata-rata *mAP* dari hasil pengujian terhadap masing-masing arsitektur yang digunakan dapat dilihat pada Tabel 4.8.

Tabel 4.8 Rata-rata *mAP* Percobaan kelas lubang.

No.	Arsitektur	Rata-rata *mAP*		
		Iterasi 2000-10.000	Semua Iterasi	Terbaik
1	*Yolo v3*	76,85	72,02	83,43
2	*Yolo v3 Tiny*	73,34	68,00	79,33
3	*Yolo v3 SPP*	82,60	78,17	88,93
Rata-rata		77,60	72,73	83,90

4.1.3.2 Evaluasi Perhitungan Luas.

Evaluasi luas pada percobaan kelas lubang dilakukan dengan melakukan deteksi pada model yang memberikan nilai *mAP* terbaik yang sudah diperoleh pada evaluasi deteksi.

4.1.3.2.1 Evaluasi Perhitungan Luas percobaan kelas lubang menggunakan Arsitektur *Yolo v3*.

Sesuai dengan hasil pengujian, *mAP* tertinggi yang diperoleh pada evaluasi deteksi menggunakan data uji sebanyak 224, 150, 96, 64, 56 dan 25 pada arsitektur *Yolo v3* masing-masing yaitu pada iterasi ke 4400, 4300, 4700, 4700, 4500 dan 4700. Model pada iterasi tersebut digunakan untuk memperoleh koordinat

bounding box untuk dilakukan perhitungan luas. Dari total perhitungan luas hasil deteksi dibandingkan dengan luas *ground truth* yang dihasilkan pada fase anotasi dan pelabelan sehingga diperoleh akurasi luas untuk masing-masing tipe kerusakan jalan.

Dari keenam pengujian yang dilakukan nilai akurasi luas paling tinggi yaitu sebesar 70,33% diperoleh dari pengujian kedua dan nilai akurasi paling kecil yaitu sebesar 52,96%. Sedangkan rata-rata akurasi perhitungan luas menggunakan arsitekur *Yolo v3* pada percobaan kelas lubang menghasilkan akurasi sebesar 64,45%. Nilai ini cukup kecil jika dibandingkan dengan nilai *mAP* yang dihasilkan pada proses deteksi dimana pada percobaan kelas lubang dan arsitektur *Yolo v3* diperoleh rata-rata *mAP* sebesar 83,43. Meskipun nilai *mAP* yang dihasilkan cukup besar namun untuk ketepatan ukuran masih belum memuaskan. Secara rinci rekapitulasi akurasi perhitungan luas untuk masing-masing data uji ditunjukkan pada Tabel 4.9.

Tabel 4.9 Akurasi perhitungan luas percobaan kelas lubang menggunakan arsitektur *Yolo v3*.

Uji.	Iterasi	Luas *Ground Truth* (m^2)	Luas Deteksi (m^2)	Selisih		Akurasi (%)
				(m^2)	(%)	
1	4400	48,53	29,51	19,03	39,20	60,80
2	4300	30,43	21,40	9,03	29,67	70,33
3	4700	17,98	11,85	6,13	34,09	65,91
4	4700	9,74	6,47	3,27	33,61	66,39
5	4500	8,77	5,83	2,94	33,52	66,48
6	4700	5,57	2,95	2,62	47,04	52,96
Total		121,02	78,00	43,02	35,55	64,45

4.1.3.2.2 Evaluasi Perhitungan Luas percobaan kelas lubang menggunakan Arsitektur *Yolo v3 Tiny*.

Sesuai dengan hasil pengujian, *mAP* tertinggi yang diperoleh pada evaluasi deteksi menggunakan data 224, 150, 96, 64, 56 dan 25 pada arsitektur *Yolo v3* tiny masing-masing yaitu pada iterasi ke 5900, 5900, 6400, 7200, 7200 dan 10000. Model pada iterasi tersebut digunakan untuk memperoleh koordinat *bounding box* untuk dilakukan perhitungan luas. Dari total perhitungan luas hasil deteksi

dibandingkan dengan luas *ground truth* yang dihasilkan pada fase anotasi dan pelabelan sehingga diperoleh akurasi luas untuk masing-masing tipe kerusakan jalan.

Dari keenam pengujian yang dilakukan nilai akurasi luas paling tinggi yaitu sebesar 55,24% diperoleh dari pengujian kedua dan nilai akurasi paling kecil yaitu sebesar 45,79% pada pengujian kelima. Sedangkan rata-rata akurasi perhitungan luas menggunakan arsitekur *Yolo v3 Tiny* pada percobaan kelas lubang sebesar 53,26. Nilai ini cukup kecil jika dibandingkan dengan nilai *mAP* yang dihasilkan pada proses deteksi dimana pada percobaan kelas lubang dan arsitektur *Yolo v3 Tiny* diperoleh rata-rata *mAP* sebesar 79,33. Meskipun nilai *mAP* yang dihasilkan cukup besar namun untuk ketepatan ukuran masih belum memuaskan. Secara rinci rekapitulasi luas untuk masing-masing data uji ditunjukkan secara pada Tabel 4.10.

Tabel 4.10 Akurasi perhitungan luas percobaan kelas lubang menggunakan arsitektur *Yolo v3 Tiny*.

Uji	Iterasi	Luas *Ground Truth* (m^2)	Luas Deteksi (m^2)	Selisih		Akurasi (%)
				(m^2)	(%)	
1	5900	48,53	26,73	21,80	44,93	55,07
2	5900	30,43	16,81	13,62	44,76	55,24
3	6400	17,98	9,38	8,60	47,82	52,18
4	7200	9,74	4,59	5,15	52,87	47,13
5	7200	8,77	4,02	4,76	54,21	45,79
6	10000	5,57	2,93	2,64	47,38	52,62
Total		121,02	64,45	56,56	46,74	53,26

4.1.3.2.3 Evaluasi Perhitungan Luas percobaan kelas lubang menggunakan Arsitektur *Yolo v3 SPP*.

Sesuai dengan hasil pengujian, *mAP* tertinggi yang diperoleh pada evaluasi deteksi menggunakan data 224, 150, 96, 64, 56 dan 25 pada arsitektur *Yolo v3 SPP* masing-masing yaitu pada iterasi ke 3200, 3300, 3200, 3200, 3200 dan 8400. Model pada iterasi tersebut digunakan untuk memperoleh koordinat *bounding box* untuk dilakukan perhitungan luas. Dari total perhitungan luas hasil deteksi dibandingkan

dengan luas *ground truth* yang dihasilkan pada fase anotasi dan pelabelan sehingga diperoleh akurasi luas untuk masing-masing tipe kerusakan jalan.

Dari keenam pengujian yang dilakukan nilai akurasi luas paling tinggi yaitu sebesar 75,11% diperoleh dari pengujian pertama dan nilai akurasi paling kecil yaitu sebesar 63,81% pada pengujian keenam. Sedangkan rata-rata akurasi perhitungan luas menggunakan arsitekur *Yolo v3 SPP* pada percobaan kelas lubang sebesar 72,10. Nilai ini cukup kecil jika dibandingkan dengan nilai *mAP* yang dihasilkan pada proses deteksi dimana pada percobaan kelas lubang dan arsitektur *Yolo v3 SPP* diperoleh rata-rata *mAP* sebesar 88,93. Meskipun nilai *mAP* yang dihasilkan cukup besar namun untuk ketepatan ukuran masih belum memuaskan. Secara rinci rekapitulasi luas untuk masing-masing data uji ditunjukkan pada Tabel 4.11.

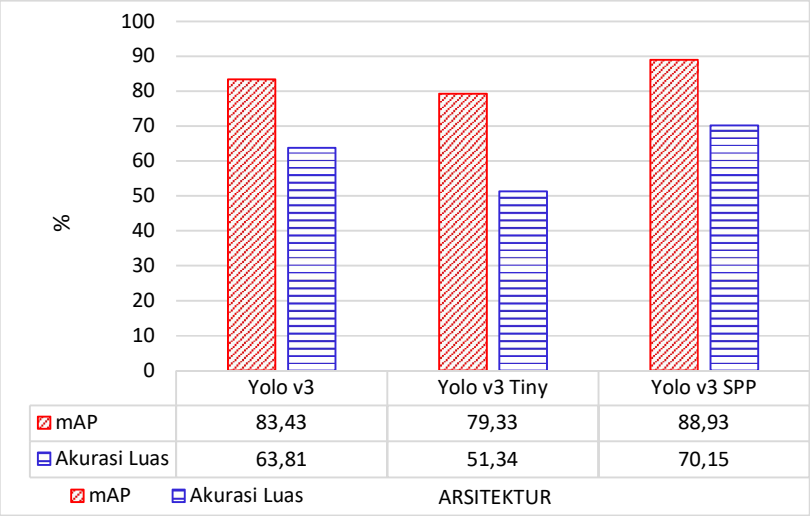
Tabel 4.11 Akurasi perhitungan luas percobaan kelas lubang menggunakan arsitektur *Yolo v3 SPP*.

Uji	Iterasi	Luas *Ground Truth* (m^2)	Luas Deteksi (m^2)	Selisih		Akurasi (%)
				(m^2)	(%)	
1	3200	48,53	36,45	12,08	24,89	75,11
2	3300	30,43	21,47	8,96	29,45	70,55
3	3200	17,98	12,81	5,16	28,71	71,29
4	3200	9,74	6,75	2,98	30,65	69,35
5	3200	8,77	6,21	2,56	29,22	70,78
6	8400	5,57	3,55	2,01	36,19	63,81
Total		121,02	87,25	33,77	27,90	72,10

4.1.3.3 Perbandingan Akurasi deteksi dan Akurasi luas.

Dari ketiga arsitektur yang digunakan sudah diperoleh nilai masing-masing akurasi deteksi maupun akurasi luas. Akurasi terbaik dihasilkan oleh *Yolo v3 SPP* baik dari sisi akurasi deteksi maupun akurasi luas. Dari keseluruhan arsitektur yang digunakan bahwa akurasi luas menunjukkan nilai yang lebih kecil jika dibandingakan dengan akurasi deteksi. Dari perbandingan kedua akurasi ini mengindikasikan bahwa model yang dibangun sudah cukup baik dalam menentukan kategori kelas kerusakan lubang, namun belum dapat memberikan perhitungan yang baik dalam menentukan ukuran *bounding box* pada kerusakan

lubang. Secara lebih detail perbandingan *mAP* dan akurasi luas untuk masing-masing arsitektur ditunjukkan pada Gambar 4.3.



Gambar 4.3 Perbandingan akurasi deteksi (*mAP)* dan akurasi luas percobaan kelas lubang.

4.2 Percobaan Kelas Lubang, Retak dan Lainnya.

Percobaan kedua dilakukan dengan target output tiga kelas kerusakan yaitu lubang, retak dan lainnya. Pada ini pemodelan kerusakan jalan dilakukan enam kali pemodelan dan tiga kali pengujian untuk masing-masing model yang dihasilkan. Adapun skema pemodelan yang dilakukan adalah:

1. Pemodelan dengan skema *fine tuning,* arsitektur *Yolo v3* dan data latih sebanyak 516 data citra.
2. Pemodelan dengan skema *fine tuning,* arsitektur *Yolo v3* dan data latih sebanyak 774 data citra.
3. Pemodelan dengan skema *transfer learning,* arsitektur *Yolo v3* dan data latih sebanyak 516 data citra.
4. Pemodelan dengan skema *transfer learning,* arsitektur *Yolo v3* dan data latih sebanyak 774 data citra.
5. Pemodelan dengan skema *transfer learning,* arsitektur *Yolo v3 SPP* dan data latih sebanyak 516 data citra.
6. Pemodelan dengan skema *transfer learning,* arsitektur *Yolo v3 SPP* dan data latih sebanyak 774 data citra.

Dari masing-masing skema pemodelan diatas akan dilakukan pengujian masing-masing sebanyak tiga kali dengan jumlah data yang berbeda, yaitu:

1. Pengujian dengan data uji sebanyak 258 data citra.
2. Pengujian dengan data uji sebanyak 516 data citra.
3. Pengujian dengan data uji sebanyak 774 data citra.

Kombinasi skema pemodelan dan pengujian diatas dapat dirangkum sesuai dengan Tabel 4.12

Tabel 4.12 Kombinasi skema pemodelan dan pengujian.

No. Model	No. Uji	Skema Pemodelan	Arsitektur	Data Latih	Data Uji	Total
1	1	*Fine Tuning*	*Yolo v3*	516	258	774
	2	*Fine Tuning*	*Yolo v3*	516	516	1032
	3	*Fine Tuning*	*Yolo v3*	516	774	1290
2	4	*Fine Tuning*	*Yolo v3*	774	258	1032
	5	*Fine Tuning*	*Yolo v3*	774	516	1290
	6	*Fine Tuning*	*Yolo v3*	774	774	1548
3	7	*Transfer Learning*	*Yolo v3*	516	258	774
	8	*Transfer Learning*	*Yolo v3*	516	516	1032
	9	*Transfer Learning*	*Yolo v3*	516	774	1290
4	10	*Transfer Learning*	*Yolo v3*	774	258	1032
	11	*Transfer Learning*	*Yolo v3*	774	516	1290
	12	*Transfer Learning*	*Yolo v3*	774	774	1548
5	13	*Transfer Learning*	*Yolo v3 SPP*	516	258	774
	14	*Transfer Learning*	*Yolo v3 SPP*	516	516	1032
	15	*Transfer Learning*	*Yolo v3 SPP*	516	774	1290
6	16	*Transfer Learning*	*Yolo v3 SPP*	774	258	1032
	17	*Transfer Learning*	*Yolo v3 SPP*	774	516	1290
	18	*Transfer Learning*	*Yolo v3 SPP*	774	774	1548

4.2.1 Data Jalan Raya.

Total data jalan raya dengan tiga kategori kerusakan yaitu lubang, retak dan lainnya yang dipilih sebanyak 1548 data dan dibagi menjadi data latih dan data uji. Pada pemodelan pertama, ketiga dan kelima yang diuraikan pada subbab 4.2 menggunakan data latih sebanyak 516 data dan tiga kali pengujian menggunakan data uji masing-masing sebanyak 258, 516 dan 774. Perbandingan prosentase data latih dan data uji dari skema ini berturut-turut 67%:33%, 50%:50% dan 40%:60%.

Sedangkan pada pemodelan kedua, keempat dan keenam yang diuraikan pada subbab 4.2 menggunakan data latih sebanyak 774 data dan tiga kali pengujian

menggunakan data uji masing-masing sebanyak 258, 516 dan 774. Perbandingan prosentase data latih dan data uji dari skema ini berturut-turut 75%:25%, 60%:40% dan 50%:50%. Pembagian data secara rinci juga dapat dilihat pada Tabel 4.12.

4.2.2 Statistik Data Penelitian.

Hasil proses anotasi dan pelabelan yaitu lima variabel yang mendeskripsikan *bounding box* kerusakan jalan. Dari kelima informasi tersebut salah satunya yaitu id_kelas kerusakan. Id_kelas kerusakan akan digunakan untuk mengelompokkan data berdasarkan kerusakan untuk mengetahui jumlah distribusi detail masing-masing kelas kerusakan jalan. Dari keseluruhan data yang disiapkan dilakukan pembagian secara acak untuk menentukan data latih dan data uji. Hasil pembagian data diperoleh sebaran kerusakan sebagaimana ditunjukkan pada Tabel 4.13.

Tabel 4.13 Distribusi Kerusakan Jalan.

No.	Kategori	Jumlah Data Citra	Jumlah Kerusakan			
			Lubang	Retak	Lainnya	Total
1	Data Latih	516	386	415	138	939
2	Data Latih	774	550	637	212	1399
3	Data Uji	258	189	194	56	439
4	Data Uji	516	507	180	76	763
5	Data Uji	774	737	321	111	1169

Secara keseluruhan dapat kita lihat sebaran kerusakan data meningkat seiring jumlah data yang digunakan. Kecuali pada data uji 516 dimana data retak jumlahnya lebih sedikit dibanding pada data uji 258. Hal ini terjadi karena pemilihan dilakukan secara acak sehingga jumlah distribusi kerusakan tidak dapat ditentukan secara pasti. Dari ketiga kategori kerusakan, jumlah kategori kerusakan paling banyak yaitu kerusakan lubang di ikuti kerusakan retak dan yang terakhir adalah kategori lainnya. Masing-masing dengan rata-rata sebesar 47%, 39% dan 13%.

4.2.3 Pemodelan menggunakan *Yolo*.

Seperti halnya pemodelan pada kelas lubang, pemodelan pada tiga kelas lubang, retak dan lainnya juga menghasilkan *loss*. Pengamatan terhadap *loss* yang dihasilkan oleh masing-masing konfigurasi arsitektur yang dipilih dilakukan. Pemodelan dilakukan dengan iterasi sebanyak 10.000, *learning rate* 0.001 dan

model/bobot disimpan dalam setiap seratus iterasi. Model/bobot akan digunakan untuk memeriksa perubahan kinerja dan hasil deteksi kerusakan jalan.

4.2.3.1 Konfigurasi Pemodelan.

Pada kelas lubang, retak dan lainnya dilakukan pemodelan dan pengujian menggunakan konfigurasi input $416 \times 416 \times 3$ yang akan diproses untuk memperoleh target akhir *feature map* dengan dimensi $13 \times 13 \times 24$. Pelatihan dilakukan dengan skema *transfer learning* dan *fine-tuning.* Dalam proses pelatihan digunakan *batch* sebanyak 64 dan *subdivision* 16 dalam sekali proses perbaikan bobot dalam setiap iterasi. Adapun rincian matriks konfigurasi percobaan kelas lubang, retak dan lainnya ditunjukkan pada Tabel 4.14.

Tabel 4.14 Matriks konfigurasi percobaan kelas lubang, retak dan lainnya.

Skema Latih	Input	Bacth	Subdivision	Maksimum Iterasi
Transfer Learning	$416 \times 416 \times 3$	64	16	10.000
Fine-Tuning	$416 \times 416 \times 3$	64	16	10.000

Baik skema *transfer learning* maupun skema *fine-tuning* memerlukan model *pre-trained* untuk dilatih sesuai dengan kelas baru yang diinginkan. Model *pre-trained* yang digunakan adalah *yolov3.weights* dan *yolov3-spp.weights* yang sudah dilatih berdasarkan dataset *Imagenet.* Masing-masing model *pre-trained* akan dilakukan *tuning* pada beberapa layer terakhir. Sehingga perlu dilakukan proses untuk membekukan bobot pada awal layer dan membuang bobot pada akhir layer yang sudah ditentukan. Secara rinci model *pre-trained* dan *layer* dari *pre-trained* yang digunakan ditunjukkan pada Tabel 4.15.

Tabel 4.15 Matriks *pre-trained* percobaan kelas lubang, retak dan lainnya.

Skema Latih	*Pre-trained*	Lapisan yang digunakan
Transfer Learning	yolov3.weights	darknet53.conv.74
	yolov3-spp.weights	yolov3-spp.conv.85
Fine-Tuning	yolov3.weights	yolov3.conv.81

4.2.3.2 Proses Pemodelan.

Sebagaimana disebutkan pada sub bab 4.2 bahwa pemodelan yang dilakukan pada percobaan kelas lubang, retak dan lainnya dilakukan sebanyak enam

kali. Dari enam kali pemodelan akan dilakukan evaluasi terhadap *loss* yang dihasilkan selama proses pemodelan. Evaluasi pemodelan yang akan dilakukan adalah:

1. Pengaruh skema pemodelan *transfer learning* dan *fine tuning* terhadap *loss* yang dihasilkan.
2. Pengaruh data latih terhadap *loss.*
3. Pengaruh arsitektur yang digunakan terhadap *loss.*

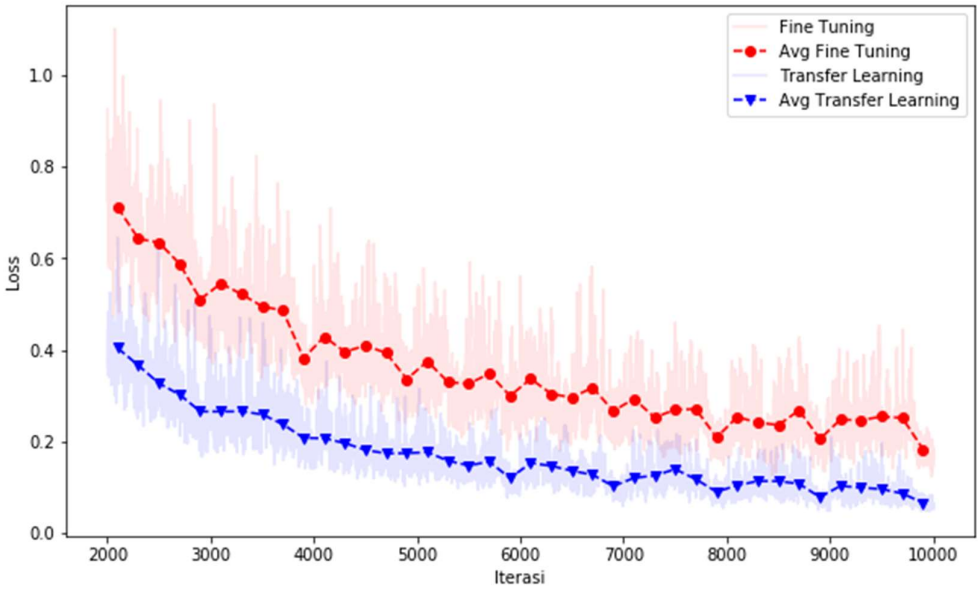
Untuk mempermudah pengamatan selama proses pemodelan rekapitulasi *loss* juga ditampilkan berdasarkan beberapa parameter yaitu:

1. *Loss* pada awal dan akhir iterasi.
2. *Loss* maksimum dan minimum.
3. Rata-rata *loss* secara keseluruhan.
4. Rata-rata *loss* pada 500 dan 2000 iterasi pertama.
5. Rata-rata *loss* pada 100 iterasi terkahir.
6. Rata-rata perubahan *loss* pada 100 iterasi terakhir sebagai pertimbangan dalam menentukan akhir iterasi.

4.2.3.2.1 Pengaruh skema *transfer learning* dan *fine tuning* terhadap *loss.*

Evaluasi ini akan dilakukan terhadap skema pemodelan *transfer learning* dan *fine tuning.* Dimana skema pemodelan memiliki konfigurasi jumlah data latih yang sama dan arsitektur yang sama. Pengamatan akan dilakukan terhadap empat pemodelan yaitu pemodelan pertama (skema *fine tuning,* arsitektur *Yolo v3,* data latih sebanyak 516 data), pemodelan kedua (skema *fine tuning,* arsitektur *Yolo v3,* data latih sebanyak 774 data), pemodelan ketiga (skema *transfer learning,* arsitektur *Yolo v3,* data latih sebanyak 516 data), dan pemodelan keempat (skema *transfer learning,* arsitektur *Yolo v3,* data latih sebanyak 774 data).

Keempat skema pemodelan tersebut dikelompokkan menjadi dua yaitu sesuai dengan skema pemodelan yang digunakan yaitu *transfer learning* dan *fine tuning*. Sehingga dapat diperoleh rata-rata *loss* yang dihasilkan dari awal iterasi sampai dengan akhir iterasi. Dari kedua skema tersebut diperoleh rata-rata *loss fine tuning* lebih besar jika dibandingkan dengan *loss transfer learning* dari awal ietrasi sampai dengan akhir iterasi sebagaimana ditunjukkan pada Gambar 4.4.



Gambar 4.4 *Loss fine tuning* dan *transfer learning.*

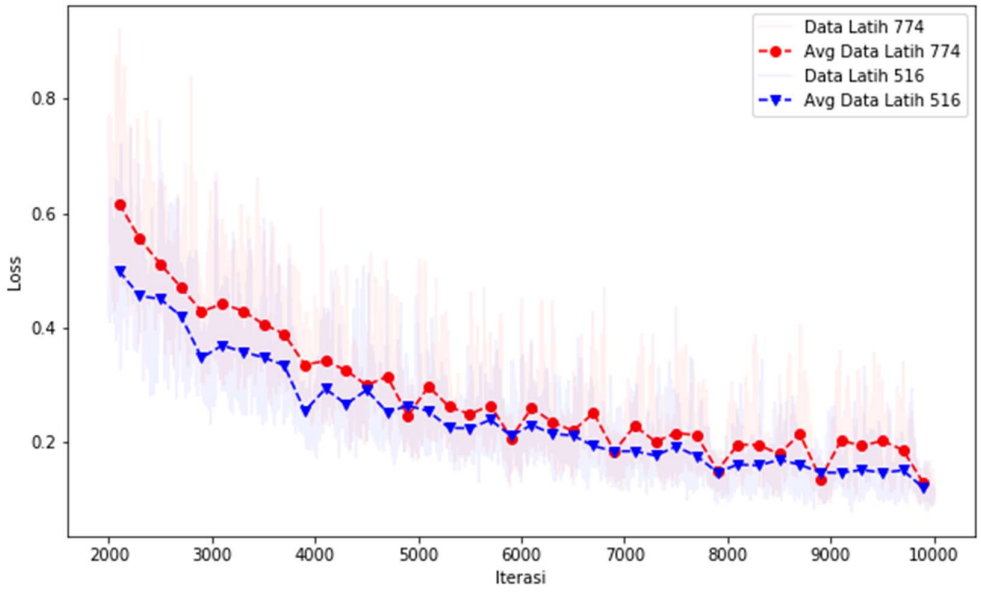
Hal tersebut terjadi karena konfigurasi yang digunakan pada skema *fine tuning* sama dengan konfigurasi *transfer learning*. Sehingga pemodelan menghasilkan *loss* yang cukup tinggi dalam kasus deteksi kerusakan jalan dengan kelas lubang, retak dan lainnya jika *tuning* hanya dilakukan pada *detector.* Sebagaimana disebutkan pada sub bab 2.3.10 bahwa *fine tuning* yang dilakukan yaitu melatih jaringan baru untuk memperoleh bobot *detector* yang baru sedangkan bobot klasifikasi diambil dari semua bobot *pre-trained.* Dari hasil percobaan yang dilakukan, meskipun nilai *loss* yang dihasilkan cukup tinggi, namun proses pemodelan membutuhkan waktu lebih sedikit jika dibandingkan dengan skema *transfer learning*.

4.2.3.2.2 Pengaruh jumlah data latih terhadap *loss.*

Evaluasi ini akan dilakukan terhadap skema pemodelan *transfer learning* dan *fine tuning.* Dimana skema pemodelan memiliki konfigurasi jumlah data latih yang sama dan arsitektur yang sama. Pengamatan akan dilakukan terhadap empat pemodelan yaitu pemodelan pertama (skema *fine tuning,* arsitektur *Yolo v3,* data latih sebanyak 516 data), pemodelan kedua (skema *fine tuning,* arsitektur *Yolo v3,* data latih sebanyak 774 data), pemodelan ketiga (skema *transfer learning,*

arsitektur *Yolo v3,* data latih sebanyak 516 data), dan pemodelan keempat (skema *transfer learning,* arsitektur *Yolo v3,* data latih sebanyak 774 data).

Keempat skema pemodelan tersebut dikelompokkan menjadi dua yaitu sesuai dengan jumlah data latih yang digunakan yaitu data latih sebanyak 516 dan data latih sebanyak 774. Sehingga dapat diperoleh rata-rata *loss* yang dihasilkan dari awal iterasi sampai dengan akhir iterasi. Dari kedua data latih yang digunakan tersebut diperoleh rata-rata *loss* data latih 774 lebih besar jika dibandingkan dengan *loss* data latih 516 dari awal iterasi sampai dengan akhir iterasi sebagaimana ditunjukkan pada Gambar 4.5.



Gambar 4.5 *Loss* data latih 516 dan 774.

4.2.3.2.3 Rekapitulasi *loss* skema *fine tuning* dan *transfer learning* dengan 516 dan 774 data latih.

Sebagaimana disebutkan pada sub bab 4.2.3.2 bahwa ada enam parameter yang digunakan untuk melihat perubahan nilai *loss* selama proses pemodelan. Pada skema *fine tuning* dan *transfer learning* baik menggunakan data latih sebanyak 516 dan 774 semua menghasilkan *loss* yang tinggi pada awal iterasi dengan *loss* diatas 1000 kemudian cenderung mengalami penurunan mendekati 0. Pada 500 iterasi yang pertama memberikan *loss* rata-rata diatas 200 dan cenderung lebih kecil lagi pada iterasi yang lebih besar, sebagaimana diperoleh rata-rata dibawah 100. *Loss*

ini cenderung mengalami penurunan mendekati 0, sehingga pada akhir iterasi dapat diperoleh *loss* yang cukup kecil.

Meskipun kesemua pemodelan menghasilkan *loss* akhir yang cukup kecil namun ada selisih yang cukup besar antara skema *fine tuning* dengan skema *transfer learning*. Pada iterasi terakhir skema *fine tuning* menghasilkan *loss* dua kali lebih besar jika dibandingkan dengan *loss* terakhir pada skema *transfer learning,* begitu juga pada rata-rata *loss* pada 100 iterasi terakhir, bahkan untuk rata-rata perubahan *loss* pada 100 iterasi terakhir mencapai tiga kali lebih besar. Secara rinci rekapitulasi *loss* skema *fine tuning* dan *transfer learning* dengan 516 dan 774 data latih di tunjukkan pada Tabel 4.16.

Tabel 4.16 Rekapitulasi *Loss* pemodelan skema *fine tuning* dan *transfer learning* dengan 516 dan 774 data latih .

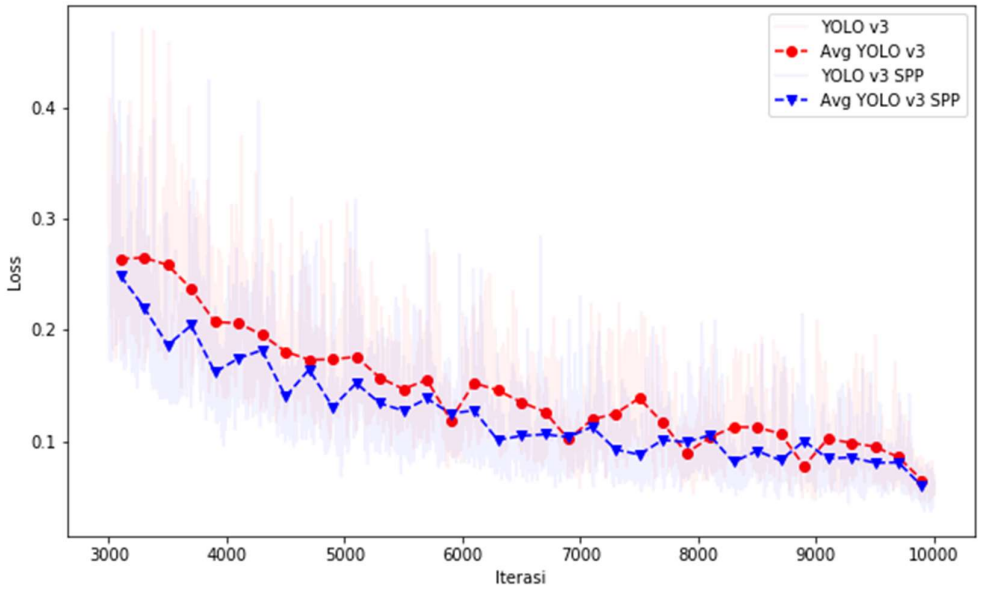
Keterangan	***Loss* (pada Iterasi ke-)**			
	***Fine Tuning* 516**	***Fine Tuning* 774**	***Transfer Learning* 516**	***Transfer Learning* 774**
Awal Iterasi	1.384,45	1.401,00	1.764,87	1.217,13
Rata-rata 500 Iterasi Pertama	262,81	291,88	220,48	252,76
Rata-rata 2000 Iterasi Pertama	66,72	74,02	55,95	64,03
Maksimum	1841,53(61)	1900,54 (51)	1765,96 (3)	1884,41 (51)
Minimum	0,100(9205)	0,099 (9982)	0,031 (9890)	0,041 (8972)
Rata-rata	13,61	15,11	11,31	12,96
Iterasi terakhir	0,14	0,15	0,06	0,06
Rata-rata 100 Iterasi Terakhir	0,17	0,18	0,05	0,07
Rata-rata Perubahan 100 Iterasi Terakhir	0,039	0,035	0,012	0,017

4.2.3.2.4 Pengaruh arsitektur *Yolo v3* dan *Yolo v3 SPP* terhadap *loss*.

Evaluasi ini akan dilakukan terhadap arsitektur yang digunakan dalam mempengaruhi nilai *loss* selama proses pemodelan. Pengamatan akan dilakukan terhadap empat pemodelan yaitu pemodelan ketiga (skema *transfer learning,* arsitektur *Yolo v3,* data latih sebanyak 516 data), pemodelan keempat (skema *transfer learning,* arsitektur *Yolo v3,* data latih sebanyak 774 data), pemodelan

kelima (skema *transfer learning,* arsitektur *Yolo v3 SPP,* data latih sebanyak 516 data), dan pemodelan keenam (skema *transfer learning,* arsitektur *Yolo v3 SPP,* data latih sebanyak 774 data).

Keempat skema pemodelan tersebut dikelompokkan menjadi dua sesuai dengan arsitektur yang digunakan yaitu *Yolo v3* dan *Yolo v3 SPP*. Sehingga dapat diperoleh rata-rata *loss* yang dihasilkan dari awal iterasi sampai dengan akhir iterasi. Dari kedua arsitektur tersebut diperoleh rata-rata *loss Yolo v3* lebih besar jika dibandingkan dengan *loss Yolo v3 SPP* dari awal ietrasi sampai dengan akhir iterasi sebagaimana ditunjukkan pada Gambar 4.6.



Gambar 4.6 *Loss Yolo v3* dan *Yolo v3 SPP.*

Dari dua arsitektur yang digunakan yaitu *Yolo v3* dan *Yolo v3 SPP* sebagian besar *loss* yang dihasilkan dengan arsitektur *Yolo v3 SPP* menunjukkan nilai yang lebih kecil jika dibandingkan dengan *Yolo v3*. Sebagai contoh pada data latih 514 pada awal iterasi *Yolo v3* spp menghasilkan *loss* yang lebih besar namun seiring bertambahnya jumlah iterasi pada akhir iterasi *Yolo v3 SPP* menghasilkan nilai loss yang lebih kecil jika dibandingkan dengan *Yolo v3*. Begitu juga pada data latih 774, *Yolo v3 SPP* menghasilkan rata-rata *loss* yang lebih rendah jika dibandingkan dengan *Yolo v3* dengan data latih yang sama. Secara rinci rekapitulasi

loss skema *transfer learning* dengan data latih sebanyak 516 dan 774 menggunakan arsitektur Yolo v3 *dan* Yolo v3 SPP di tunjukkan pada Tabel 4.17.

Tabel 4.17 Rekapitulasi *Loss* Pemodelan dengan *Yolo v3* dan *Yolo v3 SPP*.

Keterangan	**Loss (pada Iterasi ke-)**			
	***Yolo v3* 518**	***Yolo v3* 774**	***Yolo v3 SPP* 518**	***Yolo v3 SPP* 774**
Awal Iterasi	1764,87	1217,13	1971,44	777,82
Rata-rata 500 Iterasi Pertama	220,48	252,76	253,91	224,21
Rata-rata 2000 Iterasi Pertama	55,95	64,03	64,28	56,87
Maksimum	1765,96 (3)	1884,41 (51)	1972,92 (2)	1369,82 (41)
Minimum	0,031 (9.890)	0,041 (8.972)	0,029 (9.584)	0,031 (9.924)
Rata-rata	11,31	12,96	12,96	11,50
Iterasi terakhir	0,06	0,06	0,05	0,06
Rata-rata 100 Iterasi Terakhir	0,05	0,07	0,07	0,06
Rata-rata Perubahan 100 Iterasi Terakhir	0,01	0,02	0,01	0,02

4.2.4 Evaluasi.

Pada percobaan kelas lubang, retak dan lainnya dilakukan evaluasi kinerja model sebanyak dua evaluasi. Evaluasi pertama yaitu evaluasi deteksi untuk mengukur kinerja model yang dibangun menggunakan *precision, recall,* skor *f-1* dan *mAP*. Evaluasi deteksi dilakukan terhadap semua model/bobot yang sudah disimpan pada tahap pemodelan. Dari hasil evaluasi tersebut diambil nilai *mAP* tertinggi sebagai model/bobot yang akan digunakan untuk melakukan deteksi kerusakan. Hasil deteksi kerusakan berupa ukuran *bounding box* dan kelas kerusakan yang akan digunakan untuk menghitung luas hasil deteksi. Luas hasil deteksi ini lah yang digunakan sebagai bahan evaluasi yang kedua yaitu evaluasi luas untuk memperoleh nilai akurasi perhitungan luas. Evaluasi luas dilakukan dengan membandingkan luas *ground truth* dengan luas hasil deteksi.

4.2.4.1 Evaluasi Deteksi.

Pada percobaan kelas lubang, retak dan lainnya dilakukan evaluasi terhadap semua model yang sudah disimpan pada tahap pemodelan kerusakan jalan.

Sebagaimana disebutkan pada sub bab Percobaan Kelas Lubang, Retak dan Lainnya.4.2 bahwa pemodelan yang dilakukan pada percobaan kelas lubang, retak dan lainnya dilakukan sebanyak enam kali. Dan pengujian dilakukan sebanyak delapan belas kali. Dari delapan belas kali pemodelan akan dilakukan evaluasi akurasi deteksi yang dilakukan terhadap semua model yang sudah disimpan. Evaluasi deteksi dilakukan untuk memperoleh nilai *precision, recall,* skor *F1, IoU* dan *mAP* sehingga dapat diperoleh akurasi masing-masing model yang telah dibangun.

4.2.4.1.1 Evaluasi deteksi percobaan kelas lubang, retak dan lainnya dengan skema *Fine Tuning* dan Arsitektur *Yolo v3*.

Hasil pengujian menggunakan model yang dibangun berdasarkan skema *fine tuning* dan arsitektur *Yolo v3* serta data latih sebanyak 516 dan 774 dengan tiga jumlah data uji yang berbeda yaitu 258, 516 dan 774 menunjukkan nilai akurasi yang berbeda. Penambahan prosentase data uji terhadap data latih cenderung menurunkan nilai akurasi. Semakin banyak data uji maka akurasi menjadi semakin kecil, begitu juga sebaliknya semakin kecil data uji yang digunakan maka akurasi yang dihasilkan semakin besar. Namun hal ini tidak berlaku mutlak, pada pengujian dengan prosentase data latih dan data uji sebesar 40%:60% dengan data latih sebanyak 516 dan data uji sebanyak 774 menghasilkan akurasi yang lebih baik jika dibandingkan dengan prosentase data latih dan data uji sebesar 50%:50% dengan data latih sebanyak 516 dan data uji sebanyak 516. Meskipun demikian selisih akurasi khususnya *mAP* hanya sebesar 0,09.

Sedangkan jika diamati dari penambahan prosentase data latih maka hasil akurasi yang ditunjukkan memberikan nilai akurasi lebih baik. Dengan penambahan data latih sebanyak 258 dari data awal 516 menjadi 774 data latih dan menggunakan data uji yang sama memberikan rata-rata peningkatan akurasi sebesar 3,45. Dari keseluruhan pengujian, hasil *mAP* tertinggi tidak terjadi pada iterasi yang sama. Pada data latih 516 nilai *mAP* tertinggi terjadi pada iterasi ke-9200 pada pengujian dengan data uji 258 dan 774 dan iterasi ke-6600 pada pengujian dengan data latih 516. Dari ketiga pengujian diperoleh rata-rata *precision, recall* dan skor *F1* masing-masing sebesar 0,78, 0,49 dan 0,60. Sedangkan pada data latih 774 nilai *mAP* tertinggi untuk ketiga pengujian terjadi pada iterasi ke 9300, 7200 dan 7100.

Dari ketiga pengujian diperoleh rata-rata *precision, recall* dan skor *F1* masing-masing sebesar 0,75, 0,53 dan 0,62.

Pengujian pada data latih 516 menghasilkan nilai *mAP* paling tinggi untuk masing-masing pengujian sebesar 64,25, 57,34, dan 57,43 sehingga diperoleh rata-rata *mAP* sebesar 59,67. Sedangkan pada data latih 774 menghasilkan *mAP* masing-masing sebesar 66,26, 61,84, dan 61,27 sehingga diperoleh rata-rata *mAP* sebesar 63,12. Secara rinci hasil pengujian pada pemodelan kerusakan lubang, retak dan lainnya menggunakan skema *fine tuning,* data latih sebanyak 516 dan 774, serta arsitektur *Yolo v3* ditunjukkan pada Tabel 4.18 dan Tabel 4.19.

Tabel 4.18 *Precision, Recall,* dan skor *F1* percobaan kelas lubang, retak dan lainnya menggunakan skema *fine tuning* dan arsitektur *Yolo v3*.

Uji	Data Uji	Iterasi ke-	TP	FP	FN	Prec	Recall	F-Measure
Data Latih: 516								
1	258	9200	219	57	220	0,79	0,50	0,61
2	516	6600	364	107	399	0,77	0,48	0,59
3	774	9200	578	173	591	0,77	0,49	0,60
					Rata-rata	0,78	0,49	0,60
Data Latih: 774								
4	258	9300	237	73	202	0,76	0,54	0,63
5	516	7200	433	153	330	0,74	0,57	0,64
6	774	7100	562	192	607	0,75	0,48	0,58
					Rata-rata	0,75	0,53	0,62

Tabel 4.19 *IoU*, Ap dan *mAP* percobaan kelas lubang, retak dan lainnya menggunakan skema *fine tuning* dan arsitektur *Yolo v3*.

Uji	Data Uji	Iterasi	*IoU*	Ap Lubang	Ap Retak	Ap Lainnya	*mAP*
Data Latih: 516							
1	258	9200	58,97	77,14	37,09	78,53	64,25
2	516	6600	57,63	68,63	39,14	64,24	57,34
3	774	9200	57,67	67,16	36,12	69,00	57,43
		Rata-rata	58,09	70,98	37,45	70,59	59,67
Data Latih: 774							
4	258	9300	57,51	78,95	40,08	79,74	66,26
5	516	7200	55,27	72,15	40,29	73,07	61,84
6	774	7100	56,03	67,20	34,77	70,75	61,27
		Rata-rata	56,27	72,77	38,38	74,52	63,12

4.2.4.1.2 Evaluasi deteksi percobaan kelas lubang, retak dan lainnya dengan skema *Transfer Learning*, dan Arsitektur *Yolo v3*.

Hasil pengujian menggunakan model yang dibangun berdasarkan skema *transfer learning* dan arsitektur *Yolo v3* serta data latih sebanyak 516 dan 774 dengan tiga jumlah data uji yang berbeda yaitu 258, 516 dan 774 menunjukkan nilai akurasi yang berbeda. Penambahan prosentase data uji terhadap data latih cenderung menurunkan nilai akurasi. Semakin banyak data uji maka akurasi menjadi semakin kecil, begitu juga sebaliknya semakin kecil data uji yang digunakan maka akurasi yang dihasilkan semakin besar. Rata-rata penurunan akurasi khususnya *mAP* yang dihasilkan dengan menambahkan 258 data pada setiap tahap pengujian yaitu sebesar 1,5.

Sedangkan jika diamati dari penambahan prosentase data latih maka hasil akurasi yang ditunjukkan secara rata-rata tidak memberikan nilai akurasi yang lebih baik. Dari tiga kali pengujian, hanya pada pengujian pertama yang memberikan akurasi lebih baik dengan adanya penambahan jumlah data latih, sedangkan pada pengujian yang kedua memberikan akurasi yang hampir sama, bahkan pada pengujian yang ketiga justru memberikan nilai akurasi yang lebih kecil.

Dari keseluruhan pengujian, hasil *mAP* tertinggi tidak terjadi pada iterasi yang sama. Pada data latih 516 nilai *mAP* tertinggi terjadi pada iterasi ke-6400 pada pengujian dengan data uji 258 dan iterasi ke-8600 pada pengujian dengan data uji 516 dan 774. Dari ketiga pengujian diperoleh rata-rata *precision, recall* dan skor *F1* masing-masing sebesar 0,82, 0,76 dan 0,79. Sedangkan pada data latih 774 nilai *mAP* tertinggi untuk ketiga pengujian terjadi pada iterasi ke 9300 pada pengujian dengan data uji 258, dan iterasi ke-8600 pada pengujian dengan data uji 516 dan 774. Dari ketiga pengujian diperoleh rata-rata *precision, recall* dan skor *F1* masing-masing sebesar 0,82, 0,75 dan 0,78.

Pengujian pada data latih 516 menghasilkan nilai *mAP* paling tinggi untuk masing-masing pengujian sebesar 78,36, 77,35, dan 76,96 sehingga diperoleh rata-rata *mAP* sebesar 77,56. Sedangkan pada data latih 774 menghasilkan *mAP* masing-masing sebesar 80,13, 76,16, dan 75,75 sehingga diperoleh rata-rata *mAP* sebesar

77,35. Rincian *mAP*, *IoU* dan *AP* masing-masing kerusakan ditunjukkan pada Tabel 4.20 dan Tabel 4.21.

Tabel 4.20 *Precision, Recall,* dan skor *F1* percobaan kelas lubang, retak dan lainnya menggunakan skema *transfer learning* dan arsitektur *Yolo v3*.

Uji	Data Uji	Iterasi ke-	*TP*	*FP*	*FN*	*Prec*	*Recall*	*F1*
Data Latih: 516								
1	258	6400	323	69	116	0,82	0,74	0,78
2	516	8600	585	138	178	0,81	0,77	0,79
3	774	8600	888	194	281	0,82	0,76	0,79
					Rata-rata	0,82	0,76	0,79
Data Latih: 774								
4	258	9300	333	57	106	0,85	0,76	0,80
5	516	8600	592	152	171	0,80	0,78	0,79
6	774	8600	882	210	287	0,81	0,72	0,76
					Rata-rata	0,82	0,75	0,78

Tabel 4.21 *IoU*, Ap dan *mAP* percobaan kelas lubang, retak dan lainnya menggunakan skema *transfer learning* dan arsitektur *Yolo v3*.

Uji	Data Uji	Iterasi ke-	*IoU*	Ap Lubang	Ap Retak	Ap Lainnya	*mAP*
Data Latih: 516							
1	258	6400	62,96	83,15	68,58	83,34	78,36
2	516	8600	64,81	82,92	69,84	79,30	77,35
3	774	8600	65,34	84,69	66,77	79,43	76,96
	Rata-rata		64,37	83,59	68,40	80,69	77,56
Data Latih: 774							
4	258	9300	66,99	87,80	67,74	84,85	80,13
5	516	8600	62,91	82,29	67,42	78,77	76,16
6	774	8600	63,85	82,32	65,72	79,20	75,75
	Rata-rata		64,58	84,14	66,96	80,94	77,35

4.2.4.1.3 Evaluasi deteksi percobaan kelas lubang, retak dan lainnya dengan skema skema *Transfer Learning*, Arsitektur *Yolo v3 SPP*.

Hasil pengujian menggunakan model yang dibangun berdasarkan skema *transfer learning* dan arsitektur *Yolo v3 SPP* serta data latih sebanyak 516 dan 774 dengan tiga jumlah data uji yang berbeda yaitu 258, 516 dan 774 menunjukkan nilai akurasi yang berbeda. Penambahan prosentase data uji terhadap data latih

cenderung menurunkan nilai akurasi. Semakin banyak data uji maka akurasi menjadi semakin kecil, begitu juga sebaliknya semakin kecil data uji yang digunakan maka akurasi yang dihasilkan semakin besar. Namun hal ini tidak berlaku mutlak, pada pengujian dengan prosentase data latih dan data uji sebesar 50%:50% dengan data latih sebanyak 774 dan data uji sebanyak 774 menghasilkan akurasi yang lebih baik jika dibandingkan dengan prosentase data latih dan data uji sebesar 60%:40% dengan data latih sebanyak 774 dan data uji sebanyak 516 dengan selisih akurasi khususnya *mAP* sebesar 0,33.

Sedangkan jika diamati dari penambahan prosentase data latih maka hasil akurasi yang ditunjukkan memberikan nilai akurasi lebih baik. Dengan penambahan data latih sebanyak 258 dari data awal 516 menjadi 774 data latih dan menggunakan data uji yang sama memberikan rata-rata peningkatan akurasi sebesar 0,90. Dari ketiga pengujian yang dilakukan tidak semua memberikan akuras yang lebih baik pada data latih yang lebih banyak, yaitu pada pengujian kedua dimana ada penurunan akurasi *mAP* sebesar 0,18.

Meskipun data uji yang digunakan menggunakan data uji yang sama namun jumlahnya berbeda, hasil *mAP* tertinggi tidak terjadi pada iterasi yang sama. Pada data latih 516 nilai *mAP* tertinggi terjadi pada iterasi ke 9000 pada pengujian dengan data uji 258, iterasi ke 7000 pada pengujian dengan data uji 516 dan pada iterasi ke-7100 pada pengujian dengan data uji 774. Dari ketiga pengujian diperoleh rata-rata *precision, recall* dan *F1* masing-masing sebesar 0,81, 0,78 dan 0,80. Sedangkan pada data latih 774 nilai *mAP* tertinggi terjadi pada iterasi ke-8000 pada pengujian dengan data uji 258 dan iterasi ke-8500 pada pengujian dengan data uji 516 dan 774. Dari ketiga pengujian diperoleh rata-rata *precision, recall* dan *F1* masing-masing sebesar 0,80, 0,80 dan 0,80.

Pengujian pada data latih 516 menghasilkan nilai *mAP* paling tinggi untuk masing-masing pengujian sebesar 82,13, 78,94, dan 78,77 sehingga diperoleh rata-rata *mAP* sebesar 79,95. Sedangkan pada data latih 774 menghasilkan *mAP* masing-masing sebesar 84,68, 78,76, dan 79,09 sehingga diperoleh rata-rata *mAP* sebesar 80,84. Secara rinci hasil pengujian pada pemodelan kerusakan lubang, retak dan lainnya menggunakan skema *transfer learning,* data latih sebanyak 516 dan 774, serta arsitektur *Yolo v3 SPP* ditunjukkan pada Tabel 4.22 dan Tabel 4.23.

Tabel 4.22 *Precision, Recall,* dan skor *F1* Percobaan dengan target tiga kelas dan arsitektur *Yolo v3 SPP*.

Uji	Data Uji	Iterasi	*TP*	*FP*	*FN*	*Prec*	*Recall*	*F1*
Data Latih: 516								
1	258	9000	343	63	96	0,84	0,78	0,81
2	516	7000	590	147	173	0,80	0,77	0,79
3	774	7100	924	242	245	0,79	0,79	0,79
					Rata-rata	0,81	0,78	0,80
Data Latih: 774								
4	258	8000	362	81	77	0,82	0,82	362
5	516	8500	608	161	155	0,80	0,79	608
6	774	8500	922	239	247	0,79	0,79	922
					Rata-rata	0,80	0,80	0,80

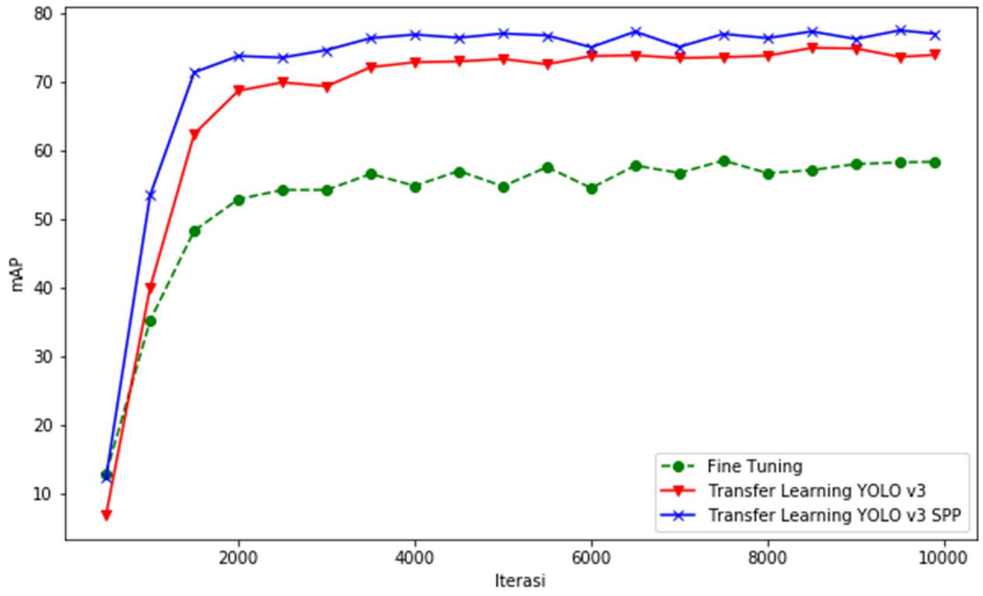
Tabel 4.23 *IoU*, *Ap* dan *mAP* percobaan dengan target tiga kelas dan arsitektur *Yolo v3 SPP*.

Uji	Data Uji	Iterasi	*IoU*	Ap Lubang	Ap Retak	Ap Lainnya	*mAP*
Data Latih: 516							
1	258	9000	66,82	88,15	69,60	88,63	82,13
2	516	7000	64,12	82,58	67,90	79,18	78,94
3	774	7100	62,18	85,12	67,95	78,19	78,77
		Rata-rata	64,37	85,28	68,48	82,00	79,95
Data Latih: 774							
4	258	8000	63,72	90,30	74,71	89,02	84,68
5	516	8500	63,14	85,40	72,52	78,36	78,76
6	774	8500	62,99	84,90	68,54	83,84	79,09
		Rata-rata	63,28	86,87	71,92	83,74	80,84

4.2.4.1.4 Pengaruh arsitektur yang digunakan terhadap *mAP*.

Hasil pengujian pada seluruh skema pemodelan menunjukkan bahwa pada awal iterasi *mAP* cenderung kecil dan stabil pada iterasi diatas 2000. Hasil pengujian menunjukkan nilai *mAP* rata-rata pada iterasi ke 2000 sampai dengan akhir iterasi untuk semua percobaan masing-masing sebesar 56,39 untuk skema *fine tuning* dan arsitektur *Yolo v3*, 72,86 untuk skema *transfer learning* dan arsitektur *Yolo v3* dan 76,17 untuk arsitektur skema *transfer learning* dan arsitektur *Yolo v3 SPP*.

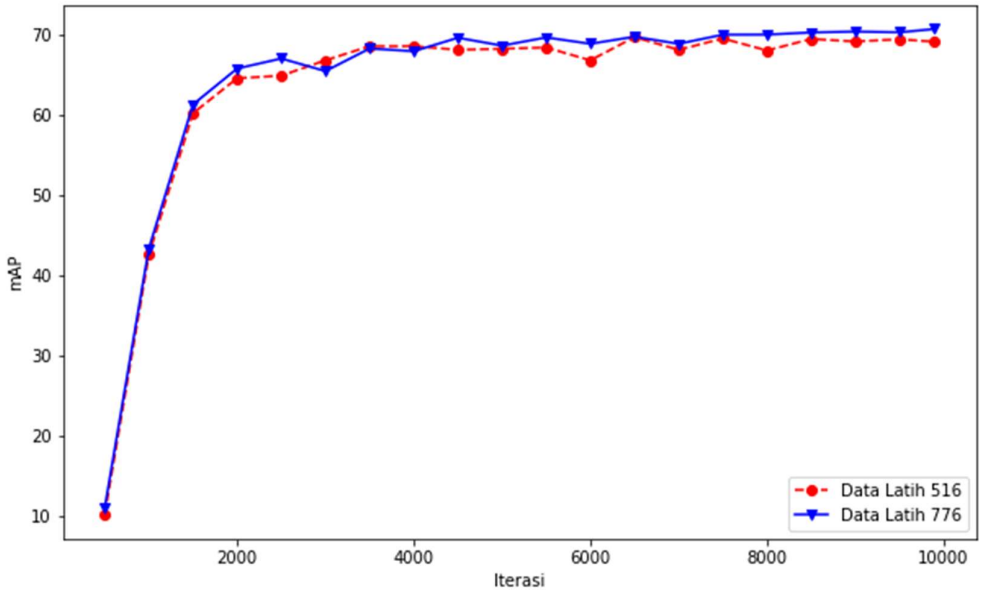
Dari dua skema training yang berbeda yaitu menggunakan *fine tuning* dan *transfer learning* menghasilkan *mAP* yang lebih baik pada skema *transfer learning*. Sedangkan pada skema *transfer learning* dengan arsitektur *Yolo v3* dan *Yolo v3 SPP* menunjukkan hasil bahwa arsitektur *yolo v3 SPP* memberikan hasil *mAP* yang lebih baik. Secara keseluruhan *mAP* rata-rata dari pengujian dengan tiga data uji yang berbeda-beda terhadap model yang dibangun menggunakan berbagai skema pemodelan dan arsitektur yang berbeda ditunjukkan pada Gambar 4.7.



Gambar 4.7 Perbandingan *mAP Fine Tuning, Transfer Learning* dengan *Yolo v3* dan *Transfer Learning* dengan *Yolo v3 SPP*.

4.2.4.1.5 Pengaruh jumlah data latih terhadap *mAP*.

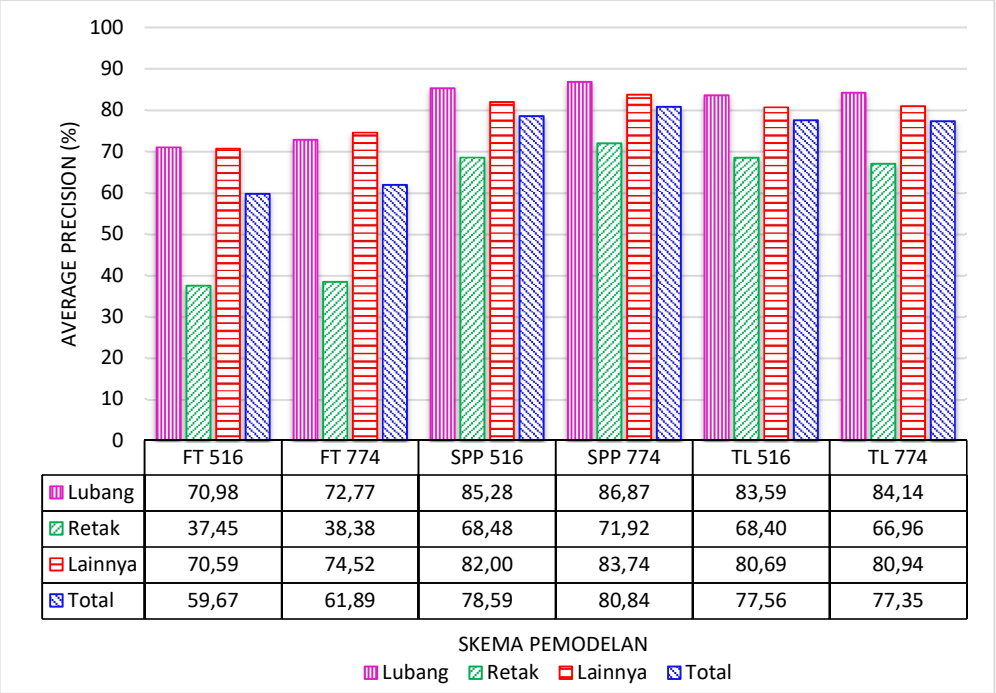
Dari hasil percobaan dapat ditampilkan rata-rata *mAP* dari ketiga skema dan arsitektur pemodelan berdasarkan data latih yang digunakan. Hasil rata-rata *mAP* menunjukkan bahwa jumlah data latih yang semakin besar memberikan *mAP* yang lebih baik jika dibandingkan dengan data latih yang lebih sedikit sebagaimana ditunjukkan pada Gambar 4.8.



Gambar 4.8 Perbandingan *mAP* berdasarkan data latih yang digunakan pada percobaan kelas lubang, retak dan lainnya.

4.2.4.1.6 Sebaran masing-masing *AP*.

Dari hasil percobaan diperoleh perbandingan *AP* masing-masing tipe kerusakan. Dari tiga tipe kerusakan yang diklasifikasikan, kerusakan retak memberikan nilai *AP* yang cukup kecil jika dibandingkan dengan dua jenis kerusakan yang lain. Sebagai contoh pada skema pemodelan *transfer learning* dan arsitektur *Yolo v3 SPP* dengan data 516 dapat kita lihat bahwa *AP* lubang sebesar 85,28, *AP* retak 68,48, dan *AP* lainnya 82,0 sehingga diperoleh *AP* Total (*mAP*) sebesar 79,95. Dari semua skenario pelatihan dan pengujian memberikan hasil yang sama dimana kategori retak memberikan *AP* yang cukup kecil sehingga berkontribusi terhadap penurunan nilai *AP* total (*mAP*). Secara rinci perbandingan nilai *AP* untuk masing-masing keruskan ditunjukkan pada Gambar 4.9.



	FT 516	FT 774	SPP 516	SPP 774	TL 516	TL 774
Lubang	70,98	72,77	85,28	86,87	83,59	84,14
Retak	37,45	38,38	68,48	71,92	68,40	66,96
Lainnya	70,59	74,52	82,00	83,74	80,69	80,94
Total	59,67	61,89	78,59	80,84	77,56	77,35

Gambar 4.9 Perbandingan *AP* masing-masing tipe kerusakan jalan.

4.2.4.2 Evaluasi Perhitungan Luas.

Evaluasi luas pada percobaan kelas lubang, retak dan lainnya dilakukan dengan melakukan deteksi pada model yang memberikan nilai *mAP* terbaik yang sudah diperoleh pada evaluasi deteksi

4.2.4.2.1 Evaluasi perhitungan luas percobaan kelas lubang, retak dan lainnya dengan skema *Fine Tuning* dan Arsitektur *Yolo v3*.

Sesuai dengan hasil pengujian, *mAP* tertinggi yang diperoleh pada evaluasi deteksi menggunakan data latih sebanyak 516 dan 774 dan data uji untuk masing-masing pemodelan sebanyak 228, 516 dan 774. Hasil *mAP* tertinggi diperoleh masing-masing pada iterasi ke-9200, 6600, dan 9200 pada data latih 516 dan pada iterasi ke-9300, 7200, dan 7100. Model pada iterasi tersebut digunakan untuk memperoleh koordinat *bounding box* untuk dilakukan perhitungan luas. Dari total perhitungan luas hasil deteksi dibandingkan dengan luas *ground truth* yang dihasilkan pada fase anotasi dan pelabelan sehingga diperoleh akurasi luas untuk keseluruhan kerusakan jalan.

Dari keenam pengujian yang dilakukan nilai akurasi luas paling tinggi yaitu sebesar 74,77% diperoleh dari pengujian dengan 516 data uji dan 774 data

latih dan nilai akurasi paling kecil yaitu sebesar 59,57% diperoleh dari pengujian dengan 774 data uji dan 516 data latih. Sedangkan rata-rata akurasi perhitungan luas menggunakan skema *fine tuning* dan arsitekur *Yolo v3* pada percobaan kelas lubang, retak dan lainnya menghasilkan akurasi sebesar 65,13%. Nilai ini lebih besar jika dibandingkan dengan nilai *mAP* yang dihasilkan pada proses deteksi dimana diperoleh rata-rata *mAP* sebesar 61,40%. Secara rinci rekapitulasi akurasi perhitungan luas untuk masing-masing data uji ditunjukkan pada Tabel 4.24.

Tabel 4.24 Akurasi perhitungan luas percobaan kelas lubang, retak dan lainnya skema *Fine Tuning* dan Arsitektur *Yolo v3*.

Uji	Data Uji	Iterasi ke-	Luas Ground Truth (m^2)	Luas Deteksi (m^2)	Selisih		Akurasi (%)
					(m^2)	(%)	
Data Latih: 516							
1	258	9200	142,52	92,64	49,88	35,00	65,00
2	516	6600	259,81	182,78	77,03	29,65	70,35
3	774	9200	384,62	229,11	155,50	40,43	59,57
		Total	786,95	504,53	282,42	35,89	64,11
Data Latih: 774							
4	258	9300	142,52	92,33	50,19	35,21	64,79
5	516	7200	259,81	194,27	65,54	25,23	74,77
6	774	7100	384,62	233,97	150,65	39,17	60,83
		Total	786,95	520,58	266,37	33,85	66,15

4.2.4.2.2 Evaluasi perhitungan luas percobaan kelas lubang, retak dan lainnya dengan skema *Transfer Learning*, dan Arsitektur Yolo v3.

Sesuai dengan hasil pengujian, *mAP* tertinggi yang diperoleh pada evaluasi deteksi menggunakan data latih sebanyak 516 dan 774 dan data uji untuk masing-masing pemodelan sebanyak 228, 516 dan 774. Hasil *mAP* tertinggi diperoleh masing-masing pada iterasi ke-6400, 8600, dan 8600 pada data latih 516 dan pada iterasi ke-9300, 8600, dan 8600. Model pada iterasi tersebut digunakan untuk memperoleh koordinat *bounding box* untuk dilakukan perhitungan luas. Dari total perhitungan luas hasil deteksi dibandingkan dengan luas *ground truth* yang dihasilkan pada fase anotasi dan pelabelan sehingga diperoleh akurasi luas untuk keseluruhan kerusakan jalan.

Dari keenam pengujian yang dilakukan nilai akurasi luas paling tinggi yaitu sebesar 94,85% diperoleh dari pengujian dengan 258 data uji dan 774 data latih dan nilai akurasi paling kecil yaitu sebesar 80,48% diperoleh dari pengujian dengan 516 data uji dan 516 data latih. Sedangkan rata-rata akurasi perhitungan luas menggunakan skema *transfer learning* dan arsitekur *Yolo v3* pada percobaan kelas lubang, retak dan lainnya menghasilkan akurasi sebesar 88,69%. Nilai ini lebih besar jika dibandingkan dengan nilai *mAP* yang dihasilkan pada proses deteksi dimana diperoleh rata-rata *mAP* sebesar 77,46%. Secara rinci rekapitulasi luas untuk masing-masing data latih dan data uji ditunjukkan pada Tabel 4.25.

Tabel 4.25 Akurasi perhitungan luas percobaan kelas lubang, retak dan lainnya skema *Transfer Learning* dan Arsitektur *Yolo v3*.

Uji	Data Uji	Iterasi ke-	Luas Ground Truth (m^2)	Luas Deteksi (m^2)	Selisih		Akurasi (%)
					(m^2)	(%)	
Data Latih: 516							
1	258	6400	142,52	128,82	13,70	9,61	90,39
2	516	8600	259,81	209,09	50,72	19,52	80,48
3	774	8600	384,62	314,11	70,51	18,33	81,67
		Total	786,95	652,02	134,93	17,15	82,85
Data Latih: 774							
4	258	9300	142,52	138,82	7,35	5,15	94,85
5	516	8600	259,81	246,36	13,45	5,18	94,82
6	774	8600	384,62	362,38	22,24	5,78	94,22
		Total	786,95	747,56	43,04	5,47	94,53

4.2.4.2.3 Evaluasi perhitungan luas percobaan kelas lubang, retak dan lainnya dengan skema *Transfer Learning*, dan Arsitektur *Yolo v3 SPP*.

Sesuai dengan hasil pengujian, *mAP* tertinggi yang diperoleh pada evaluasi deteksi menggunakan data latih sebanyak 516 dan 774 dan data uji untuk masing-masing pemodelan sebanyak 228, 516 dan 774. Hasil *mAP* tertinggi diperoleh masing-masing pada iterasi ke-9000, 7000, dan 7100 pada data latih 516 dan pada iterasi ke-8000, 8500, dan 8500. Model pada iterasi tersebut digunakan untuk memperoleh koordinat *bounding box* untuk dilakukan perhitungan luas. Dari total perhitungan luas hasil deteksi dibandingkan dengan luas *ground truth* yang

dihasilkan pada fase anotasi dan pelabelan sehingga diperoleh akurasi luas untuk keseluruhan kerusakan jalan.

Dari keenam pengujian yang dilakukan nilai akurasi luas paling tinggi yaitu sebesar 96,74% diperoleh dari pengujian dengan 774 data uji dan 774 data latih dan nilai akurasi paling kecil yaitu sebesar 84,85% diperoleh dari pengujian dengan 516 data uji dan 516 data latih. Sedangkan rata-rata akurasi perhitungan luas menggunakan skema *transfer learning* dan arsitekur *Yolo v3 SPP* pada percobaan kelas lubang, retak dan lainnya menghasilkan akurasi sebesar 91,13%. Nilai ini lebih besar jika dibandingkan dengan nilai *mAP* yang dihasilkan pada proses deteksi dimana diperoleh rata-rata *mAP* sebesar 80,40%. Secara rinci rekapitulasi luas untuk masing-masing data latih dan data uji ditunjukkan pada Tabel 4.26.

Tabel 4.26 Akurasi perhitungan luas percobaan kelas lubang, retak dan lainnya skema *Transfer Learning* dan Arsitektur *Yolo v3 SPP*.

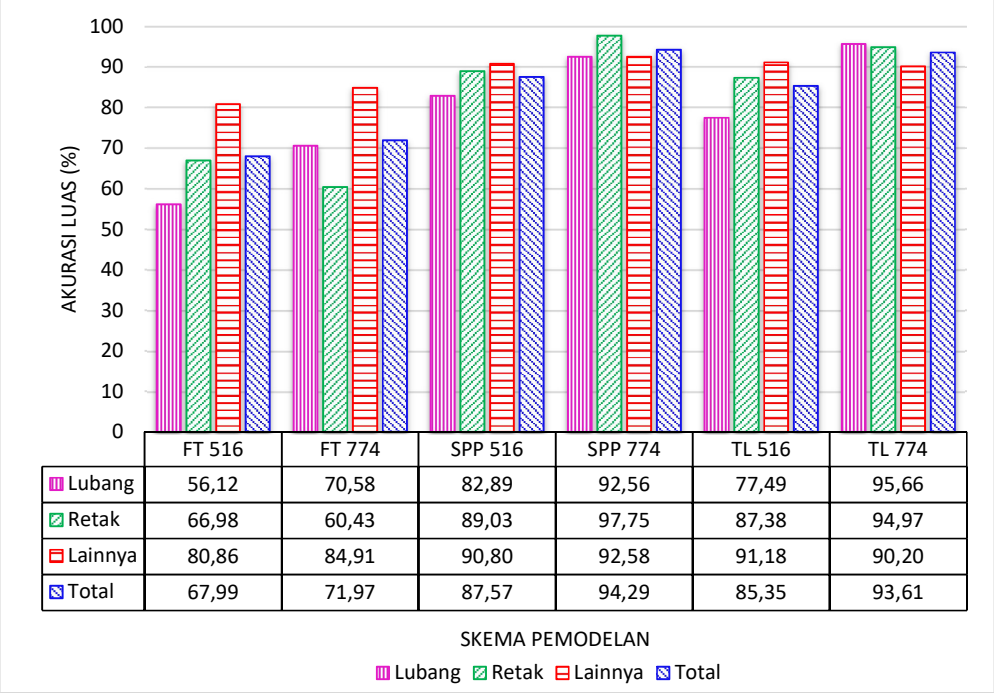
Uji	Data Uji	Iterasi ke-	Luas Ground Truth (m^2)	Luas Deteksi (m^2)	Selisih		Akurasi (%)
					(m^2)	(%)	
Data Latih: 516							
1	258	9000	142,52	139,45	13,27	9,31	90,69
2	516	7000	259,81	267,12	39,36	15,15	84,85
3	774	7100	384,62	376,73	52,79	13,72	86,28
	Total		786,95	783,30	105,42	13,40	86,60
Data Latih: 774							
4	258	8000	142,52	139,12	10,25	7,19	92,81
5	516	8500	259,81	249,03	11,41	4,39	95,61
6	774	8500	384,62	380,12	12,54	3,26	96,74
	Total		786,95	768,27	34,19	4,35	95,65

4.2.4.2.4 Sebaran masing-masing akurasi luas.

Dari hasil percobaan dapat diperoleh perbandingan akurasi luas masing-masing tipe kerusakan. Dari tiga tipe kerusakan yang diklasifikasikan, rata-rata kerusakan lubang memberikan nilai akurasi luas yang lebih kecil jika dibandingkan dengan dua jenis kerusakan yang lain. Pada skema pemodelan pertama (skema *fine tuning*, arsitektur *Yolo v3,* data latih 516) masing-masing akurasi luas lubang, retak, lainnya dan total sebesar 56,12, 66,98, 80,86 dan 67,99. Pada skema pemodelan

kedua (skema *fine tuning*, arsitektur *Yolo v3,* data latih 774) masing-masing akurasi luas lubang, retak, lainnya dan total sebesar 70,58, 60,43, 84,91 dan 71,97. Pada skema pemodelan ketiga (skema *transfer learning*, arsitektur *Yolo v3,* data latih 516) masing-masing akurasi luas lubang, retak, lainnya dan total sebesar 77,59, 87,38, 91,18 dan 85,35. Pada skema pemodelan keempat (skema *transfer learning*, arsitektur *Yolo v3,* data latih 774) masing-masing akurasi luas lubang, retak, lainnya dan total sebesar 95,66, 94,97, 90,20 dan 93,61. Pada skema pemodelan kelima (skema *transfer learning*, arsitektur *Yolo v3 SPP,* data latih 516) masing-masing akurasi luas lubang, retak, lainnya dan total sebesar 82,89, 89,03, 90,80 dan 87,57. Dan yang terakhir pada skema pemodelan keenam (skema *transfer learning*, arsitektur *Yolo v3 SPP,* data latih 774) masing-masing akurasi luas lubang, retak, lainnya dan total sebesar 92,56, 97,75, 92,58 dan 94,29.

Dari keenam skema pemodelan tersebut, empat skema pemodelan memberikan hasil akurasi luas lubang paling kecil diantara dua kategori kerusakan jalan lainnya. Namun demikian hasil akurasi luas yang dihasilkan menunjukkan nilai yang cukup memuaskan. Secara detail dapat diamati dengan lebih mudah pada Gambar 4.10.

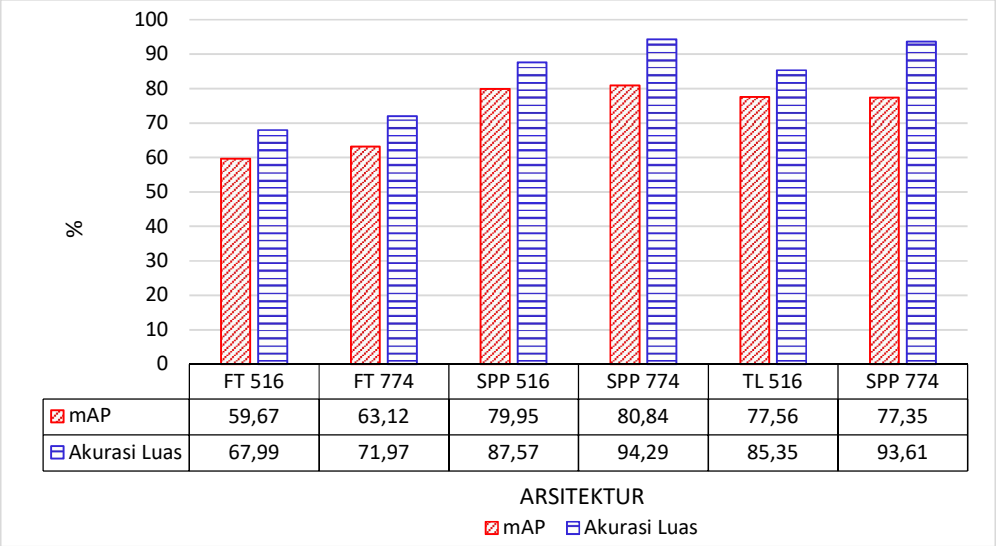


	FT 516	FT 774	SPP 516	SPP 774	TL 516	TL 774
Lubang	56,12	70,58	82,89	92,56	77,49	95,66
Retak	66,98	60,43	89,03	97,75	87,38	94,97
Lainnya	80,86	84,91	90,80	92,58	91,18	90,20
Total	67,99	71,97	87,57	94,29	85,35	93,61

Gambar 4.10 Sebaran akurasi luas percobaan kelas lubang, retak dan lainnya.

4.2.4.2.5 Perbandingan Akurasi deteksi dan Akurasi luas.

Akurasi luas menunjukkan nilai yang senada dengan akurasi deteksi yang ditunjukkan dengan *mean average precision*. Semakin besar nilai *mAP* maka nilai akurasi luas yang dihasilkan juga semakin tinggi. Dari sisi arsitektur yang digunakan, *Yolo v3 SPP* memberikan akurasi deteksi maupun akurasi luas terrbaik diantara arsitektur yang lain. Dari sisi penggunaan jumlah data latih, data latih yang lebih besar menghasilkan peningkatan akurasi jika dibandingkan dengan data latih yang lebih kecil. Secara lebih rinci perbandingan rata-rata *mAP* dan akurasi luas dari keenam skema pelatihan dan arsitektur yang digunakan ditunjukkan pada Gambar 4.11.



	FT 516	FT 774	SPP 516	SPP 774	TL 516	SPP 774
mAP	59,67	63,12	79,95	80,84	77,56	77,35
Akurasi Luas	67,99	71,97	87,57	94,29	85,35	93,61

Gambar 4.11 Perbandingan *mAP* dan Akurasi Luas Percobaan kelas lubang, retak dan lainnya.

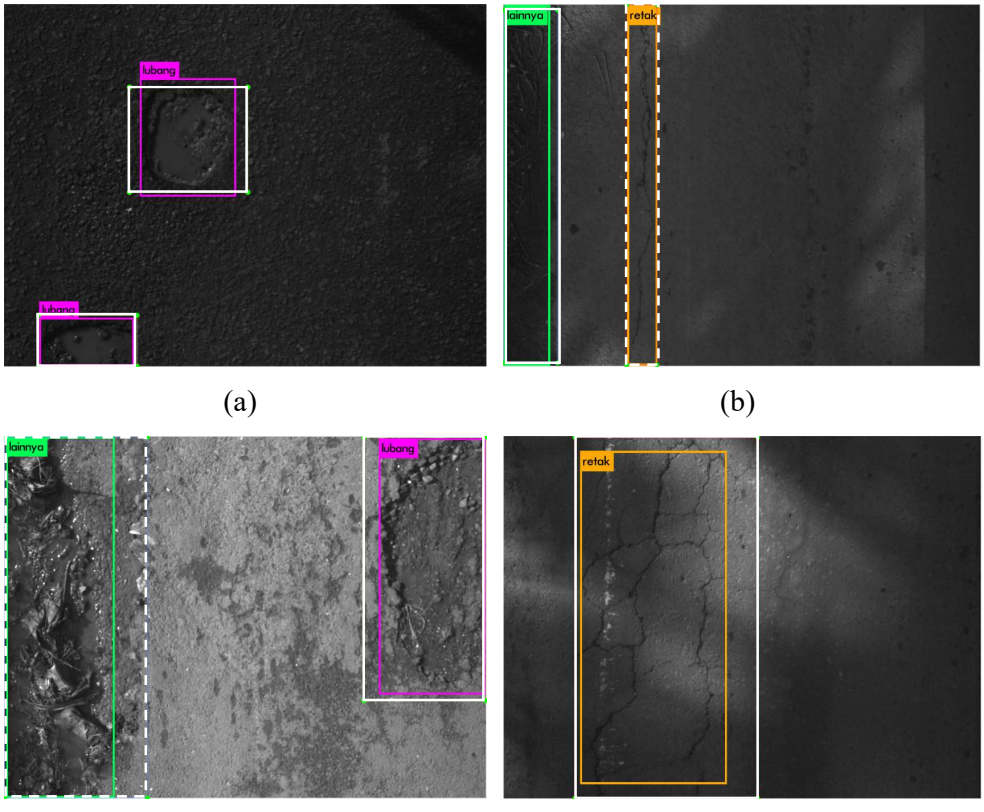
4.3 Hasil deteksi secara visual.

Dari hasil evaluasi deteksi yang sudah dibahas pada sub bab 4.1.3.1 dan 4.2.4.1, dipilih hasil deteksi dari iterasi yang menghasilkan *mAP* terbaik untuk dilakukan pengamatan secara visual. Pengamatan dipilih dari beberapa data jalan raya hasil deteksi yang menunjukkan deteksi sukses *(true positive)* dan deteksi gagal *(false negative)*. Hasil deteksi ditampilkan dalam bentuk *bounding box* dilengkapi dengan kelas kerusakan pada gambar. Selain itu hasil deteksi juga dapat ditampilkan berupa teks dalam format *.txt* yang berisi informasi kelas kerusakan dan koordinat *bounding box* seperti yang dihasilkan pada proses anotasi dan

pelabelan. Dan untuk memeriksa kesuksesan deteksi pengamatan dilakukan dengan membandingkan *bounding box ground truth* dengan *bounding box* hasil deteksi. Semua gambar yang terdapat pada sub bab ini akan dilengkapi dengan *bounding box ground truth* dan *bounding box* deteksi untuk obyek yang sukses dikenali. *Bounding box* hasil deteksi ditunjukkan dengan kotak yang dilengkapi dengan nama kelas kerusakan jalan, sedangkan *bounding box ground truth* ditunjukkan dengan kotak yang biasanya berimpit dengan *bounding box* hasil deteksi dengan warna lain.

4.3.1 Deteksi Sukses.

Hasil deteksi sukses diperoleh pada kerusakan yang jika diamati secara visual mudah diinterpretasikan, misalnya memiliki tekstur yang berbeda secara nyata jika dibandingkan dengan tekstur disekelilingnya yang tidak rusak, dan memiliki tepi yang jelas meskipun dengan pencahayaan yang kurang. Deteksi sukses sering ditemui pada kelas kerusakan lubang dan lainnya. Sebagai contoh deteksi sukses ditunjukkan pada Gambar 4.12.



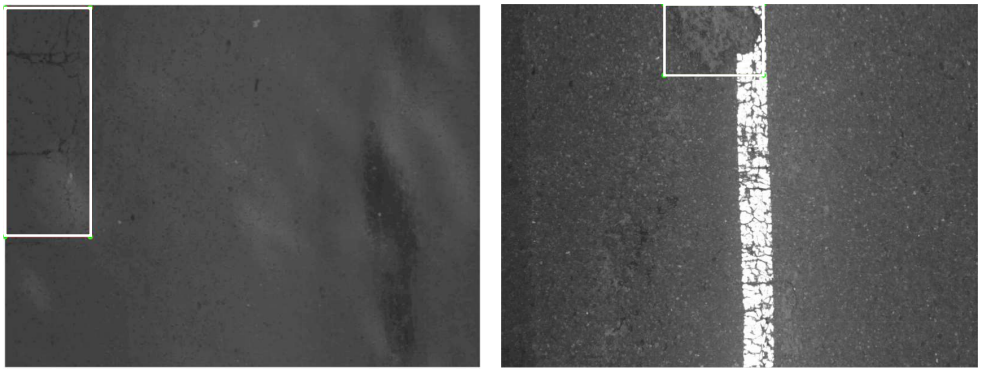
(a) (b)

(c) (d)

Gambar 4.12 Hasil deteksi sukses.

4.3.2 Hasil deteksi gagal.

Deteksi gagal ditemui pada beberapa citra jalan raya dengan kelas retak. Hal ini terjadi karena guratan retak seringkali tipis sekali, jadi meskipun secara visual berbeda dengan permukaan yang tidak rusak, deteksi gagal dilakukan. Sebagaimana di contohkan pada Gambar 4.13 a, retak gagal dideteksi dan pada Gambar 4.13 b, lubang gagal dideteksi. Lubang pada gambar tersebut memiliki tekstur yang cukup kasar dan tepi yang jelas, namun gagal dikenali karena permukaan disekitar lubang juga menunjukkan tekstur yang kasar dan merata. Dua gambar ini pada semua skema pemodelan yang dilakukan gagal dideteksi.



(a) (b)

Gambar 4.13 Hasil deteksi gagal. a. deteksi gagal pada kelas retak. b. deteksi gagal pada kelas lubang.

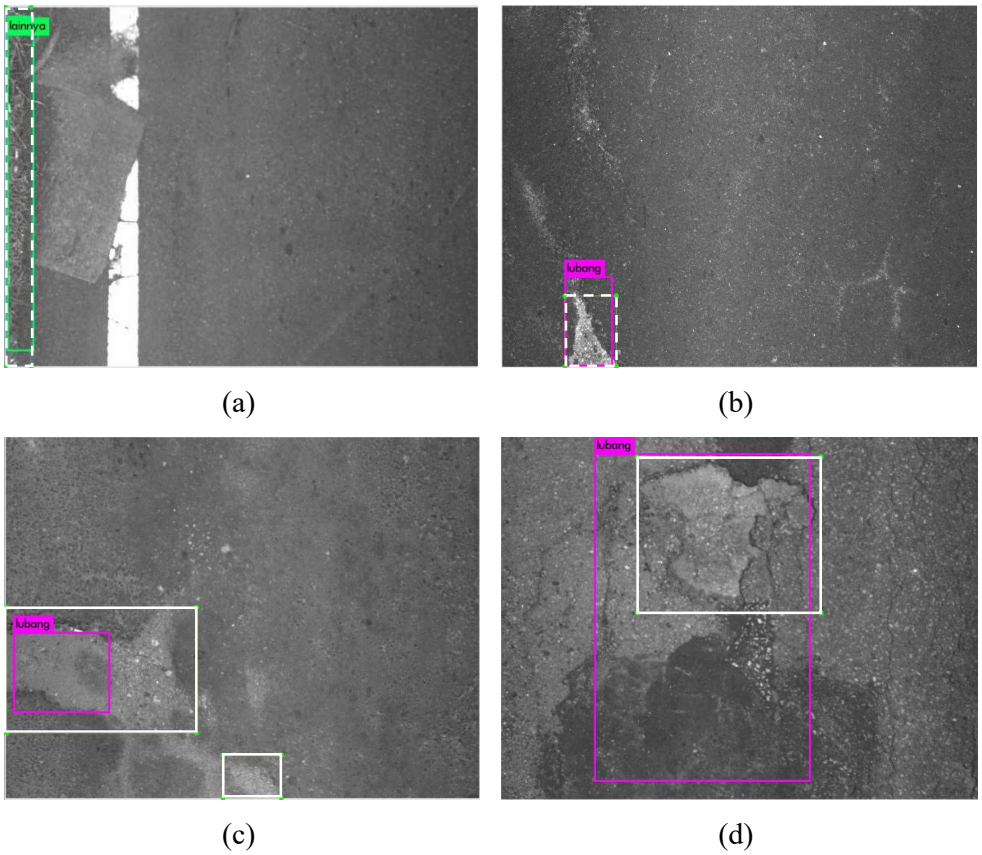
4.3.3 Deteksi pada skema pemodelan *Yolo v3* dan *Yolo v3 SPP*.

Dari pemodelan yang dilakukan menggunakan skema *transfer learning* dan arsitektur *Yolo v3* dan *Yolo v3 SPP* menunjukkan hasil yang baik dalam melakukan deteksi jika dibandingkan dengan skema pemodelan yang lain baik dari sisi arsitektur yang digunakan maupun dari sisi skema pemodelan yang dilakukan. Oleh karena itu hasil deteksi dari dua arsitektur dan skema pemodelan *transfer learning* pada *Yolo v3* dan *Yolo v3 SPP* dilakukan pengamatan secara visual. Secara umum deteksi arsitektur *Yolo v3 SPP* memberikan deteksi yang lebih baik jika dibandingkan dengan arsitektur *Yolo v3.* Meskipun kadang pada salah satu skema pemodelan *Yolo v3 SPP* tidak dapat melakukan deteksi pada satu obyek kerusakan, namun pada skema pemodelan *Yolo v3 SPP* yang lain dapat dilakukan deteksi

dengan baik. Sebagai contoh dipilih empat data citra jalan raya yang gagal dideteksi menggunakan *Yolo v3* dan *Yolo v3 SPP*, yaitu:

1. Satu citra jalan raya dengan kategori lainnya sukses dideteksi menggunakan arsitektur *Yolo v3 SPP*, namun gagal dideteksi menggunakan arsitektur *Yolo v3.*
2. Tiga citra jalan raya dengan kategori lubang sukses dideteksi menggunakan arsitektur *Yolo v3 SPP*, namun gagal dideteksi menggunakan arsitektur *Yolo v3.*

Secara lebih jelas contoh hasil deteksi ditunjukkan pada Gambar 4.14. Pada Gambar 4.14 c, satu obyek kerusakan lubang dapat dideteksi menggunakan arsitektur *Yolo v3 SPP* namun satu obyek yang lain deteksi gagal dilakukan. Hal ini tidak hanya menggunakan arsitektur *Yolo v3 SPP* namun pada semua pemodelan yang lain gagal dilakukan. Hal ini terjadi karena tekstur pada kerusakan lubang cukup halus meskipun bentuknya tidak rata jika dibandingkan dengan permukaan jalan tanpa kerusakan.

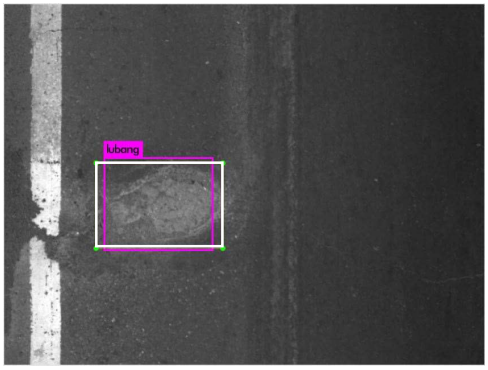


(a) (b)

(c) (d)

Gambar 4.14 Deteksi gagal pada *Yolo v3* dan sukses pada *Yolo v3 SPP.*

Meskipun secara umum, *Yolo v3 SPP* lebih sukses melakukan deteksi dibandingkan *Yolo v3*, namun terdapat obyek yang gagal dideteksi oleh *Yolo v3 SPP* namun sukses dilakukan deteksi menggunakan arsitektur *Yolo v3.* Hasil deteksi sukses yang dilakukan menggunakan arsitektur *Yolo v3* ditunjukkan pada Gambar 4.15. Obyek lubang pada Gambar 4.15 ini gagal dideteksi pada semua pemodelan yang dilakukan menggunakan *Yolo v3 SPP.*



Gambar 4.15 Deteksi gagal pada *Yolo v3 SPP* dan sukses pada *Yolo v3.*

4.4 Proses identifikasi kerusakan jalan secara manual dan yang dilakukan pada penelitian ini.

Penelitian ini diangkat dengan tujuan untuk melengkapi metode penilaian kondisi jalan yang sudah ada. Dengan batasan masalah menggunakan data dari kendaraan *survey hawk eye 2000* maka dapat diuraikan metode identifikasi kerusakan jalan yang sudah ada pada kendaraan *survey hawk eye 2000* yang akan diadopsi untuk melakukan perbaikan. Sebagaimana sudah dibahas pada metode penyiapan data penelitian, bahwa data citra yang diperoleh merupakan hasil ekstraksi *video* menjadi citra dilakukan menggunakan *software* pelengkap kendaraan survai tersebut, maka proses sebelumnya diabaikan.

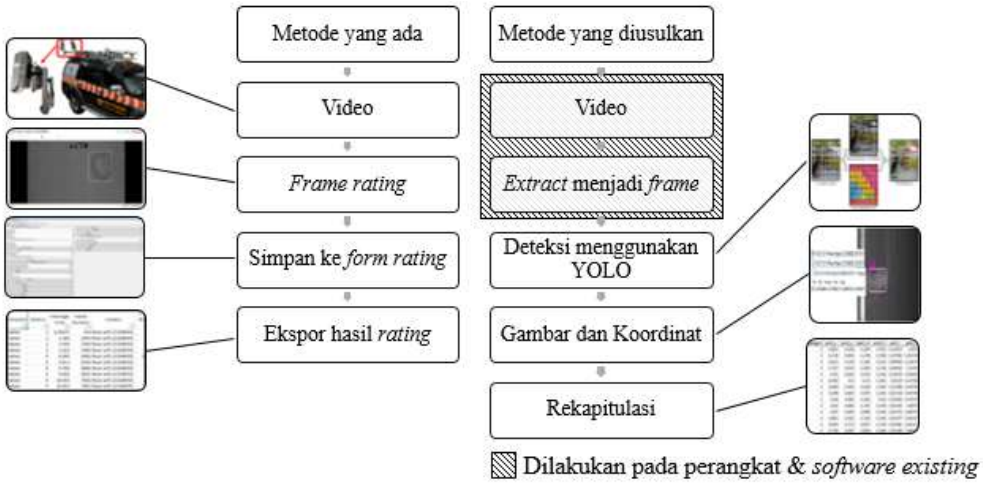
Jadi jika diuraikan perbandingan antara proses identifikasi kerusakan jalan yang sudah ada dengan proses identifikasi dan klasifikasi kerusakan jalan yang diusulkan dalam penelitian ini, maka terdapat tiga langkah yang berbeda yang dilakukan. Dimana tiga langkah tersebut dalam identifikasi kerusakan jalan secara manual, yaitu:

1. Melakukan pengamatan pada setiap *frame* menggunakan *software hawkeye processing toolkit* yang ditunjukkan pada Gambar 2.2.

2. membubuhkan kotak pada permukaan yang terdapat kerusakan yang ditunjukkan pada Gambar 2.2.
3. Melakukan *input* ukuran luas kedalam *form* yang sudah ditentukan, ditunjukkan pada Gambar 2.3.

Sedangkan pada metode yang diusulkan pada penelitian ini, ada satu langkah tambahan sebelum dilakukan identifikasi dan klasifikasi kerusakan jalan, yaitu melakukan ekstraksi video menjadi *frame* data citra tunggal yang dilakukan menggunakan *software hawkeye processing toolkit*, untuk kemudian dilakukan identifikasi dan klasifikasi kerusakan jalan secara otomatis. Adapun langkah-langkah yang dilakukan secara lebih rinci, yaitu:

1. Melakukan ekstraksi video menjadi *frame* data citra tunggal menggunakan *software hawkeye processing toolkit* kedalam satu *folder* tertentu.
2. *Folder* yang berisi citra jalan raya dilakukan deteksi menggunakan bobot/model hasil pemodelan kerusakan jalan.
3. Output deteksi dapat berupa gambar dengan deteksi kerusakan jalan, dan *file* teks untuk masing-masing data citra yang digunakan. Hasil deteksi berupa gambar ditunjukkan pada Gambar 4.12, sedangkan *file* teks dengan *format* .txt seperti ditunjukkan pada Gambar 3.4.



Gambar 4.16 Identifikasi kerusakan jalan metode manual dengan metode yang diusulkan.

Dari ketiga langkah tersebut dapat dilakukan rekapitulasi untuk memperoleh total kerusakan jalan untuk masing-masing ruas sesuai dengan yang

dibutuhkan dalam laporan penilaian kondisi jalan. Secara lebih rinci perbandingan proses identifikasi dan klasifikasi secara manual dan yang diusulkan dalam penelitian ini dapat diilustrasikan sebagaimana ditunjukkan pada Gambar 4.16.

4.5 Evaluasi waktu komputasi.

Evaluasi waktu komputasi tidak diterapkan pada proses pemodelan, namun hanya diterapkan pada proses deteksi. Meskipun waktu yang diperlukan pada proses pemodelan cukup lama sebagaimana layaknya proses pemodelan berbasis *deep learning.* Namun proses ini hanya dilakukan sekali dan bukan pada *end user.* Sehingga proses pemodelan yang lama tidak berpengaruh terhadap proses komputasi pada implementasi deteksi kerusakan jalan. Evaluai komputasi diterapkan pada dua perangkat komputer dengan spesifikasi yang berbeda, yaitu komputasi berbasis *cloud* dengan spesifikasi komputasi menggunakan *GPU* sebagaimana yang digunakan pada proses *pemodelan* dan komputasi berbasis *offline* dengan spesifikasi komputasi menggunakan *CPU.*

Pada pengukuran waktu komputasi, tidak diperhitungkan waktu yang diperlukan untuk menyimpan hasil deteksi kedalam *hardisk*, namun hanya memperhitungkan waktu satu kali *load model* yang dilanjutkan dengan deteksi dan perbandingan dengan *ground truth* keseluruhan data citra yang diujikan. *Output* berupa rekapitulasi lebih cepat diperoleh jika dibandingkan dengan deteksi kemudian dilanjutkan penulisan hasil deteksi citra satu persatu ke-*hardisk.* Kecepatan komputasi juga tidak dipengaruhi ukuran citra masukan, karena semua citra masukan akan dilakukan penyesuaian menjadi input berukuran 416×416 sesuai dengan pengaturan *network* yang diterapkan pada penelitian ini. Semua pengujian yang dilakukan pada penelitian ini menggunakan spesifikasi perangkat sebagaimana dijelaskan pada sub bab 3.7. Dengan kondisi perangkat hanya melakukan proses deteksi pada komputasi *cloud*. Sedangkan pada komputasi *ofline* perangkat disamping melakukan proses deteksi juga menjalankan aplikasi perkantoran sederhana seperti *Ms. Word* dan aplikasi pencarian *(browser)*. Proses lain yang juga berjalan pada perangkat komputasi yang digunakan juga memberikan perbedaan waktu komputasi yang dihasilkan.

4.5.1 Evaluasi waktu komputasi berbasis *cloud.*

Waktu komputasi *online* berbasis *cloud* menggunakan *google colaboratory* baik pada proses deteksi dengan satu kelas lubang maupun tiga kelas lubang, retak, dan lainnya menunjukkan waktu rata-rata yang cukup kecil untuk memproses satu gambar yaitu 0,04 detik pada deteksi lubang, dan 0,038 detik pada deteksi lubang, retak dan lainnya.

Dari hasil pengujian, semakin banyak data yang diujikan dalam satu kali proses (*batch)* maka waktu komputasi yang diperlukan untuk masing-masing gambar cenderung semakin kecil. Sebagaimana dalam pengujian kelas lubang data terbanyak dalam satu kali pengujian yaitu sebanyak 224 data, dan pada pengujian yang lain menggunakan data yang lebih kecil lagi. Jumlah data uji ini berbeda jauh jika dibandingkan data yang digunakan pada pengujian dengan tiga kelas kerusakan, kelas lubang, retak dan lainnya, yaitu sebanyak 258, 516 dan 774 data. Sehingga memberikan perbedaan waktu komputasi, dimana data pada pengujian kelas lubang, retak dan lainnya memberikan waktu komputasi yang lebih kecil.

Hal ini juga dapat diamati pada proses deteksi tiga kelas kerusakan, karena jumlah data uji yang digunakan antara pengujian yang satu dengan pengujian yang lain memiliki selisih yang cukup besar, yaitu sebanyak 258 data citra antara pengujian pertama, dan pengujian kedua, dan antara pengujian kedua dan pengujian ketiga. Sedangkan antara pengujian yang pertama dan pengujian yang ketiga mempunyai selisih sebanyak 516 data citra. Secara lebih rinci rekapitulasi waktu komputasi yang diperlukan untuk melakukan deteksi pada satu data citra secara *online* berbasis komputasi *cloud* ditunjukkan pada Tabel 4.27 dan Tabel 4.28.

Tabel 4.27 Waktu komputasi deteksi kelas lubang pada komputasi berbasis *cloud.*

No.	Jumlah Data Uji	Waktu Deteksi (dt)		
		Yolo v3	Yolo v3 Tiny	Yolo v3 SPP
1	224	0,040	0,036	0,040
2	150	0,040	0,040	0,040
3	96	0,042	0,031	0,052
4	64	0,047	0,031	0,031
5	56	0,036	0,036	0,054
6	25	0,040	0,040	0,040
Rata-rata		0,041	0,036	0,043

Tabel 4.28 Waktu komputasi deteksi kelas lubang, retak, dan lainnya pada komputasi berbasis *cloud*.

No.	Jumlah Data Latih	Jumlah Data Uji	Waktu Deteksi (dt)		
			Fine Tuning (Yolo v3)	*Transfer Learning (Yolo v3)*	*Transfer Learning (Yolo v3 SPP)*
1	516	258	0,039	0,043	0,047
2	516	516	0,037	0,039	0,039
3	516	774	0,035	0,037	0,037
4	774	258	0,043	0,039	0,043
5	774	516	0,039	0,035	0,039
6	774	774	0,035	0,032	0,039
Rata-rata			0,038	0,037	0,040

Pada deteksi satu kelas lubang dengan tiga arsitektur yaitu *Yolo v3, Yolo v3 Tiny* dan *Yolo v3 SPP* memberikan perbedaan waktu komputasi yang tidak signifikan antar arsitektur yang berbeda. Hal ini sangat dipengaruhi perangkat yang digunakan, karena cukup cepat dalam melakukan komputasi untuk semua arsitektur, maka meskipun *Yolo v3 Tiny* mempunyai arsitektur yang paling sederhana diantara dua arsitektur lainnya, selisih waktu yang dibutuhkan untuk melakukan deteksi cukup kecil, yaitu 0,005 detik lebih cepat jika dibandingkan dengan *Yolo v3* dan 0,007 detik lebih cepat jika dibandingkan dengan *Yolo v3 SPP.* Sedangkan waktu yang diperlukan antara arsitektur *Yolo v3* dengan *Yolo v3 SPP* memberikan selisih rata-rata waktu deteksi untuk masing-masing data citra sebesar 0,002 detik pada deteksi satu kelas kerusakan dan 0,003 detik pada deteksi dengan tiga kelas kerusakan.

4.5.2 Evaluasi waktu komputasi berbasis *offline* menggunakan *CPU*.

Waktu komputasi *offline* menggunakan *CPU* baik pada proses deteksi dengan satu kelas lubang ataupun tiga kelas lubang, retak, dan lainnya menunjukkan waktu rata-rata untuk memproses satu data citra lebih dari 10 detik, kecuali deteksi menggunakan model yang dikembangkan menggunakan arsitektur *Yolo v3 Tiny.* Dimana *Yolo v3 Tiny* dapat melakukan deteksi untuk masing-masing data citra dengan waktu kurang dari 2 detik. Hal ini menunjukkan bahwa *Yolo v3 Tiny* dapat melakukan deteksi dengan rata-rata kecepatan $6\times$ lebih cepat dibandingkan dengan dua arsitektur lainnya yaitu *Yolo v3* dan *Yolo v3 SPP.*

Dari hasil pengujian waktu komputasi untuk masing-masing data citra memiliki sebaran nilai yang beragam khususnya pada arsitektur *Yolo v3* dan *Yolo v3 SPP*. Hal ini dipengaruhi oleh proses lain yang berjalan pada perangkat yang digunakan untuk melakukan deteksi. Dengan rata-rata proses tercepat selama 9,24 detik untuk satu data citra dan rata-rata proses terlama selama 17,87 detik untuk satu data citra. Secara lebih rinci rekapitulasi waktu komputasi yang diperlukan untuk melakukan deteksi pada satu data citra ditunjukkan pada Tabel 4.29 dan Tabel 4.30.

Tabel 4.29 Waktu komputasi deteksi kelas lubang pada komputasi berbasis *offline*.

No.	Jumlah Data Uji	Waktu Deteksi (dt)		
		Yolo v3	Yolo v3 Tiny	Yolo v3 SPP
1	224	11,47	1,86	9,75
2	150	13,15	1,83	11,78
3	96	12,75	1,73	10,78
4	64	9,58	1,83	10,75
5	56	9,73	1,86	10,45
6	25	9,64	1,92	10,68
Rata-rata		11,05	1,84	10,70

Tabel 4.30 Waktu komputasi deteksi kelas lubang, retak, dan lainnya pada komputasi berbasis *offline*.

No.	Jumlah Data Latih	Jumlah Data Uji	Waktu Deteksi (dt)		
			Fine Tuning (Yolo v3)	*Transfer Learning (Yolo v3)*	*Transfer Learning (Yolo v3 SPP)*
1	516	258	9,31	17,87	13,18
2	516	516	9,38	9,44	9,37
3	516	774	14,21	13,15	13,38
4	774	258	9,24	9,71	16,84
5	774	516	9,37	9,41	9,53
6	774	774	11,16	12,75	10,14
Rata-rata			10,44	12,06	12,07

Pada komputasi *offline,* deteksi dijalankan pada perangkat dengan spesifikasi Intel(R) Core(TM) i5-7200U CPU @ 2,50GHz menggunakan arsitektur *Yolo v3, Yolo v3 Tiny* dan *Yolo v3 SPP*. Dari seluruh pengujian menghasilkan rata-

rata waktu pemrosesan masing-masing gambar sebesar 11,47 detik atau ±0,083 *fps*. Sedangkan pada komputasi *cloud* menggunakan GPU, rata-rata waktu komputasi seluruh pengujian dengan arsitektur *Yolo v3* dan *Yolo v3 SPP* menghasilkan rata-rata waktu pemrosesan masing-masing gambar sebesar 0,04 detik atau ±25 *fps*. Hal ini menunjukkan bahwa komputasi menggunakan *CPU* standar menghasilkan waktu pemrosesan 283 × lebih lambat jika dibandingkan komputasi menggunakan *GPU*. Sehingga komputasi menggunakan *CPU* memberikan kemungkinan yang kecil untuk dapat diimplementasikan karena kecepatan komputasi cukup lambat. Namun komputasi menggunakan *GPU* sangat besar kemungkinannya untuk dapat memproses identifikasi dan klasifikasi kerusakan jalan secara otomatis, karena dapat memproses gambar yang cukup cepat. Sebagai contoh, jika terdapat 1 km jalan raya dengan masing-masing *frame* mewakili panjang 1,391 meter, maka hanya diperlukan waktu kurang lebih 40 detik untuk memproses semua gambar dalam rentang jarak tersebut. Hal ini menunjukkan bahwa komputasi menggunakan *GPU* dapat memproses gambar dengan cukup cepat dan memberikan peluang yang cukup bagus untuk dapat diimplementasikan.

BAB 5
PENUTUP

5.1 Kesimpulan.

Pada penelitian ini telah dilakukan proses deteksi kerusakan jalan untuk mendapatkan kelas kerusakan dan luas kerusakan dengan dua kali percobaan. Pada percobaan pertama pemodelan yang dikembangkan menggunakan lubang sebagai target keluaran. Sedangkan pada percobaan kedua target keluaran adalah lubang, retak dan lainnya. Dari masing-masing percobaan dilakukan pemodelan menggunakan beberapa varian arsitektur *Yolo v3* dan berbagai skema pemodelan dengan jumlah data latih dan data uji yang berbeda-beda.

Percobaan untuk deteksi kelas lubang dengan tiga arsitektur *Yolo v3*, yaitu *Yolo v3*, *Yolo v3 Tiny* dan *Yolo v3 SPP* memberikan hasil rata-rata akurasi *mAP* terbaik masing-masing sebesar 83,43%, 79,33%, dan 88,93%. Sedangkan perhitungan luas menunjukkan akurasi masing-masing sebesar 63,81%, 51,34%, dan 70,15%.

Pada proses deteksi dengan tiga kelas yaitu lubang, retak dan lainnya skema pemodelan yang digunakan adalah *Fine Tuning* dan *Transfer Learning*. Pada skema *Fine tuning* hasil akurasi yang dihasilkan tidak cukup bagus. Akan tetapi pada skema *transfer learning* menunjukkan hasil akurasi yang cukup memuaskan. Pada skema *transfer learning* dan menggunakan arsitektur *Yolo v3* menghasilkan akurasi *mAP* terbaik rata-rata sebesar 77,45%. Sedangkan pada skema *transfer learning* dan menggunakan arsitektur *Yolo v3* dan penambahan *Spatial Pyramid Pooling* (*Yolo v3 SPP*) menghasilkan akurasi *mAP* rata-rata sebesar 80,40%. Adapun untuk perhitungan luas kerusakan pada arsitektur *Yolo v3* menghasilkan akurasi sebesar 88,69% dan arsitektur *Yolo v3 SPP* menghasilkan akurasi sebesar 91,13%.

Dari beberapa arsitektur yang digunakan, *Yolo v3 SPP* dapat melakukan deteksi lebih peka jika dibandingkan dengan arsitektur lainnya. Dimana dari sebagian besar obyek yang tidak dapat dikenali menggunakan model yang dikembangkan dengan arsitektur lain, berhasil dikenali pada model dengan

arsitektur *Yolo v3 SPP* dan hanya sebagian kecil obyek yang gagal dikenali menggunakan arsitektur *Yolo v3 SPP* akan tetapi sukses dikenali menggunakan arsitektur yang lain.

Waktu komputasi secara *online* berbasis *cloud* menggunakan *GPU* memberikan waktu rata-rata komputasi yang cukup cepat yaitu sebesar 0,04 detik per gambar atau setara dengan 25 *fps*, hal ini memberikan peluang untuk dapat dikembangkan dan diimplementasikan dalam identifikasi dan klasifikasi kerusakan jalan secara otomatis. Sedangkan komputasi secara *offline* menggunakan *CPU* kurang relevan untuk diimplementasikan karena diperlukan waktu 283× lebih lambat jika dibandingkan komputasi menggunakan *GPU.*

5.2 Saran.

Saran untuk penelitian selanjutnya yang berhubungan dengan topik penelitian ini adalah :

1. Penelitian berikutnya dapat dilakukan ekskalasi dengan melibatkan tenaga *surveyor, operator, rater* atau Ahli jalan raya untuk menentukan klasifikasi kerusakan jalan yang lebih rinci dan disepakati sesuai dengan kebutuhan peraturan yang ada. Sehingga dapat diperoleh akurasi yang lebih baik dan dapat membantu secara menyeluruh dalam proses identifikasi dan klasifikasi kerusakan jalan pada sistem penilaian kondisi jalan yang ada.
2. Penelitian berikutnya dapat dilakukan penelitian dengan basis *3D* untuk memperoleh kedalaman kerusakan jalan sehingga dapat digunakan untuk menentukan tingkat keparahan kerusakan jalan sesuai dengan kebutuhan peraturan yang ada.
3. Karena pada penelitian ini hanya menggunakan data dari beberapa ruas jalan dan distribusi data diabaikan antar ruas, maka pada penelitian selanjutnya dapat dilakukan pembagian distribusi data yang seimbang diantara beberapa ruas. Karena selama dilakukan penelitian ini ditemui karakteristik citra yang berbeda-beda meskipun memiliki kategori kerusakan yang sama.
4. Pada penelitian ini komputasi dilakukan secara *online* berbasis *cloud* dengan memanfaatkan *GPU*, dimana jika diimplementasikan maka akan

diperlukan biaya untuk *upload* data citra jalan raya yang jumlahnya cukup besar. Oleh karena itu investigasi komputasi dengan memanfaatkan *GPU* pada komputasi *offline* perlu dilakukan sehingga dapat memberikan manfaat secara menyeluruh dalam proses identifikasi dan klasifikasi kerusakan jalan pada perangkat yang sudah ada.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

AlexeyAB (2019) *The difference between Transfer learning and Fine tuning*, *January 3*. Tersedia pada: https://github.com/AlexeyAB/darknet/issues/2139 (Diakses: 26 Maret 2019).

Balai Besar Pelaksanaan Jalan Nasional VIII (2018) "Laporan Kinerja Instansi Pemerintah TA. 2017."

Beitzel, S. M., Jensen, E. C. dan Frieder, O. (2009) *MAP. In: LIU L., ÖZSU M.T. (eds) Encyclopedia of Database Systems.* Springer, Boston, MA.

Boyd, K., Eng, K. H. dan Page, C. D. (2013) "Area Under the Precision-Recall Curve : Point Estimates and Confidence Intervals."

Cesar, R. M. dan da Fontoura Costa, L. (2002) *An introduction to neural networks*, *Neurocomputing*. doi: 10.1016/s0925-2312(96)00046-x.

Chang, K. T., Chang, J. R. dan Liu, J. K. (2005) "Detection of Pavement Distresses Using 3D Laser Scanning Technology," in. doi: 10.1061/40794(179)103.

Coenen, T. B. J. dan Golroo, A. (2017) "A review on automated pavement distress detection methods," *Cogent Engineering*. Cogent, 4(1), hal. 1–23. doi: 10.1080/23311916.2017.1374822.

Colaboratory: Frequently Asked Questions (2019). Tersedia pada: https://research.google.com/colaboratory/faq.html (Diakses: 30 Maret 2019).

Goodfellow, I., Bengio, Y. dan Courville, A. (2016) *Deep learning*, *MIT Press*. Springer US. doi: 10.1007/s10710-017-9314-z.

Hidayatullah, P. *et al.* (2012) "PENDETEKSIAN LUBANG DI JALAN SECARA SEMI-OTOMATIS," *Sigma-Mu*, 4(No.1 – Maret).

Hoang, N. (2018) "An Artificial Intelligence Method for Asphalt Pavement Pothole Detection Using Least Squares Support Vector Machine and Neural Network with Steerable Filter-Based Feature Extraction," *Advances in Civil*

Engineering, 2018, hal. 1–12. doi: 10.1155/2018/7419058.

Huang, Z. dan Wang, J. (2019) "DC-SPP-YOLO: Dense Connection and Spatial Pyramid Pooling Based YOLO for Object Detection," hal. 1–23. Tersedia pada: http://arxiv.org/abs/1903.08589.

Huidrom, L., Das, L. K. dan Sud, S. K. (2013) "Method for Automated Assessment of Potholes, Cracks and Patches from Road Surface Video Clips," *Procedia - Social and Behavioral Sciences*. Elsevier B.V., 104, hal. 312–321. doi: 10.1016/j.sbspro.2013.11.124.

Kementerian Pekerjaan Umum dan Perumahan Rakyat (2016a) "Alat Survei Interurban Road Management System," *Tidak diterbitkan*.

Kementerian Pekerjaan Umum dan Perumahan Rakyat (2016b) "Penentuan Indeks Kondisi Perkerasan (IKP)," *SE Menteri PUPR*, (19/SE/M/2016).

Koch, C. dan Brilakis, I. (2011) "Pothole detection in asphalt pavement images," *Advanced Engineering Informatics*. Elsevier Ltd, 25(3), hal. 507–515. doi: 10.1016/j.aei.2011.01.002.

Koch, C., Jog, G. M. dan Brilakis, I. (2013) "Automated Pothole Distress Assessment Using Asphalt Pavement Video Data," *Journal of Computing in Civil Engineering*, 27(4), hal. 370–378. doi: 10.1061/(ASCE)CP.1943-5487.0000232.

LeCun, Y. *et al.* (1989) "Backpropagation applied to handwritten zip code recognition," *Neural Comput*, hal. 541–551. Tersedia pada: https://www.mitpressjournals.org/doi/abs/10.1162/neco.1989.1.4.541.

LeCun, Y., Bengio, Y. dan Hinton, G. (2015) "Deep learning," *Nature Methods*, 13(1), hal. 35. doi: 10.1038/nmeth.3707.

Maeda, H. *et al.* (2018) "Road Damage Detection Using Deep Neural Networks with Images Captured Through a Smartphone," 0, hal. 1–15. doi: 10.3390/ijerph13010031.

Mednis, A. *et al.* (2011) "Real time pothole detection using Android smartphones with accelerometers," in *2011 International Conference on Distributed*

Computing in Sensor Systems and Workshops (DCOSS). IEEE, hal. 1–6. doi: 10.1109/DCOSS.2011.5982206.

Moazzam, I. *et al.* (2013) "Metrology and visualization of potholes using the microsoft kinect sensor," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, (Itsc), hal. 1284–1291. doi: 10.1109/ITSC.2013.6728408.

Nair, V. dan Hinton, G. (2010) "Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proceedings of the 27th International Conference on Machine Learning*.

Nasr, G., Badr, E. dan Joun, C. (2002) "Cross Entropy Error Function in Neural Networks: Forecasting Gasoline Demand.," *FLAIRS Conference*.

Oquab, M. *et al.* (2014) "Learning and transferring mid-level image representations using convolutional neural networks," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, hal. 1717–1724. doi: 10.1109/CVPR.2014.222.

Ouma, Y. O. dan Hahn, M. (2017) "Pothole detection on asphalt pavements from 2D-colour pothole images using fuzzy c-means clustering and morphological reconstruction," *Automation in Construction*. Elsevier, 83(October 2016), hal. 196–211. doi: 10.1016/j.autcon.2017.08.017.

Parthy, K. (2018) *CS231n Convolutional Neural Networks for Visual Recognition*, *Stanford University Course cs231n*.

Redmon, J. *et al.* (2015) "You Only Look Once: Unified, Real-Time Object Detection." doi: 10.1109/CVPR.2016.91.

Redmon, J. dan Farhadi, A. (2017) "YOLO9000: Better, faster, stronger," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua, hal. 6517–6525. doi: 10.1109/CVPR.2017.690.

Redmon, J. dan Farhadi, A. (2018) "YOLOv3: An Incremental Improvement." doi: 10.1109/CVPR.2017.690.

Republik Indonesia (2004) “Undang-Undang Republik Indonesia Nomor 38 Tahun 2004 Tentang Jalan,” *Lembaran Negara RI Tahun 2004*, 132, hal. 1–43.

Turner-Fairbank Highway Research Center (2000) “Variability of Pavement Distress Data from Manual Surveys,” *Maintenance Management*, (202), hal. 0–3. Tersedia pada: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Variability+of+Pavement+Distress+Data+From+Manual+Surveys#5.

Tzutalin (2018) *LabelImg*, *Github*.

Vigneshwar, K. dan Kumar, B. H. (2016) “Detection and counting of pothole using image processing techniques,” in *2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*. IEEE, hal. 1–4. doi: 10.1109/ICCIC.2016.7919622.

Wang, H. W. *et al.* (2015) “A Real-Time Pothole Detection Approach for Intelligent Transportation System,” *Mathematical Problems in Engineering*. doi: 10.1155/2015/869627.

Zhang, L. *et al.* (2016) “Road crack detection using deep convolutional neural network,” in *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, hal. 3708–3712. doi: 10.1109/ICIP.2016.7533052.

Zhang, Z. *et al.* (2014) “An efficient algorithm for pothole detection using stereo vision,” in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. doi: 10.1109/ICASSP.2014.6853659.

Zhao, Z. Q. *et al.* (2019) “Object Detection With Deep Learning: A Review,” *IEEE Transactions on Neural Networks and Learning Systems*, 14(8). doi: 10.1109/TNNLS.2018.2876865.

LAMPIRAN

Langkah-langkah implementasi.

Garis besar implementasi proses deteksi kerusakan jalan menggunakan *CNN* dengan arsitektur *YOLO* secara teknis terdiri dari tiga hal pokok, yaitu:

1. Pemodelan dengan komputasi *cloud* menggunakan *google colaboratory.*
2. Pengujian dengan komputasi *cloud* menggunakan *google colaboratory.*
3. Pengujian dengan komputasi *offline* menggunakan sistem operasi *windows.*

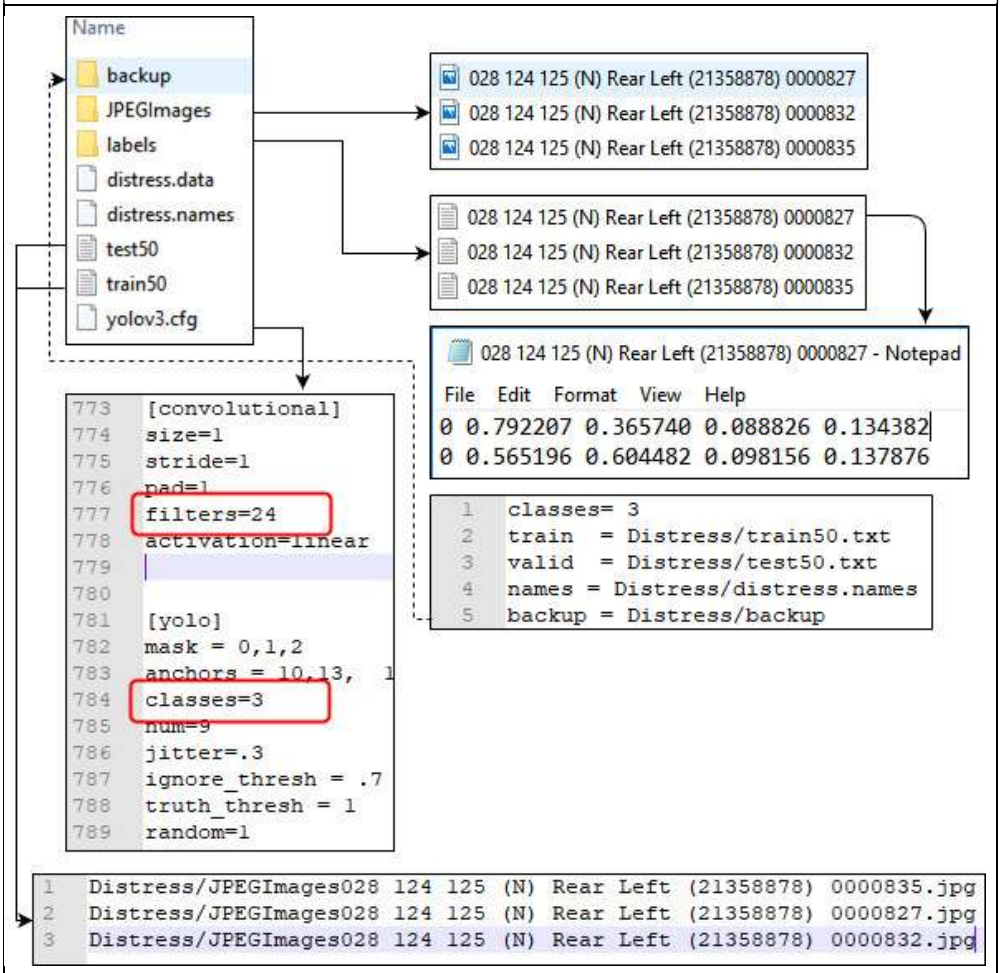
Pemodelan dengan komputasi *cloud* menggunakan *google colaboratory.*

Google colaboratory (Google colab) adalah layanan *cloud* tidak berbayar untuk pendidikan dan penelitian *machine learning*. *Google colab* memberikan *runtime* yang sepenuhnya dapat dikonfigurasikan untuk komputasi *deep learnig* dan memberikan akses tidak berbayar ke *GPU*. Oleh karena itu *Google Colab* memiliki kelemahan dalam hal waktu pemanfaatan *server*, dimana akan dilakukan *reset* secara otomatis setiap 12 jam sekali. Oleh karena itu perlu dilakukan konfigurasi agar setiap proses yang dijalankan tetap tersimpan. Skema yang dapat dilakukan yaitu:

Komputer | *Backup dan Sync* ke *google drive* | *Google drive* | *Google Colab*

1. Komputer: semua data disimpan dalam komputer.
2. *Backup dan Sync* ke *google drive*: lakukan konfigurasi untuk sinkronisasi ke *google drive* yang akan digunakan menyimpan *file* baik *file dataset*, konfigurasi, maupun hasil pemodelan (model/bobot).
3. *Google drive*
4. Jalankan pemodelan melalui *google colab.*

Struktur *File*



***SCRIPT* PEMODELAN**	
STEP 0. Konfigurasi *GPU Runtime* Arahkan ke > Menu > Runtime > Configure Runtime Type dan pilih GPU dari *Hardware accelerator drop down menu.* STEP 1. Hubungkan *colllab notebook* dengan *Google drive*	

	`from google.colab import drive`
	`drive.mount('/content/drive', force_remount=True)`
	`#arahkan ke folder tujuan`
	`ls`
	`%cd drive/'My Drive'/`

STEP 2. Download darknet	

	`!git clone https://github.com/pjreddie/darknet`
	`%cd darknet`
	`!ls`

STEP 3. Konfigurasi / Aktivasi GPU dan CUDnn	

	`#ubah makefile`
	`#GPU=1`
	`#CUDNN=1`

***SCRIPT* PEMODELAN**
STEP 4. Instalasi Darknet/Compile
!make
STEP 5. Download bobot pre-trained
!wget https://pjreddie.com/media/files/yolov3-spp.weights
STEP 6. Setting berapa layer yang akan digunakan untuk transfer learning
!./darknet partial 320U/3kelas/SPPTL50/yolov3-spp.cfg yolov3-spp.weights yolov3-spp.conv.85 85
STEP 7. Trainin Model
!./darknet detector train Distress/distress.data Distress/yolov3-spp.cfg yolov3-spp.conv.85 2>&1 | tee Distress/log/000.txt #log/xxx.txt digunakan untuk menyimpan *log*, digunakan untuk mengetahu pergerakan *loss* selama proses pemodelan
Jika Proses training terputus, lanjutkan training dengan script berikut
!./darknet detector train Distress/distress.data Distress/yolov3-spp.cfg Distress/backup/yolov3-spp_1000.weights 2>&1 | tee Distress/log/000.txt

***SCRIPT* PENGUJIAN**
Pengujian satu data gambar -> output berupa gambar
!./darknet detector test 320U/3kelas/7525/distress.data 320U/3kelas/7525/yolov3.cfg 320U/3kelas/7525/backup/yolov3_8800.weights '320U/3kelas/PREDIKSI/DataTest25/028 124 125 (N) Rear Left (21358878) 0000874.jpg'
Pengujiaan banyak data gambar -> input satu folder PREDIKSI/DataTest75, Output satu folder JPG untuk gambar terdeteksinya, dan satu folder TXT untuk labelnya.

import os
import glob
import PIL
import PIL.Image as Image
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import subprocess
import time
 	
%matplotlib inline
i = 0
images = []
for img_path in glob.glob('Distress/PREDIKSI/DataTest75/*.jpg'):
  images.append(mpimg.imread(img_path))
  print('Proses gambar '+str(i)+" : "+img_path)

***SCRIPT* PENGUJIAN**

```
    commands      =      ['./darknet      detector      test
Distress/distress.data                    Distress/yolov3.cfg
Distress/backup/yolov3_8800.weights',"'"+ img_path +"'"]
    perintah = ' '
    filename_w_ext = os.path.basename(img_path)
    filename,                file_extension                =
os.path.splitext(filename_w_ext)
    mulai = time.asctime(time.localtime(time.time()) )
    print("Mulai :", mulai)

    print ('Predictions..................')
    os.system(perintah.join(commands))
    print('Eksekusi : ' + perintah.join(commands))

    exists = os.path.isfile('predictions.jpg')
    if exists:
      #print ('Predictions_' + filename + '.jpg  Sudah Ada')
      #print ('Nama File Akan di Ganti')

      os.rename("predictions.jpg", "Distress/PREDIKSI/ JPG/"
+ filename + ".jpg")
      os.rename("predictions.txt", "Distress/PREDIKSI/TXT/ "
+ filename + ".txt")
      print ('Predictions Done')
      selesai = time.asctime(time.localtime(time.time()) )
      print("Selesai :", selesai)
    else:
      print (filename + '.jpg  Tidak Ada')

    i += 1
```

Untuk dapat menampilkan koordinat hasil deteksi, lakukan konfigurasi pada file src/image.c kemudian lakukan compile kembali dengan perintah !make

```
void draw_detections(image im, detection *dets, int num, float
thresh, char **names, image **alphabet, int classes)
{
    int i,j;

        /*
     * when producing the output image, output the prediction
as a text file
     * Print the output class, confidence and the bounding box
coordinate
     * Prediction: <class>   <confidence>  Location: <Left>
<Right> <Top> <Bottom>
     */
    FILE *out_fd = fopen("predictions.txt", "w");

    if (out_fd == NULL)
    {
        printf("Error opening file!\n");
        exit(1);
    }
```

***SCRIPT* PENGUJIAN**

```
    for(i = 0; i < num; ++i){
        char labelstr[4096] = {0};
        int class = -1;
        for(j = 0; j < classes; ++j){
            if (dets[i].prob[j] > thresh){
                if (class < 0) {
                    strcat(labelstr, names[j]);
                    class = j;
                } else {
                    strcat(labelstr, ", ");
                    strcat(labelstr, names[j]);
                }
                printf("%s:       %.0f%%\n",       names[j],
dets[i].prob[j]*100);
            }
        }
        if(class >= 0){
            int width = im.h * .006;

            /*
               if(0){
               width = pow(prob, 1./2.)*10+1;
               alphabet = 0;
               }
             */

            //printf("%d  %s:  %.0f%%\n",  i,  names[class],
prob*100);
            int offset = class*123457 % classes;
            float red = get_color(2,offset,classes);
            float green = get_color(1,offset,classes);
            float blue = get_color(0,offset,classes);
            float rgb[3];

            //width = prob*20+2;

            rgb[0] = red;
            rgb[1] = green;
            rgb[2] = blue;
            box b = dets[i].bbox;
            //printf("%f %f %f %f\n", b.x, b.y, b.w, b.h);

            int left  = (b.x-b.w/2.)*im.w;
            int right = (b.x+b.w/2.)*im.w;
            int top   = (b.y-b.h/2.)*im.h;
            int bot   = (b.y+b.h/2.)*im.h;

            if(left < 0) left = 0;
            if(right > im.w-1) right = im.w-1;
            if(top < 0) top = 0;
            if(bot > im.h-1) bot = im.h-1;
```

***SCRIPT* PENGUJIAN**

```
            fprintf(out_fd,"%d %f %f %f %f\n",class, b.x, b.y,
b.w, b.h);

            draw_box_width(im, left, top, right, bot, width,
red, green, blue);
            if (alphabet) {
                image label = get_label(alphabet, labelstr,
(im.h*.03));
                draw_label(im, top + width, left, label, rgb);
                free_image(label);
            }
            if (dets[i].mask){
                image mask = float_to_image(14, 14, 1,
dets[i].mask);
                image resized_mask = resize_image(mask,
b.w*im.w, b.h*im.h);
                image tmask = threshold_image(resized_mask,
.5);
                embed_image(tmask, im, left, top);
                free_image(mask);
                free_image(resized_mask);
                free_image(tmask);
            }
        }
    }
    fclose(out_fd);
}
```

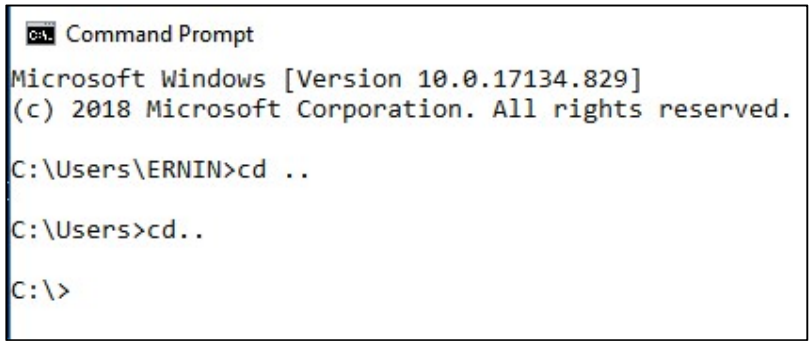
Pengujian untuk memperoleh rekapitulasi mAP secara keseluruhan

```
!./darknet detector map Distress/distress50.data Distress/mp-
yolov3-spp.cfg Distress/backup/yolov3-spp_1100.weights 2>&1 |
tee Distress/hasil50/1100.txt
```

Langkah-langkah pengujian *offline* menggunakan sistem operasi Windows 10

1. Download Cygwin : http://www.cygwin.com/
2. Install → pilih fiturnya : git, make, ssh, ssl, dan python.
3. Agar dapat diakses di cmd windows : windows → start → system → advanced system setting → environment variable → di tab system variable → pilih path → edit → tambahkan dimana Cygwin di install → misalnya C:Cygwin → OK
4. `cmd`
5. Arahkan ke drive C:

Langkah-langkah pengujian *offline* menggunakan sistem operasi Windows 10



6. Clone darknet :

```
git clone https://github.com/pjreddie/darknet
```

7. `cd darknet`
8. `make`
9. Installasi darknet sukses.
10. *Copy folder* Distress ke dalam *folder* darknet.
11. Jalankan script berikut untuk ngetes berhasil atau tidak untuk deteksi:

```
darknet.exe detector map Distress/distress.data Distress
/yolov3.cfg Distress /backup/yolov3_6400.weights Distress
/JPEGImages/'028 124 125 (N) Rear Left (21358878)
0005369.jpg'
```

12. Output terdapat di *folder* darknet/prediction.jpg
13. Gunakan *command* yang sama dengan proses pengujian secara *online* untuk kebutuhan lainnya.

Arsitektur *Yolo v3* Percobaan Kelas Lubang.

Arsitektur *Yolo v3* Percobaan Kelas Lubang

```
layer     filters    size              input                output
   0 conv     323 x 3 / 1 416 x 416 x   3 ->  416 x 416 x  32 0.299 BF
   1 conv     643 x 3 / 2 416 x 416 x  32 ->  208 x 208 x  64 1.595 BF
   2 conv      32 1 x 1 / 1 208 x 208 x  64 ->  208 x 208 x  32 0.177 BF
   3 conv      64 3 x 3 / 1 208 x 208 x  32 ->  208 x 208 x  64 1.595 BF
   4 Shortcut Layer: 1
   5 conv    128 3 x 3 / 2 208 x 208 x  64 ->  104 x 104 x 128 1.595 BF
   6 conv     64 1 x 1 / 1 104 x 104 x 128 ->  104 x 104 x  64 0.177 BF
   7 conv    128 3 x 3 / 1 104 x 104 x  64 ->  104 x 104 x 128 1.595 BF
   8 Shortcut Layer: 5
   9 conv     64 1 x 1 / 1 104 x 104 x 128 ->  104 x 104 x  64 0.177 BF
  10 conv    128 3 x 3 / 1 104 x 104 x  64 ->  104 x 104 x 128 1.595 BF
  11 Shortcut Layer: 8
  12 conv    256 3 x 3 / 2 104 x 104 x 128 ->   52 x  52 x 256 1.595 BF
  13 conv    128 1 x 1 / 1  52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
  14 conv    256 3 x 3 / 1  52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
  15 Shortcut Layer: 12
  16 conv    128 1 x 1 / 1  52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
  17 conv    256 3 x 3 / 1  52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
  18 Shortcut Layer: 15
```

Arsitektur *Yolo v3* Percobaan Kelas Lubang

```
  19 conv    128 1 x 1 / 1  52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
  20 conv    256 3 x 3 / 1  52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
  21 Shortcut Layer: 18
  22 conv    128 1 x 1 / 1  52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
  23 conv    256 3 x 3 / 1  52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
  24 Shortcut Layer: 21
  25 conv    128 1 x 1 / 1  52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
  26 conv    256 3 x 3 / 1  52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
  27 Shortcut Layer: 24
  28 conv    128 1 x 1 / 1  52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
  29 conv    256 3 x 3 / 1  52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
  30 Shortcut Layer: 27
  31 conv    128 1 x 1 / 1  52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
  32 conv    256 3 x 3 / 1  52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
  33 Shortcut Layer: 30
  34 conv    128 1 x 1 / 1  52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
  35 conv    256 3 x 3 / 1  52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
  36 Shortcut Layer: 33
  37 conv    512 3 x 3 / 2  52 x  52 x 256 ->   26 x  26 x 512 1.595 BF
  38 conv    256 1 x 1 / 1  26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
  39 conv    512 3 x 3 / 1  26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
  40 Shortcut Layer: 37
  41 conv    256 1 x 1 / 1  26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
  42 conv    512 3 x 3 / 1  26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
  43 Shortcut Layer: 40
  44 conv    256 1 x 1 / 1  26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
  45 conv    512 3 x 3 / 1  26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
  46 Shortcut Layer: 43
  47 conv    256 1 x 1 / 1  26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
  48 conv    512 3 x 3 / 1  26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
  49 Shortcut Layer: 46
  50 conv    256 1 x 1 / 1  26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
  51 conv    512 3 x 3 / 1  26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
  52 Shortcut Layer: 49
  53 conv    256 1 x 1 / 1  26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
  54 conv    512 3 x 3 / 1  26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
  55 Shortcut Layer: 52
  56 conv    256 1 x 1 / 1  26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
  57 conv    512 3 x 3 / 1  26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
  58 Shortcut Layer: 55
  59 conv    256 1 x 1 / 1  26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
  60 conv    512 3 x 3 / 1  26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
  61 Shortcut Layer: 58
  62 conv   1024 3 x 3 / 2  26 x  26 x 512 ->   13 x  13 x1024 1.595 BF
  63 conv    512 1 x 1 / 1  13 x  13 x1024 ->   13 x  13 x 512 0.177 BF
  64 conv   1024 3 x 3 / 1  13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
  65 Shortcut Layer: 62
  66 conv    512 1 x 1 / 1  13 x  13 x1024 ->   13 x  13 x 512 0.177 BF
  67 conv   1024 3 x 3 / 1  13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
  68 Shortcut Layer: 65
  69 conv    512 1 x 1 / 1  13 x  13 x1024 ->   13 x  13 x 512 0.177 BF
  70 conv   1024 3 x 3 / 1  13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
  71 Shortcut Layer: 68
  72 conv    512 1 x 1 / 1  13 x  13 x1024 ->   13 x  13 x 512 0.177 BF
  73 conv   1024 3 x 3 / 1  13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
  74 Shortcut Layer: 71
  75 conv    512 1 x 1 / 1  13 x  13 x1024 ->   13 x  13 x 512 0.177 BF
  76 conv   1024 3 x 3 / 1  13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
  77 conv    512 1 x 1 / 1  13 x  13 x1024 ->   13 x  13 x 512 0.177 BF
  78 conv   1024 3 x 3 / 1  13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
  79 conv    512 1 x 1 / 1  13 x  13 x1024 ->   13 x  13 x 512 0.177 BF
```

Arsitektur *Yolo v3* Percobaan Kelas Lubang

```
  80 conv   1024 3 x 3 / 1  13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
  81 conv     18 1 x 1 / 1  13 x  13 x1024 ->   13 x  13 x  18 0.006 BF
  82 yolo
  83 route  79
  84 conv    256 1 x 1 / 1  13 x  13 x 512 ->   13 x  13 x 256 0.044 BF
  85 upsample            2x 13 x  13 x 256 ->   26 x  26 x 256
  86 route  85 61
  87 conv    256 1 x 1 / 1  26 x  26 x 768 ->   26 x  26 x 256 0.266 BF
  88 conv    512 3 x 3 / 1  26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
  89 conv    256 1 x 1 / 1  26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
  90 conv    512 3 x 3 / 1  26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
  91 conv    256 1 x 1 / 1  26 x  26 x 512 ->   26 x  26 x 256 0.177 BF
  92 conv    512 3 x 3 / 1  26 x  26 x 256 ->   26 x  26 x 512 1.595 BF
  93 conv     18 1 x 1 / 1  26 x  26 x 512 ->   26 x  26 x  18 0.012 BF
  94 yolo
  95 route  91
  96 conv    128 1 x 1 / 1  26 x  26 x 256 ->   26 x  26 x 128 0.044 BF
  97 upsample            2x 26 x  26 x 128 ->   52 x  52 x 128
  98 route  97 36
  99 conv    128 1 x 1 / 1  52 x  52 x 384 ->   52 x  52 x 128 0.266 BF
 100 conv    256 3 x 3 / 1  52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
 101 conv    128 1 x 1 / 1  52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
 102 conv    256 3 x 3 / 1  52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
 103 conv    128 1 x 1 / 1  52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
 104 conv    256 3 x 3 / 1  52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
 105 conv     18 1 x 1 / 1  52 x  52 x 256 ->   52 x  52 x  18 0.025 BF
 106 yolo
```

Arsitektur *Yolo v3 Tiny* Percobaan Kelas Lubang

```
layer     filters    size              input                output
   0 conv     16 3 x 3 / 1 416 x 416 x   3  -> 416 x 416 x  16 0.150 BF
   1 max         2 x 2 / 2 416 x 416 x  16  -> 208 x 208 x  16 0.003 BF
   2 conv     32 3 x 3 / 1 208 x 208 x  16  -> 208 x 208 x  32 0.399 BF
   3 max         2 x 2 / 2 208 x 208 x  32  -> 104 x 104 x  32 0.001 BF
   4 conv     64 3 x 3 / 1 104 x 104 x  32  -> 104 x 104 x  64 0.399 BF
   5 max         2 x 2 / 2 104 x 104 x  64  ->  52 x  52 x  64 0.001 BF
   6 conv    128 3 x 3 / 1  52 x  52 x  64  ->  52 x  52 x 128 0.399 BF
   7 max         2 x 2 / 2  52 x  52 x 128  ->  26 x  26 x 128 0.000 BF
   8 conv    256 3 x 3 / 1  26 x  26 x 128  ->  26 x  26 x 256 0.399 BF
   9 max         2 x 2 / 2  26 x  26 x 256  ->  13 x  13 x 256 0.000 BF
  10 conv    512 3 x 3 / 1  13 x  13 x 256  ->  13 x  13 x 512 0.399 BF
  11 max         2 x 2 / 1  13 x  13 x 512  ->  13 x  13 x 512 0.000 BF
  12 conv   1024 3 x 3 / 1  13 x  13 x 512  ->  13 x  13 x1024 1.595 BF
  13 conv    256 1 x 1 / 1  13 x  13 x1024  ->  13 x  13 x 256 0.089 BF
  14 conv    512 3 x 3 / 1  13 x  13 x 256  ->  13 x  13 x 512 0.399 BF
  15 conv     18 1 x 1 / 1  13 x  13 x 512  ->  13 x  13 x  18 0.003 BF
  16 yolo
  17 route  13
  18 conv    128 1 x 1 / 1  13 x  13 x 256  ->  13 x  13 x 128 0.011 BF
  19 upsample           2x  13 x  13 x 128  ->  26 x  26 x 128
  20 route  19 8
  21 conv    256 3 x 3 / 1  26 x  26 x 384  ->  26 x  26 x 256 1.196 BF
  22 conv    18  1 x 1 / 1  26 x  26 x 256  ->  26 x  26 x  18 0.006 BF
  23 yolo
```

Arsitektur *Yolo V3 SPP* Percobaan Kelas Lubang

```
layer     filters    size              input                output
   0 conv     32  3 x 3 / 1 416 x 416 x   3 -> 416 x 416 x  32 0.299 BF
   1 conv     64  3 x 3 / 2 416 x 416 x  32 -> 208 x 208 x  64 1.595 BF
   2 conv     32  1 x 1 / 1 208 x 208 x  64 -> 208 x 208 x  32 0.177 BF
   3 conv     64  3 x 3 / 1 208 x 208 x  32 -> 208 x 208 x  64 1.595 BF
   4 Shortcut Layer: 1
   5 conv    128  3 x 3 / 2 208 x 208 x  64 -> 104 x 104 x 128 1.595 BF
   6 conv     64  1 x 1 / 1 104 x 104 x 128 -> 104 x 104 x  64 0.177 BF
   7 conv    128  3 x 3 / 1 104 x 104 x  64 -> 104 x 104 x 128 1.595 BF
   8 Shortcut Layer: 5
   9 conv     64  1 x 1 / 1 104 x 104 x 128 -> 104 x 104 x  64 0.177 BF
  10 conv    128  3 x 3 / 1 104 x 104 x  64 -> 104 x 104 x 128 1.595 BF
  11 Shortcut Layer: 8
  12 conv    256  3 x 3 / 2  104 x 104 x 128 ->  52 x  52 x 256 1.595 BF
  13 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
  14 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
  15 Shortcut Layer: 12
  16 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
  17 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
  18 Shortcut Layer: 15
  19 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
  20 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
  21 Shortcut Layer: 18
  22 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
  23 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
  24 Shortcut Layer: 21
  25 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
  26 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
  27 Shortcut Layer: 24
  28 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
  29 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
  30 Shortcut Layer: 27
  31 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
  32 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
  33 Shortcut Layer: 30
  34 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
  35 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
  36 Shortcut Layer: 33
  37 conv    512  3 x 3 / 2  52 x  52 x 256 ->  26 x  26 x 512 1.595 BF
  38 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
  39 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
  40 Shortcut Layer: 37
  41 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
  42 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
  43 Shortcut Layer: 40
  44 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
  45 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
  46 Shortcut Layer: 43
  47 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
  48 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
  49 Shortcut Layer: 46
  50 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
  51 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
  52 Shortcut Layer: 49
  53 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
  54 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
  55 Shortcut Layer: 52
  56 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
  57 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
  58 Shortcut Layer: 55
  59 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
```

Arsitektur *Yolo V3 SPP* Percobaan Kelas Lubang

```
  60 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
  61 Shortcut Layer: 58
  62 conv   1024  3 x 3 / 2  26 x  26 x 512 ->  13 x  13 x1024 1.595 BF
  63 conv    512  1 x 1 / 1  13 x  13 x1024 ->  13 x  13 x 512 0.177 BF
  64 conv   1024  3 x 3 / 1  13 x  13 x 512 ->  13 x  13 x1024 1.595 BF
  65 Shortcut Layer: 62
  66 conv    512  1 x 1 / 1  13 x  13 x1024 ->  13 x  13 x 512 0.177 BF
  67 conv   1024  3 x 3 / 1  13 x  13 x 512 ->  13 x  13 x1024 1.595 BF
  68 Shortcut Layer: 65
  69 conv    512  1 x 1 / 1  13 x  13 x1024 ->  13 x  13 x 512 0.177 BF
  70 conv   1024  3 x 3 / 1  13 x  13 x 512 ->  13 x  13 x1024 1.595 BF
  71 Shortcut Layer: 68
  72 conv    512  1 x 1 / 1  13 x  13 x1024 ->  13 x  13 x 512 0.177 BF
  73 conv   1024  3 x 3 / 1  13 x  13 x 512 ->  13 x  13 x1024 1.595 BF
  74 Shortcut Layer: 71
  75 conv    512  1 x 1 / 1  13 x  13 x1024 ->  13 x  13 x 512 0.177 BF
  76 conv   1024  3 x 3 / 1  13 x  13 x 512 ->  13 x  13 x1024 1.595 BF
  77 conv    512  1 x 1 / 1  13 x  13 x1024 ->  13 x  13 x 512 0.177 BF
  78 max          5 x 5 / 1  13 x  13 x 512 ->  13 x  13 x 512 0.002 BF
  79 route  77
  80 max          9 x 9 / 1  13 x  13 x 512 ->  13 x  13 x 512 0.007 BF
  81 route  77
  82 max         13 x 13 / 1 13 x  13 x 512 ->  13 x  13 x 512 0.015 BF
  83 route  82 80 78 77
  84 conv    512  1 x 1 / 1  13 x  13 x2048 ->  13 x  13 x 512 0.354 BF
  85 conv   1024  3 x 3 / 1  13 x  13 x 512 ->  13 x  13 x1024 1.595 BF
  86 conv    512  1 x 1 / 1  13 x  13 x1024 ->  13 x  13 x 512 0.177 BF
  87 conv   1024  3 x 3 / 1  13 x  13 x 512 ->  13 x  13 x1024 1.595 BF
  88 conv     18  1 x 1 / 1  13 x  13 x1024 ->  13 x  13 x  18 0.006 BF
  89 yolo
  90 route  86
  91 conv    256  1 x 1 / 1  13 x  13 x 512 ->  13 x  13 x 256 0.044 BF
  92 upsample            2x  13 x  13 x 256 ->  26 x  26 x 256
  93 route  92 61
  94 conv    256  1 x 1 / 1  26 x  26 x 768 ->  26 x  26 x 256 0.266 BF
  95 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
  96 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
  97 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
  98 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
  99 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
 100 conv     18  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x  18 0.012 BF
 101 yolo
 102 route  98
 103 conv    128  1 x 1 / 1  26 x  26 x 256 ->  26 x  26 x 128 0.044 BF
 104 upsample            2x  26 x  26 x 128 ->  52 x  52 x 128
 105 route  104 36
 106 conv    128  1 x 1 / 1  52 x  52 x 384 ->  52 x  52 x 128 0.266 BF
 107 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
 108 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
 109 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
 110 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
 111 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
 112 conv     18  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x  18 0.025 BF
 113 yolo
```

Arsitektur *Yolo v3* Percobaan Kelas Lubang, Retak dan Lainnya.

Arsitektur *Yolo V3* Percobaan Kelas Lubang, Retak dan Lainnya

```
layer     filters    size              input                output
   0 conv     32  3 x 3 / 1 416 x 416 x   3 -> 416 x 416 x  32 0.299 BF
   1 conv     64  3 x 3 / 2 416 x 416 x  32 -> 208 x 208 x  64 1.595 BF
   2 conv     32  1 x 1 / 1 208 x 208 x  64 -> 208 x 208 x  32 0.177 BF
   3 conv     64  3 x 3 / 1 208 x 208 x  32 -> 208 x 208 x  64 1.595 BF
   4 Shortcut Layer: 1
   5 conv    128  3 x 3 / 2 208 x 208 x  64 -> 104 x 104 x 128 1.595 BF
   6 conv     64  1 x 1 / 1 104 x 104 x 128 -> 104 x 104 x  64 0.177 BF
   7 conv    128  3 x 3 / 1 104 x 104 x  64 -> 104 x 104 x 128 1.595 BF
   8 Shortcut Layer: 5
   9 conv     64  1 x 1 / 1 104 x 104 x 128 -> 104 x 104 x  64 0.177 BF
  10 conv    128  3 x 3 / 1 104 x 104 x  64 -> 104 x 104 x 128 1.595 BF
  11 Shortcut Layer: 8
  12 conv    256  3 x 3 / 2 104 x 104 x 128 ->  52 x  52 x 256 1.595 BF
  13 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
  14 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
  15 Shortcut Layer: 12
  16 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
  17 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
  18 Shortcut Layer: 15
  19 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
  20 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
  21 Shortcut Layer: 18
  22 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
  23 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
  24 Shortcut Layer: 21
  25 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
  26 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
  27 Shortcut Layer: 24
  28 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
  29 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
  30 Shortcut Layer: 27
  31 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
  32 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
  33 Shortcut Layer: 30
  34 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
  35 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
  36 Shortcut Layer: 33
  37 conv    512  3 x 3 / 2  52 x  52 x 256 ->  26 x  26 x 512 1.595 BF
  38 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
  39 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
  40 Shortcut Layer: 37
  41 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
  42 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
  43 Shortcut Layer: 40
  44 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
  45 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
  46 Shortcut Layer: 43
  47 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
  48 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
  49 Shortcut Layer: 46
  50 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
  51 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
  52 Shortcut Layer: 49
  53 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
  54 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
  55 Shortcut Layer: 52
  56 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
```

Arsitektur *Yolo V3* Percobaan Kelas Lubang, Retak dan Lainnya

```
  57 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
  58 Shortcut Layer: 55
  59 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
  60 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
  61 Shortcut Layer: 58
  62 conv   1024  3 x 3 / 2  26 x  26 x 512 ->  13 x  13 x1024 1.595 BF
  63 conv    512  1 x 1 / 1  13 x  13 x1024 ->  13 x  13 x 512 0.177 BF
  64 conv   1024  3 x 3 / 1  13 x  13 x 512 ->  13 x  13 x1024 1.595 BF
  65 Shortcut Layer: 62
  66 conv    512  1 x 1 / 1  13 x  13 x1024 ->  13 x  13 x 512 0.177 BF
  67 conv   1024  3 x 3 / 1  13 x  13 x 512 ->  13 x  13 x1024 1.595 BF
  68 Shortcut Layer: 65
  69 conv    512  1 x 1 / 1  13 x  13 x1024 ->  13 x  13 x 512 0.177 BF
  70 conv   1024  3 x 3 / 1  13 x  13 x 512 ->  13 x  13 x1024 1.595 BF
  71 Shortcut Layer: 68
  72 conv    512  1 x 1 / 1  13 x  13 x1024 ->  13 x  13 x 512 0.177 BF
  73 conv   1024  3 x 3 / 1  13 x  13 x 512 ->  13 x  13 x1024 1.595 BF
  74 Shortcut Layer: 71
  75 conv    512  1 x 1 / 1  13 x  13 x1024 ->  13 x  13 x 512 0.177 BF
  76 conv   1024  3 x 3 / 1  13 x  13 x 512 ->  13 x  13 x1024 1.595 BF
  77 conv    512  1 x 1 / 1  13 x  13 x1024 ->  13 x  13 x 512 0.177 BF
  78 conv   1024  3 x 3 / 1  13 x  13 x 512 ->  13 x  13 x1024 1.595 BF
  79 conv    512  1 x 1 / 1  13 x  13 x1024 ->  13 x  13 x 512 0.177 BF
  80 conv   1024  3 x 3 / 1  13 x  13 x 512 ->  13 x  13 x1024 1.595 BF
  81 conv     18  1 x 1 / 1  13 x  13 x1024 ->  13 x  13 x  24 0.006 BF
  82 yolo
  83 route  79
  84 conv    256  1 x 1 / 1  13 x  13 x 512 ->  13 x  13 x 256 0.044 BF
  85 upsample            2x  13 x  13 x 256 ->  26 x  26 x 256
  86 route  85 61
  87 conv    256  1 x 1 / 1  26 x  26 x 768 ->  26 x  26 x 256 0.266 BF
  88 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
  89 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
  90 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
  91 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
  92 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
  93 conv     18  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x  24 0.012 BF
  94 yolo
  95 route  91
  96 conv    128  1 x 1 / 1  26 x  26 x 256 ->  26 x  26 x 128 0.044 BF
  97 upsample            2x  26 x  26 x 128 ->  52 x  52 x 128
  98 route  97 36
  99 conv    128  1 x 1 / 1  52 x  52 x 384 ->  52 x  52 x 128 0.266 BF
 100 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
 101 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
 102 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
 103 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
 104 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
 105 conv     18  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x  24 0.025 BF
 106 yolo
```

Arsitektur *Yolo V3 SPP* Percobaan Kelas Lubang, Retak dan Lainnya

```
layer     filters    size              input                output
   0 conv     32  3 x 3 / 1 416 x 416 x   3 -> 416 x 416 x  32 0.299 BF
   1 conv     64  3 x 3 / 2 416 x 416 x  32 -> 208 x 208 x  64 1.595 BF
   2 conv     32  1 x 1 / 1 208 x 208 x  64 -> 208 x 208 x  32 0.177 BF
   3 conv     64  3 x 3 / 1 208 x 208 x  32 -> 208 x 208 x  64 1.595 BF
```

Arsitektur *Yolo V3 SPP* Percobaan Kelas Lubang, Retak dan Lainnya

```
  4 Shortcut Layer: 1
  5 conv    128  3 x 3 / 2 208 x 208 x  64 -> 104 x 104 x 128 1.595 BF
  6 conv     64  1 x 1 / 1 104 x 104 x 128 -> 104 x 104 x  64 0.177 BF
  7 conv    128  3 x 3 / 1 104 x 104 x  64 -> 104 x 104 x 128 1.595 BF
  8 Shortcut Layer: 5
  9 conv     64  1 x 1 / 1 104 x 104 x 128 -> 104 x 104 x  64 0.177 BF
 10 conv    128  3 x 3 / 1 104 x 104 x  64 -> 104 x 104 x 128 1.595 BF
 11 Shortcut Layer: 8
 12 conv    256  3 x 3 / 2  104 x 104 x 128 ->  52 x  52 x 256 1.595 BF
 13 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
 14 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
 15 Shortcut Layer: 12
 16 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
 17 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
 18 Shortcut Layer: 15
 19 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
 20 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
 21 Shortcut Layer: 18
 22 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
 23 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
 24 Shortcut Layer: 21
 25 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
 26 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
 27 Shortcut Layer: 24
 28 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
 29 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
 30 Shortcut Layer: 27
 31 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
 32 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
 33 Shortcut Layer: 30
 34 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
 35 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
 36 Shortcut Layer: 33
 37 conv    512  3 x 3 / 2  52 x  52 x 256 ->  26 x  26 x 512 1.595 BF
 38 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
 39 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
 40 Shortcut Layer: 37
 41 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
 42 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
 43 Shortcut Layer: 40
 44 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
 45 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
 46 Shortcut Layer: 43
 47 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
 48 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
 49 Shortcut Layer: 46
 50 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
 51 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
 52 Shortcut Layer: 49
 53 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
 54 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
 55 Shortcut Layer: 52
 56 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
 57 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
 58 Shortcut Layer: 55
 59 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
 60 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
 61 Shortcut Layer: 58
 62 conv   1024  3 x 3 / 2  26 x  26 x 512 ->  13 x  13 x1024 1.595 BF
 63 conv    512  1 x 1 / 1  13 x  13 x1024 ->  13 x  13 x 512 0.177 BF
 64 conv   1024  3 x 3 / 1  13 x  13 x 512 ->  13 x  13 x1024 1.595 BF
```

Arsitektur *Yolo V3 SPP* Percobaan Kelas Lubang, Retak dan Lainnya

```
  65 Shortcut Layer: 62
  66 conv    512  1 x 1 / 1  13 x  13 x1024 ->  13 x  13 x 512 0.177 BF
  67 conv   1024  3 x 3 / 1  13 x  13 x 512 ->  13 x  13 x1024 1.595 BF
  68 Shortcut Layer: 65
  69 conv    512  1 x 1 / 1  13 x  13 x1024 ->  13 x  13 x 512 0.177 BF
  70 conv   1024  3 x 3 / 1  13 x  13 x 512 ->  13 x  13 x1024 1.595 BF
  71 Shortcut Layer: 68
  72 conv    512  1 x 1 / 1  13 x  13 x1024 ->  13 x  13 x 512 0.177 BF
  73 conv   1024  3 x 3 / 1  13 x  13 x 512 ->  13 x  13 x1024 1.595 BF
  74 Shortcut Layer: 71
  75 conv    512  1 x 1 / 1  13 x  13 x1024 ->  13 x  13 x 512 0.177 BF
  76 conv   1024  3 x 3 / 1  13 x  13 x 512 ->  13 x  13 x1024 1.595 BF
  77 conv    512  1 x 1 / 1  13 x  13 x1024 ->  13 x  13 x 512 0.177 BF
  78 max          5 x 5 / 1  13 x  13 x 512 ->  13 x  13 x 512 0.002 BF
  79 route  77
  80 max          9 x 9 / 1  13 x  13 x 512 ->  13 x  13 x 512 0.007 BF
  81 route  77
  82 max         13 x 13 / 1 13 x  13 x 512 ->  13 x  13 x 512 0.015 BF
  83 route  82 80 78 77
  84 conv    512  1 x 1 / 1  13 x  13 x2048 ->  13 x  13 x 512 0.354 BF
  85 conv   1024  3 x 3 / 1  13 x  13 x 512 ->  13 x  13 x1024 1.595 BF
  86 conv    512  1 x 1 / 1  13 x  13 x1024 ->  13 x  13 x 512 0.177 BF
  87 conv   1024  3 x 3 / 1  13 x  13 x 512 ->  13 x  13 x1024 1.595 BF
  88 conv     18  1 x 1 / 1  13 x  13 x1024 ->  13 x  13 x  24 0.006 BF
  89 yolo
  90 route  86
  91 conv    256  1 x 1 / 1  13 x  13 x 512 ->  13 x  13 x 256 0.044 BF
  92 upsample            2x  13 x  13 x 256 ->  26 x  26 x 256
  93 route  92 61
  94 conv    256  1 x 1 / 1  26 x  26 x 768 ->  26 x  26 x 256 0.266 BF
  95 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
  96 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
  97 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
  98 conv    256  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x 256 0.177 BF
  99 conv    512  3 x 3 / 1  26 x  26 x 256 ->  26 x  26 x 512 1.595 BF
 100 conv     18  1 x 1 / 1  26 x  26 x 512 ->  26 x  26 x  24 0.012 BF
 101 yolo
 102 route  98
 103 conv    128  1 x 1 / 1  26 x  26 x 256 ->  26 x  26 x 128 0.044 BF
 104 upsample            2x  26 x  26 x 128 ->  52 x  52 x 128
 105 route  104 36
 106 conv    128  1 x 1 / 1  52 x  52 x 384 ->  52 x  52 x 128 0.266 BF
 107 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
 108 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
 109 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
 110 conv    128  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x 128 0.177 BF
 111 conv    256  3 x 3 / 1  52 x  52 x 128 ->  52 x  52 x 256 1.595 BF
 112 conv     18  1 x 1 / 1  52 x  52 x 256 ->  52 x  52 x  24 0.025 BF
 113 yolo
```

KEMENTERIAN RISET, TEKNOLOGI, DAN PENDIDIKAN TINGGI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
FAKULTAS TEKNOLOGI ELEKTRO
DEPARTEMEN TEKNIK ELEKTRO
Gedung B, C & AJ Kampus ITS Sukolilo, Surabaya 60111
Telp. (031) 5947302, 5994251-55 (Ext.1206, 1239) Fax. (031) 5931237
Email: elits@ee.its.ac.id ; www.ee.its.ac.id

Nomor : 2827 /IT2.VI.3.1/PN.01.00.00/2019 02 APR 2019
Lampiran : --
Perihal : **Permohonan Ijin Penelitian Dan Bantuan Data**

Yth. : Kasubdit. Analisis Data dan Pengembangan Sitem,
Direktorat Pengembangan Jaringan Jalan,
Ditjen Bina Marga, Kementrian PUPR
Jl. Patimura 20, Kebayoran Baru Jakarta Selatan,12110

Dengan hormat,

Dengan ini kami hadapkan mahasiswa Program Pascasarjana Departemen Teknik Elektro FTE-ITS:

Nama	: Ernin Niswatul Ukhwah
NRP	: 0711175006007
Program Studi	: Magister Teknik Elektro
Bidang Keahlian	: Telematika (PETIK)

Mahasiswa tersebut sedang mengerjakan Tesis dengan judul:

"Deteksi Kerusakan jalan berdasarkan GLCM Menggunakan Backpropagation Neural Network".

Sehubungan dengan hal tersebut diatas, kami mohon bantuan agar mahasiswa di atas dapat di berikan kesempatan untuk melakukan penelitian/pencarian data dan informasi di instansi yang Bapak/Ibu Pimpin.
Untuk data yang diambil adalah:

- Foto/Video Survei Jalan

Atas perhatian dan kerjasama yang baik, kami ucapkan terima kasih.

Kepala,

Dr. Ardyono Priyadi, ST., M.Eng.
NIP. 19730927 199803 1 004