

**IMPLEMENTASI *DEEP LEARNING OBJECT DETECTION*
RAMBU K3 PADA VIDEO MENGGUNAKAN METODE
CONVOLUTIONAL NEURAL NETWORK (CNN) DENGAN
*TENSORFLOW***

(Studi Kasus : Rambu Kesehatan dan Keselamatan Kerja (K3) Jalur Evakuasi dan
Alat Pemadam Api pada Gedung FMIPA UII)

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Jurusan



Syinta Nuri Mashita

16611128

**PROGRAM STUDI STATISTIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2020**



Handwritten signature

HALAMAN PENGESAHAN**TUGAS AKHIR****IMPLEMENTASI *DEEP LEARNING OBJECT DETECTION* RAMBU K3
PADA VIDEO MENGGUNAKAN METODE *CONVOLUTIONAL NEURAL
NETWORK (CNN)* DENGAN *TENSORFLOW***

Nama Mahasiswa : Syinta Nuri Mashita

NIM : 16611128

**TUGAS AKHIR INI TELAH DIUJIKAN
PADA TANGGAL: 20 Juli 2020**

Nama Penguji:

Tanda Tangan

1. Dr. Raden Bagus Fajriya Hakim, S.Si., M.Si.

2. Tuti Purwaningsih, S.Stat., M.Si.

3. Mujiati Dwi Kartikasari, S.Si., M.Sc.

Mengetahui,

Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam



(Prof. Riyanto, S.Pd., M.Si., Ph.D.)

KATA PENGANTAR



Assalamu'alaikum Wr.Wb

Dengan menyebut nama Allah SWT yang Maha Pengasih lagi Maha Panyayang, Kami panjatkan puja dan puji syukur kehadiran-Nya, yang telah melimpahkan rahmat, hidayah, dan inayah-Nya kepada penulis, sehingga penulis dapat menyelesaikan Tugas Akhir atau Skripsi sebagai syarat dalam mendapat gelar Sarjana Statistika.

Dalam penyelesaian Tugas Akhir dengan judul : **“Implementasi *Deep Learning Object Detection* Rambu K3 pada Video menggunakan Metode *Convolutional Neural Network (CNN)* dengan *Tensorflow*”** ini, tidak lepas dari banyak pihak yang telah memberi bantuan dan dukungan berupa saran, kritik, bimbingan, maupun bantuan lainnya. Untuk itu, pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Bapak Prof. Riyanto, S.Pd., M.Si., Ph.D. selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia.
2. Bapak Dr. Edy Widodo, S.Si., M.Si., selaku Ketua Program Studi Statistika Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia.
3. Ibu Mujiati Dwi Kartikasari, S.Si., M.Sc., selaku dosen pembimbing Tugas Akhir yang dengan sabar telah meluangkan waktu untuk membimbing dan mengarahkan penulis dalam penyusunan Tugas Akhir ini.
4. Seluruh dosen Program Studi Statistika yang telah memberikan arahan, bantuan, bimbingan serta ilmu di dalam maupun di luar lingkungan perkuliahan.
5. Keluarga yaitu Bapak Muhari, Ibu Nurhidayati, dan Adik Amalia Nuri Ma'rifah yang selalu mendukung dan memberikan doa yang tiada henti serta memberi motivasi kepada penulis untuk menyelesaikan tugas akhir.
6. Sahabat-sahabat tercinta, Erzylia Herlin Brilliant, Barlinda Titania, dan Maudi Mirqoatul Mafa'atih yang selalu ada, selalu mengingatkan kebaikan

DAFTAR ISI

HALAMAN JUDUL	i
HALAMAN PERSETUJUAN PEMBIMBING.....	ii
HALAMAN PENGESAHAN	iii
KATA PENGANTAR	iv
DAFTAR ISI	vi
DAFTAR TABEL	ix
DAFTAR GAMBAR.....	x
DAFTAR LAMPIRAN.....	xiii
PERNYATAAN	xvii
ABSTRAK	xviii
ABSTRACT	xix
BAB I PENDAHULUAN	1
1.1 Latar Belakang Masalah.....	1
1.2 Rumusan Masalah.....	4
1.3 Batasan Masalah	5
1.4 Tujuan Penelitian	5
1.5 Manfaat Penelitian	6
BAB II TINJAUAN PUSTAKA	7
BAB III LANDASAN TEORI.....	11
3.1 Kesehatan dan Keselamatan Kerja (K3)	11
3.2 Rambu Kesehatan dan Keselamatan Kerja (K3)	12
3.3 Manfaat Rambu K3.....	12
3.4 Kategori Rambu K3	13
3.5 Contoh objek rambu K3	14
3.5.1 Rambu Jalur Evakuasi	14
3.5.2 Rambu Alat Pemadam Api	15
3.6 Citra.....	16
3.6.1 Citra Digital	17
3.6.2 Jenis Citra.....	19

3.6.3	Resolusi Citra	23
3.7	Cara Kerja Kamera	24
3.8	<i>Computer Vision</i>	26
3.8.1	Image Processing (Pengolahan Citra).....	27
3.8.2	Pattern Recognition (Pengenalan Pola)	27
3.9	<i>Object Detection</i>	28
3.10	<i>Artificial Intelligence</i>	29
3.11	<i>Machine Learning</i>	30
3.12	<i>Deep Learning</i>	31
3.13	<i>Artificial Neural Network</i>	32
3.13.1	Arsitektur Jaringan Artificial Neural Network.....	33
3.13.2	Backpropagation.....	35
3.14	<i>Convolutional Neural Network</i>	36
3.14.1	Convolutional Layer	38
3.14.2	Pooling Layer	40
3.14.3	Fungsi Aktivasi.....	40
3.14.4	Stride	42
3.14.5	Padding	42
3.14.6	Fully Connected Layer.....	43
3.14.7	Dropout	43
3.14.8	Softmax Classifier	44
3.14.9	Loss Function	44
3.15	<i>Integral Image</i>	44
3.16	<i>Confusion Matrix</i>	47
3.17	Tingkat Akurasi	48
3.18	<i>Python</i>	49
3.19	<i>Tensorflow</i>	49
3.20	<i>Single Shot Multibox Detector (SSD)</i>	50
3.21	<i>Mobilenet V1</i>	52
BAB IV METODE PENELITIAN		53
4.1	Populasi dan Sampel	53

4.2	Variabel dan Definisi Operasional Variabel Penelitian	53
4.3	Metode Pengumpulan Data	53
4.4	Perangkat Penelitian.....	54
4.5	Metode Penelitian	55
4.6	Diagram Alir Penelitian	56
BAB V PEMBAHASAN		60
5.1	Rancangan Sistem.....	60
5.2	Pembuatan Dataset.....	61
5.3	Rancangan <i>Output</i>	63
5.4	<i>Preprocessing</i> Data.....	64
5.4.1	Pelabelan Gambar.....	64
5.4.2	Konversi Dataset XML ke csv	65
5.4.3	Konversi Dataset csv ke TFRecord	66
5.4.4	Konfigurasi Label Map	66
5.5	Konfigurasi <i>Object Detection Training Pipeline</i>	67
5.6	Arsitektur Jaringan.....	67
5.6.1	Proses pada Convolutional Layer	69
5.6.2	Fungsi Aktivasi.....	70
5.6.3	Pooling Layer	71
5.6.4	Fully Connected Layer.....	72
5.7	<i>Training Model</i>	74
5.8	Model Hasil <i>Training</i>	75
5.8.1	Hasil Training Steps	75
5.8.2	Export Model.....	76
5.8.3	Total Loss.....	77
5.9	Hasil Pendeteksian Objek pada Video	78
BAB VI PENUTUP		81
6.1	Kesimpulan.....	81
6.2	Saran.....	81
DAFTAR PUSTAKA		83
LAMPIRAN		89

DAFTAR TABEL

Tabel 4.1 Definisi Operasional Variabel	53
Tabel 4.2 Spesifikasi <i>Iphone 6S plus</i>	54
Tabel 4.3 Spesifikasi Lenovo Ideapad C340	55
Tabel 4.4 Perangkat Lunak yang Digunakan.....	55
Tabel 5.1. Pembagian Data <i>Train</i> dan Data <i>Test</i>	62



DAFTAR GAMBAR

Gambar 3.1 warna rambu K3.....	12
Gambar 3.2 Kategori Rambu K3	14
Gambar 3.3 Rambu Jalur Evakuasi	15
Gambar 3.4 Rambu Alat Pemadam Api	16
Gambar 3.5 Koordinat Citra Digital (Putra D. , 2010).....	17
Gambar 3.6 Representasi Citra Digital.....	18
Gambar 3.7 Pixel Citra Biner.....	19
Gambar 3.8 Ilustrasi Citra Biner	20
Gambar 3.9 Palet <i>grayscale</i> pada nilai bagian <i>Red</i> , <i>Green</i> , dan <i>Blue</i>	20
Gambar 3.10 Ilustrasi Citra Grayscale	21
Gambar 3.11 Koordinat warna RGB (Pamungkas, 2016).....	22
Gambar 3.12 Ilustrasi Citra Warna	23
Gambar 3.13 Resolusi Pixel (Putra D. , 2010).....	24
Gambar 3.14 <i>Preview</i> kamera DSLR	24
Gambar 3.15 <i>Capture</i> Kamera DSLR	25
Gambar 3.16 <i>Object Detection</i> (Tanner, 2019).....	29
Gambar 3.17 Hubungan antara <i>Human Vision</i> , <i>Computer Vision</i> , <i>Machine Learning</i> , <i>Deep Learning</i> , dan CNN (Khan, Rahmani, Shah, & Bennamoun, 2018).....	32
Gambar 3.18 <i>Single Layer Neural Network</i>	34
Gambar 3.19 <i>Multi Layer Neural Network</i>	35
Gambar 3.20 Arsitektur Jaringan <i>Backpropagation</i>	35
Gambar 3.21 Ilustrasi Arsitektur CNN (Matlab)	36
Gambar 3.22 Arsitektur <i>Convolutional Neural Network</i>	37
Gambar 3.23 Proses konvolusi.....	38
Gambar 3.24 Proses konvolusi.....	39
Gambar 3.25 Proses <i>Max Pooling Layer</i>	40
Gambar 3.26 Neural Network sebelum dropout	43

Gambar 3.27 Neural Network setelah dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014)	44
Gambar 3.28 Visualisasi Citra Masukan	45
Gambar 3.29 Matriks Original	45
Gambar 3.30 Matriks Original	46
Gambar 3.31 Matriks Original	46
Gambar 3.32 Integral Image	46
Gambar 3.32 Area <i>Rectangle</i>	47
Gambar 3.34 Integral	47
Gambar 3.35 Tabel <i>Confusion Matrix</i>	48
Gambar 3.36 (a) Input yang dibutuhkan untuk <i>training</i> pada SSD berupa kotak untuk masing-masing objek (b) Visualisasi <i>feature map</i> berukuran 8 x 8 (c) <i>Feature map</i> 4 x 4 berupa detail dari <i>feature map</i> 8 x 8 (Liu, 2016)	51
Gambar 3.37 Arsitektur SSD	51
Gambar 4.1 <i>Flowchart</i> Alur Penelitian	56
Gambar 5.1 Visualisasi Arsitektur Jaringan	60
Gambar 5.2 Dataset Jalur Evakuasi	61
Gambar 5.3 Dataset Alat Pemadam Api	62
Gambar 5.4 Rancangan <i>Output</i> Rambu Jalur Evakuasi	63
Gambar 5.5 Rancangan <i>Output</i> Rambu Alat Pemadam Api	63
Gambar 5.6 Pemberian label rambu alat pemadam api	64
Gambar 5.7 Pemberian label rambu jalur evakuasi	64
Gambar 5.8 Perintah menjalankan konversi <i>XML</i> ke <i>csv</i>	65
Gambar 5.9 Hasil konversi file <i>csv</i>	65
Gambar 5.10 Perintah menjalankan <i>script</i> konversi <i>csv</i> ke <i>TFRecord</i>	66
Gambar 5.11 Konfigurasi Label Map	67
Gambar 5.12 Jaringan <i>VGG-16 Net</i>	68
Gambar 5.13 Matriks Sampel Kode Warna	68
Gambar 5.14 Ilustrasi Perhitungan pada <i>Convolutional Layer</i>	69
Gambar 5.15 Ilustrasi Pergerakan Kernel	70
Gambar 5.16 Ilustrasi Operasi ReLU	71

Gambar 5.17 Proses <i>Pooling Layer</i>	71
Gambar 5.18 Proses <i>Flattening</i>	72
Gambar 5.19 Ilustrasi <i>Fully Connected Layer</i>	73
Gambar 5.20 Perintah <i>Training Model</i>	74
Gambar 5.21 Perintah untuk melihat <i>Tensorboard</i>	74
Gambar 5.22 Tampilan Awal <i>Tensorboard</i>	75
Gambar 5.23 <i>Log Training Step Proses</i>	75
Gambar 5.24 Grafik <i>Global Training Step</i>	76
Gambar 5.25 Perintah <i>Export Model</i>	77
Gambar 5.26 Model Hasil Training	77
Gambar 5.27 Grafik <i>Total Loss</i>	78
Gambar 5.28 Perintah uji coba model	78
Gambar 5.29 Hasil Deteksi Rambu Jalur Evakuasi pada Video Pertama	79
Gambar 5.30 Hasil Deteksi Rambu Alat Pemadam Api pada Video Pertama	79
Gambar 5.31 Hasil Deteksi Rambu Jalur Evakuasi pada Video Kedua.....	80
Gambar 5.32 Hasil Deteksi Rambu Alat Pemadam Api pada Video Kedua.....	80

DAFTAR LAMPIRAN

Lampiran 1 Hasil Perhitungan pada <i>Convolutional Layer</i>	89
Lampiran 2 Hasil Perhitungan pada <i>Pooling Layer</i>	91
Lampiran 3 Proses Pelabelan gambar	92
Lampiran 4 <i>Script</i> Konversi Dataset <i>XML</i> ke <i>CSV</i>	93
Lampiran 5 <i>Script</i> Konversi Dataset <i>CSV</i> ke <i>TFRecord</i>	94
Lampiran 6 <i>Script</i> Konfigurasi Label Map	96
Lampiran 7 <i>Script</i> Konfigurasi <i>Pipeline</i>	97
Lampiran 8 <i>Script Training</i>	101
Lampiran 9 <i>Script Export Graph Model</i>	105
Lampiran 10 <i>Script</i> untuk mendeteksi rambu K3 pada video	107



DAFTAR ISTILAH

<i>Activation Map</i>	: Output hasil perhitungan <i>feature map</i> dengan kernel
Akurasi	: Tingkat kedekatan hasil pengukuran terhadap nilai yang sebenarnya
<i>Batch</i>	: Set data yang dimasukkan dalam satu iterasi <i>training model</i>
<i>Batch Normalization</i>	: Normalisasi input atau output dari fungsi aktivasi dalam <i>hidden layer</i> yang memiliki fungsi untuk membuat jaringan neural lebih stabil dengan melindunginya terhadap bobot outlier, memungkinkan peningkatan kecepatan <i>learning</i> , mengurangi <i>overfitting</i> .
<i>Class/Label</i>	: Variabel atau atribut yang digunakan dalam penelitian
<i>Gradient Descent</i>	: Algoritma untuk mengoptimalkan iterasi atau meminimalkan <i>error</i> yang digunakan pada <i>Machine Learning</i> untuk menemukan hasil yang terbaik
<i>Learning Rate</i>	: Parameter dari <i>Gradient Descent</i>
<i>Loss Function</i>	: Nilai kerugian yang diperoleh dari proses pelatihan
<i>Epoch</i>	: Ketika seluruh <i>dataset</i> sudah melalui proses pelatihan pada <i>Neural Network</i> sampai dikembalikan ke awal untuk sekali putaran
<i>Batch Size</i>	: Jumlah sampel data yang disebarkan ke <i>neural network</i> atau ukuran dari satuan kecil Epoch yang dimasukkan (<i>feeding</i>) kedalam komputer
<i>Feeding</i>	: Meng-input data dengan <i>Tensorflow Record</i> (TFRecord) pada API <i>Tensorflow</i>
<i>Feature Extraction</i>	: Teknik pengambilan ciri / <i>feature</i> dari suatu bentuk yang nantinya nilai yang didapatkan akan dianalisis untuk proses selanjutnya
<i>Feature Map</i>	: Input yang digunakan pada sebuah layer yang merupakan

	hasil dari perhitungan layer sebelumnya
<i>Framework</i>	: Sebuah <i>software</i> yang dapat memudahkan dalam membuat sebuah aplikasi atau proses perhitungan untuk membentuk suatu sistem tertentu agar tersusun dan terstruktur secara rapi
<i>Stride</i>	: Parameter yang menentukan berapa jumlah pergeseran <i>filter/kernel</i>
<i>Padding</i>	: Parameter yang menentukan jumlah piksel yang berisi nilai nol yang akan ditambahkan disetiap sisi <i>input</i>
<i>Dropout</i>	: Teknik regulasi jaringan saraf dimana beberapa neuron akan dipilih secara acak dan tidak dipakai selama proses pelatihan
<i>Step</i>	: Sejumlah langkah yang didefinisikan pada konfigurasi <i>pipeline</i> untuk proses pelatihan yang menentukan tingkat keberhasilan pelatihan <i>neural network</i>
<i>Filter/Kernel</i>	: Matriks untuk menghitung dan mendeteksi suatu pola atau ciri yang digunakan untuk perhitungan <i>convolution</i>
<i>Checkpoint</i>	: Berkas yang dihasilkan dari proses pelatihan yang disimpan dalam format <i>.chkp</i>
<i>Neuron</i>	: Dapat disebut juga dengan node/unit yaitu simpul dalam <i>neural network</i> yang biasanya menggunakan beberapa nilai <i>input</i> dan menghasilkan satu nilai output
<i>Model</i>	: Representasi dari apa yang telah dipelajari oleh sistem <i>Machine Learning</i> dari data <i>training</i>
<i>Object Detection</i>	: Proses memindai dan mencari objek dalam gambar atau video
<i>Overfitting</i>	: Suatu kondisi dimana model yang dibangun memperhitungkan seluruh ciri yang ada, termasuk <i>noise</i>

<i>Parameter</i>	: Variabel dalam model yang dilatih oleh sistem <i>machine learning</i> . Contohnya ialah bobot, bobot merupakan parameter yang nilainya dipelajari secara bertahap oleh sistem <i>machine learning</i> melalui iterasi pelatihan berurutan
<i>Pipeline</i>	: Infrastruktur yang berkaitan dengan algoritma <i>Machine Learning</i> yang meliputi proses mengumpulkan data, memasukkan data kedalam file data pelatihan, melatih satu atau beberapa model, dan mengeksport model
<i>Central Processing Unit (CPU)</i>	: Perangkat keras komputer yang memahami dan menjalankan instruksi dari perangkat lunak
<i>Graphics Processing Unit (GPU)</i>	: <i>Processor</i> yang bertugas memproses instruksi oleh sebuah perangkat lunak dengan tujuan khusus yaitu pengolahan grafis
<i>Loss</i>	: Ukuran seberapa jauh prediksi model tidak sesuai dengan label yang sebenarnya atau dengan kata lain ukuran seberapa buruk model
<i>Tensorboard</i>	: Dashboard yang menampilkan ringkasan yang disimpan selama satu atau beberapa program <i>TensorFlow</i>

ABSTRAK

IMPLEMENTASI *DEEP LEARNING OBJECT DETECTION* RAMBU K3 PADA VIDEO MENGGUNAKAN METODE *CONVOLUTIONAL NEURAL NETWORK (CNN)* DENGAN *TENSORFLOW*

(Studi Kasus : Rambu Kesehatan dan Keselamatan Kerja (K3) Jalur Evakuasi dan
Alat Pemadam Api pada Gedung FMIPA UII)

Syinta Nuri Mashita

Program Studi Statistika, Fakultas MIPA

Universitas Islam Indonesia

Kesehatan dan Keselamatan Kerja (K3) merupakan pengawasan terhadap mesin, orang, metode, dan material yang berada di dalam lingkungan kerja agar pekerja tersebut tidak mengalami cedera maupun penyakit akibat kerja. Salah satu penunjang agar terciptanya keselamatan kerja pada lingkungan kerja adalah dengan menyadari dan mematuhi rambu-rambu K3. Rambu-rambu K3 adalah tanda-tanda yang dipasang di tempat kerja, guna mengingatkan atau mengidentifikasi pada semua pelaksana kegiatan atau pekerja disekeliling tempat tersebut terhadap kondisi, resiko, yang terkait dengan keselamatan dan kesehatan kerja. *International labor Organization* (ILO) secara tegas memberikan perlindungan terhadap hak atas jaminan keselamatan dan kesehatan di dunia kerja. Hak atas K3 berlaku bagi semua tenaga kerja, termasuk tenaga kerja peyandang disabilitas. Pentingnya K3 dalam lingkungan kerja tidak lepas dari sarana dan prasarana yang menunjang hal tersebut. Berdasarkan peraturan pemerintah dalam setiap bangunan gedung wajib menyediakan sarana evakuasi meliputi sistem peringatan bahaya, pintu keluar darurat, dan jalur evakuasi. Maka dari itu untuk mempermudah pengenalan dan petunjuk arah keberadaan Rambu K3 diperlukannya sistem yang dapat mengenali dan mendeteksi rambu-rambu K3. Pengembangan cabang ilmu kecerdasan buatan (*Artificial Intelligence*), salah satunya adalah *computer vision* yaitu *object detection*. *Deep learning* yang digunakan untuk pengenalan dan klasifikasi objek adalah *Convolutional Neural Network* karena menghasilkan hasil yang signifikan dalam pengenalan citra. Pada Penelitian ini dilakukan pendeteksian objek Rambu K3 Jalur Evakuasi dan Alat pemadam Api dengan menggunakan *framework Tensorflow* dan model SSD (*Single Shot Multibox Detector*). Dataset yang digunakan dalam penelitian ini sebanyak 1500 data citra. Hasil penelitian deteksi objek rambu K3 pada video menggunakan metode CNN didapatkan tingkat akurasi berkisar antara 50%-97%.

Kata Kunci: *Artificial Intelligence, Convolutional Neural Network, Object Detection*, Rambu Kesehatan dan Keselamatan Kerja (K3), *Tensorflow*

ABSTRACT

IMPLEMENTATION OF DEEP LEARNING OBJECT DETECTION OHS SIGN ON VIDEO USING CONVOLUTIONAL NEURAL NETWORK (CNN) METHOD WITH TENSORFLOW

*(Case study: Occupational Health and Safety signs (OHS) evacuation and fire
extinguisher in building FMIPA UII)*

Syinta Nuri Mashita

Departement of Statistics

Faculty of Mathematics and Natural Science

Islamic University of Indonesia

Occupational health and safety (OHS) is the supervision of machines, people, methods and materials in the work environment so that the worker does not experience work injuries or illnesses. One of the supports for the creation of work safety in the work environment is to realize and comply with OHS guidelines. OHS signs are signs posted in the workplace, to remind or identify all implementers of activities or workers around the place of conditions, risks, related to occupational safety and health. The International Labor Organization (ILO) expressly protects the right to safety and health in the world of work. The right to OHS applies to all workers, including workers with disabilities. The importance of OHS in the work environment is inseparable from the facilities and infrastructure that support this. Based on government regulations in each building, buildings are required to provide evacuation facilities, including hazard warning systems, emergency exits and evacuation routes. Therefore, to facilitate the introduction and direction of the existence of OHS Signs, a system that can recognize and detect OHS signs is needed. Development of the branch of artificial intelligence (Artificial Intelligence), one of which is computer vision, namely object detection. Deep learning used for object recognition and classification is Convolutional Neural Network because it produces significant results in image recognition. In this research, the detection of OHS Signs of Evacuation Paths and Fire Extinguishers is done by using the Tensorflow framework and SSD (Single Shot Multibox Detector) models. The dataset used in this study was 1500 image data. The results of the detection of OHS sign object detection on video using the CNN method obtained an accuracy rate ranging from 50% -97%.

Keywords: Artificial Intelligence, Convolutional Neural Network, Object Detection, Signs of Occupational Health and Safety (OHS), Tensorflow

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Indonesia yang terletak di *Ring of Fires* dengan diapit tiga lempeng dunia aktif menjadi potensi bencana alam gempa bumi yang bisa terjadi sewaktu-waktu, membuat masyarakat Indonesia perlu memiliki kecakapan dalam menghadapi bencana alam. Tidak hanya siaga menghadapi gempa bumi, bencana lain seperti banjir, gunung meletus, kebakaran hingga tsunami, penting bagi masyarakat memiliki pengetahuan dan kesadaran kesiapsiagaan (Rizka, 2020). Bencana dapat disebabkan oleh kejadian alam maupun oleh ulah manusia.

Salah satu Provinsi yang telah dan memiliki banyak potensi bencana adalah Provinsi D.I. Yogyakarta, sebanyak 12 dari 13 potensi bencana yang disebutkan BNPB diantaranya letusan gunung api, tanah longsor dan erosi, banjir, kekeringan, tsunami, angin kencang, gelombang ekstrim dan abrasi, gempa bumi, epidemic dan wabah penyakit, kebakaran, konflik sosial, dan Kegagalan Teknologi. Dari beberapa bencana yang telah disebutkan, letusan gunung api dan gempa bumi merupakan bencana dengan tingkat potensi yang cukup besar pada Kabupaten Sleman.

Gedung Fakultas Matematika dan Ilmu Pengetahuan Alam yang berada di Jalan Kaliurang, Kecamatan Ngaglik, Kabupaten Sleman merupakan salah satu tempat kerja berupa gedung bertingkat yang ada di Universitas Islam Indonesia. Dalam UU No.1 Tahun 1970, yang dimaksud dengan tempat kerja adalah tiap ruangan atau lapangan, tertutup atau terbuka, bergerak atau tetap, tempat tenaga kerja bekerja, atau yang sering dimasuki tenaga kerja untuk keperluan suatu usaha dan terdapat sumber-sumber bahaya. Tempat kerja memiliki potensi terjadinya kecelakaan kerja. Beberapa bencana juga dapat disebut sebagai kecelakaan kerja. Kecelakaan kerja adalah suatu kejadian atau peristiwa yang tidak diinginkan yang merugikan terhadap manusia, merusak harta benda atau kerugian terhadap proses (Sugandi, 2003).

Menurut menteri tenaga kerja Ida Fauzi, berdasarkan data BPJS Ketenagakerjaan tahun 2018 terjadi kecelakaan kerja sebanyak 114.148 kasus sedangkan tahun 2019 terdapat 77.295 kasus (Tanjung, 2020). Hal ini menunjukkan terdapat penurunan kecelakaan yang terjadi di tempat kerja sebesar 33,05%. Namun, angka kecelakaan kerja tersebut tetap menunjukkan masih banyaknya kecelakaan kerja yang terjadi di Indonesia. Beberapa aspek keselamatan kerja perlu diamati untuk mencegah terjadinya kecelakaan kerja diantaranya meliputi kesehatan dan keamanan kerja para pekerja, konstruksi dan dampak lingkungan sekitar yang ditimbulkan untuk para pekerja.

Berdasarkan Undang Undang No. 14 Tahun 1969 Pasal 9, K3 merupakan pengawasan terhadap mesin, orang, metode, dan material yang berada di dalam lingkungan kerja agar pekerja tersebut tidak mengalami cedera maupun penyakit akibat kerja, karena tiap tenaga kerja berhak mendapatkan perlindungan terutama keselamatan, kesehatan, kesusilaan, pemeliharaan moral, serta mendapatkan perlakuan yang sesuai dengan moral agama dan martabat manusia. *International labor Organization* (ILO) yang secara tegas memberikan perlindungan terhadap hak atas pekerjaan, jelas hal ini melandasi terpenuhinya hak atas pekerjaan dan serta memberikan kepastian atas jaminan keselamatan dan kesehatan di dunia kerja. Hak atas K3 berlaku bagi semua tenaga kerja, termasuk tenaga kerja peyandang disabilitas.

Indonesia merupakan negara berkembang yang saat ini sedang giat-giatnya melakukan pembangunan, baik dalam bidang infrastruktur maupun dalam bidang industri. Hasil pembangunan tersebut berupa gedung-gedung bertingkat yang masih sering menyepelkan prinsip – prinsip K3. Sangat disayangkan para tenaga kerja masih banyak yang belum paham bahkan belum tahu mengenai rambu-rambu K3. Rambu-rambu K3, sangat penting guna meminimalisir kecelakaan kerja ataupun bahaya yang akan ditimbulkan saat bekerja. Hal ini tercantum dalam UU No.1 Tahun 1970 pasal 14b yang berbunyi “Memasang dalam tempat kerja yang dipimpinnya, semua gambar keselamatan kerja yang diwajibkan dan semua bahan pembinaan lainnya, pada tempat-tempat yang mudah dilihat dan terbaca menurut petunjuk pegawai pengawas atau ahli keselamatan kerja” dan

juga diatur dalam Permenaker No. 05/MEN/1996 tentang Sistem Manajemen Keselamatan dan Kesehatan Kerja Kriteria audit 6.4.4. yang berbunyi “ Rambu-rambu mengenai keselamatan dan tanpa pintu darurat harus dipasang sesuai dengan standar dan pedoman”. Rambu-rambu keselamatan dan kesehatan kerja adalah tanda-tanda yang dipasang di tempat kerja/laboratorium, guna mengingatkan atau mengidentifikasi pada semua pelaksana kegiatan disekeliling tempat tersebut terhadap kondisi, resiko, yang terkait dengan keselamatan dan kesehatan kerja.

Beberapa ancaman dalam keselamatan kerja tidak lepas dari sering terjadinya bencana alam di Indonesia. Menyikapi isu aktual yakni kebencanaan yang sering terjadi di Indonesia, menurut Kepala Badan Pengkajian dan Penerapan Teknologi (BPPT) Dr. Hammam Riza, M. Sc. diperlukan adanya langkah mitigasi dan evakuasi bencana yang berbasis teknologi. Seiring berkembangnya teknologi yang semakin pesat, hal ini terlihat hampir semua aspek kehidupan manusia memiliki hubungan yang kuat dengan teknologi. BPPT telah melakukan berbagai rekayasa teknologi guna reduksi risiko bencana. Perkembangan teknologi tidak lepas dari perkembangan banyaknya data yang ada, yaitu berupa data gambar maupun video. Dalam proses mitigasi bencana telah direncanakan penerapan *computer vision*, yang mana dengan adanya data gambar dan video tersebut dapat digunakan untuk mendapatkan berbagai macam informasi yang berguna seperti mendeteksi, mengklasifikasikan, dan melacak objek untuk memahami peristiwa yang terjadi pada dunia nyata.

Pada hakikatnya, *computer vision* bertujuan untuk meniru cara kerja visual manusia atau yang disebut dengan *human vision*. Dalam prosesnya *human vision* sebenarnya sangat kompleks yaitu diawali dari manusia melihat objek dengan indera penglihatan, kemudian objek citra tersebut diteruskan ke otak sehingga objek tersebut diinterpretasi yang mana pada akhirnya manusia dapat mengerti objek apa yang terlihat dalam pandangan mata manusia. Hasil interpretasi tersebut dapat digunakan dalam mengambil suatu keputusan. (Kusumanto & Tomponu, 2011). Adapun didalam *computer vision* terdapat beberapa penerapan seperti *image classification* dan *object detection* (Dewi, 2018). *Object detection* kini

masih terus berkembang dalam meniru kemampuan manusia untuk dapat melihat dan memahami objek yang terlihat oleh indera penglihatannya sehingga komputer dapat memiliki kemampuan selayaknya manusia. Salah satu metode yang digunakan dalam pendeteksian objek adalah *Convolutional Neural Network (CNN)*.

Convolutional Neural Network (CNN) termasuk dalam jenis *deep learning* karena kedalaman jaringannya. *Deep learning* adalah cabang dari *machine learning* yang dapat mengajarkan komputer untuk melakukan pekerjaan selayaknya manusia, seperti komputer dapat belajar dari proses *training*. CNN merupakan salah satu metode *deep learning* yang mampu melakukan otomatisasi terhadap ekstraksi fitur. Metode ini banyak digunakan dalam melakukan ekstraksi citra/gambar, ekstraksi tersebut menghasilkan sebuah informasi yang kemudian dilakukan pengolahan informasi, salah satunya adalah melakukan klasifikasi citra dengan tingkat akurasi yang baik. Sehingga dalam perkembangannya dapat membentuk sebuah model CNN yang dapat di gunakan untuk mendeteksi suatu objek.

Berdasarkan uraian diatas, dalam penelitian ini peneliti menerapkan *Convolutional Neural Network (CNN)* dalam pembuatan sebuah model untuk mendeteksi rambu-rambu K3 yang mana kedepannya pendeteksian objek rambu-rambu K3 tersebut dapat dijadikan langkah awal dalam sistem pendeteksian objek khususnya objek rambu K3 sehingga dapat diterapkan dan dikembangkan oleh perusahaan untuk mendeteksi rambu-rambu K3 bagi seluruh tenaga kerja.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas rumusan masalah penelitian ini adalah:

1. Bagaimana implementasi metode *deep learning* menggunakan *Convolutional Neural Network (CNN)* untuk mendeteksi rambu K3 pada video?
2. Bagaimana model yang terbentuk dari hasil pelatihan pada deteksi rambu K3 menggunakan *Convolutional Neural Network (CNN)*?

3. Bagaimana tingkat akurasi yang dihasilkan untuk mendeteksi rambu K3 pada sebuah video menggunakan model hasil pelatihan jaringan *Convolutional Neural Network* (CNN)?

1.3 Batasan Masalah

Penelitian ini memiliki beberapa batasan agar lebih spesifik dan terarah, maka diberikan batasan-batasan sebagai berikut :

1. *Software* yang digunakan adalah *Python* dengan *Framework Tensorflow*.
2. Pada penelitian ini data diambil secara primer yaitu peneliti mengambil data dengan memotret objek yang akan dideteksi pada pukul 10.00 - 12.00 selama 3 hari dengan estimasi jarak pengambilan antara 2 – 10 meter dari objek menggunakan kamera smartphone dengan ukuran citra digital 1:1 yang beresolusi 12 *Mega Pixel* (MP).
3. Data yang digunakan adalah rambu-rambu keselamatan dan kesehatan kerja (K3) yang ada pada gedung Fakultas Matematika dan Ilmu Pengetahuan Alam (FMIPA) UIL.
4. Terdapat dua kelas rambu K3 untuk pendeteksian objek yaitu, Jalur Evakuasi dan Alat Pemadam Api.
5. Jumlah data citra yang diolah sebanyak 1500 dengan masing-masing kelas berjumlah 750.
6. Output yang dihasilkan ialah label berupa teks, tingkat akurasi serta bidang segiempat pada sebuah video yang menandakan perbedaan kelas.

1.4 Tujuan Penelitian

Berdasarkan rumusan masalah di atas, tujuan penelitian ini adalah:

1. Mengetahui implementasi metode *deep learning* menggunakan *Convolutional Neural Network* (CNN) untuk mendeteksi rambu K3 pada video.
2. Mengetahui model yang terbentuk dari hasil pelatihan pada deteksi rambu K3 menggunakan *Convolutional Neural Network* (CNN).
3. Mengetahui tingkat akurasi yang dihasilkan untuk mendeteksi rambu K3 pada sebuah video menggunakan model hasil pelatihan jaringan *Convolutional Neural Network* (CNN).

1.5 Manfaat Penelitian

Adapun manfaat yang diharapkan dari penelitian ini diantaranya sebagai berikut :

1. Penelitian ini diharapkan dapat memberikan pengetahuan mengenai implementasi *deep learning* menggunakan *Convolutional Neural Network* dengan bantuan sistem komputer dan teknologi, untuk mengatasi permasalahan khususnya pada bidang teknologi dan mitigasi bencana.
2. Mengetahui tingkat akurasi dari implementasi *Convolutional Neural Network* untuk dijadikan acuan pengembangan penelitian berikutnya.
3. Proses dan hasil yang dipaparkan dapat berguna sebagai langkah awal penerapan *Artificial Intelligence* dalam bidang mitigasi sehingga dapat dikembangkan untuk pembuatan aplikasi yang dapat bermanfaat untuk menunjukkan keberadaan rambu-rambu K3 agar dapat berguna sebagai petunjuk ketika dalam keadaan darurat khususnya bagi penyandang disabilitas.
4. Penelitian ini dapat digunakan sebagai acuan pada penelitian selanjutnya yang serupa.

BAB II

TINJAUAN PUSTAKA

Penelitian-penelitian terdahulu baik yang memiliki tema maupun tentang penggunaan metode yang memiliki keterkaitan dengan penelitian ini menjadi acuan penulis guna mengetahui hubungan dengan penelitian terdahulu dan menghindari unsur duplikasi dengan penelitian sebelumnya. Penelitian ini juga mengkaji dari beberapa penelitian yang bersumber dari referensi skripsi maupun jurnal yang berhubungan sehingga penelitian ini dapat memberikan kontribusi terhadap perkembangan ilmu pengetahuan. Berikut ini merupakan penelitian terdahulu yang berkaitan dengan data maupun metode yang digunakan.

Penelitian yang telah dipublikasikan oleh I Wayan Suartika E.P., dkk pada tahun 2016 pada Jurnal Teknik ITS yang berjudul “Klasifikasi Citra ,enggunakan *Convolutional Neural Network* pada *Caltech 101*” merupakan penelitian dengan metode *deep learning* yaitu *Convolutional Neural Network*. Penelitian ini menggunakan basis data *Caltech 101* yang bertujuan untuk mengklasifikasikan unggas dengan kategori Emu, Flamingo, Ibis, Pigeon, dan Rooster yang memiliki 150 citra. Selain itu, dihasilkan pula 3 kategori lainnya yaitu *Cougar*, *Crocodile*, dan *Face*. Hasil klasifikasi dengan menggunakan *Convolutional Neural Network* cukup handal untuk menentukan klasifikasi citra objek yang dibuktikan dengan tingkat akurasi sebesar 20% - 50%.

Pada tahun 2016 terdapat penelitian dengan penerapan *Deep Learning* dengan menggunakan *Convolutional Neural Network* yang dilakukan oleh Muhammad Zufar dan Budi Setiyono dengan judul “*Convolutional Neural Network* untuk Pengenalan Wajah secara *Real-Time*”. Metode ini diimplementasikan dengan *Library OpenCV* untuk deteksi wajah pada perangkat *Web Cam M-Tech 5 MP*. Dataset yang digunakan yaitu himpunan gambar wajah yang dibagi menjadi dua jenis himpunan yaitu himpunan wajah *indoor* (kondisi pencahayaan gelap) dan himpunan wajah *outdoor* (kondisi pencahayaan terang). Hasil uji coba dengan menggunakan konstruksi model CNN sampai kedalaman 7 lapisan dengan input dari hasil ekstraksi Extended Local Binary Pattern dengan

radius 1 dan neighbor 15 menunjukkan kinerja pengenalan wajah meraih rata-rata akurasi lebih dari 89% dalam 2 frame perdetik. Penelitian ini menunjukkan bahwa implementasi dari model CNN dapat di terapkan pada proses pengenalan wajah secara *real-time* dengan akurasi yang cukup tinggi.

Pada tahun 2018 Fadwa Al-Azzoa, Arwa Mohammed Taqia dan Mariofanna Milanovab dalam jurnalnya yang berjudul “*Human Related-Health Actions Detection using Android Camera based on TensorFlow Object Detection API*” melakukan sebuah penelitian dengan video aksi terkait kesehatan manusia menggunakan teknik API deteksi objek *TensorFlow*. Mereka menggunakan dua model pra-training baru (*Faster RCNN-Resnet* dan *SSD-Mobilenet*) yang telah diterapkan untuk pelatihan dataset mengenai tindakan manusia. Presisi rata-rata (AP) adalah rata-rata prediksi kelas yang diperkirakan pada beberapa ambang batas. Akurasi deteksi (mAP) pada 0,5IoU adalah nilai tinggi dengan banyaknya *step* yang berbeda. Hal ini disebabkan oleh fakta bahwa jaringan berhubungan dengan gambar video, oleh karena itu perlu waktu lama untuk melatih seluruh sampel dari setiap *frame* untuk setiap tindakan dan tergantung pada jenis arsitektur model. Dari hasil penelitian ini, tindakan atau gerakan terkait kesehatan dapat dideteksi dengan deteksi kecepatan tinggi dan akurasinya yang luar biasa. Ini bisa mengetahui tindakan yang benar yang diperlukan untuk menangani situasi yang sesuai menggunakan kamera ponsel. Sebuah model detektor baru dibangun untuk tujuh aksi video terkait kesehatan manusia yang berbeda menggunakan dua teknik API deteksi objek *TensorFlow*, yaitu *notebook* pendeteksi objek *TensorFlow* dan *TensorFlow* di Android, menggunakan kamera ponsel. Selain itu, model deteksi HHRA dilatih dan dievaluasi menggunakan *dataset* NTU RGB + D berdasarkan dua model pra-training (*FasterR-CNN-Resnet* dan *SSD-Mobilenet*). Didapatkan hasil dalam kategori terbaik yang dapat dideteksi, total mAP mencapai 93,8% untuk *F-R-CNN-Resnet* lebih cepat dan 95,8% untuk *SSD-Mobilenet*.

Salah satu penelitian yang menerapkan algoritma *Convolutional Neural Network* pada implementasi *object detection* yaitu penelitian dengan judul “*Deep Learning Object Detection* pada Video menggunakan *Tensorflow* dan

Convolutional Neural Network” yang dilakukan oleh Syarifah Rosita Dewi (Dewi, 2018). Penelitian ini menyelesaikan masalah *object detection* pada *computer vision* dengan menggunakan algoritma *Convolutional Neural Network* dan *framework Tensorflow*. Objek yang dideteksi pada penelitian ini adalah objek meja dan kursi motif ukiran Jepara dengan dataset sebanyak 500 gambar. Model yang terbentuk dari hasil training dengan jumlah steps 250.000 dan batch size sebanyak 2 dapat mendeteksi citra meja dan kursi motif ukiran Jepara dengan menghasilkan tingkat akurasi berkisar antara 70% hingga 99%

Penelitian yang dilakukan oleh Ervin Milos, Aliaksei Kolesau dan Dmitriy Sesok (2018) yang berjudul “*Traffic Sign Recognition Using Convolutional Neural Networks*” membahas tentang masalah pengenalan rambu lalu lintas menggunakan *Convolutional Neural Networks (CNN)*, metode tersebut dapat diadopsi karena kinerjanya yang terbukti dengan baik untuk aplikasi *computer vision*. Penelitian ini mengusulkan *Histogram Equalization Preprocessing (HOG)* dan CNN dengan operasi tambahan - *batch normalization*, *dropout* dan augmentasi data. Dalam penelitian ini dipilih enam model arsitektur CNN untuk membandingkan nilai akurasi dari klasifikasi *training* model. Data yang digunakan ialah *German Traffic Sign Recognition Benchmark (GTSRB)* dengan 43 kategori dan terdapat 39.000 data *training* dan 12.000 data *testing* atau dengan perbandingan sekitar 75:25. Hasil tingkat akurasi model yang dihasilkan sangat tinggi yaitu mencapai 99,24%.

Penelitian tentang CNN sebelumnya telah dilakukan oleh Rizky Dwi Novyantika pada tahun 2018 dengan judul “Deteksi Tanda Nomor Kendaraan Bermotor Pada Media Streaming Dengan Algoritma *Convolutional Neural Network* Menggunakan *Tensorflow*”. Dalam penelitian ini, peneliti menggunakan data citra tanda nomor kendaraan bermotor dan pada proses trainingnya penelitian ini menggunakan 25.000 step dan batch size 8. Dari penelitian yang dilakukan menghasilkan pendeteksi TNKB dengan algoritma *Convolutional Neural Network* pada media streaming dengan nilai akurasi yang tinggi yaitu antara 70 - 99%.

Arwa Mohammed Taqi, Fadwa Al-Azzo, Ahmed Awad dan Mariofanna Milanova (2019) mempublikasikan penelitian pada *American Journal of*

Advanced Research yang berjudul “*Skin Lesion Detection by Android Camera based on SSD- Mobilenet and TensorFlow Object Detection API*”. Data yang digunakan diambil dari *International Skin Imaging Collaboration “ISIC 2018”*. Dalam penelitian ini, lesi kanker kulit telah terdeteksi menggunakan sistem API deteksi objek *TensorFlow*. Kesimpulannya, sistem detektor baru telah diselesaikan untuk lesi kanker kulit menggunakan dua metode yaitu *TensorFlow notebook* deteksi objek API, dan *TensorFlow* di Android menggunakan kamera ponsel. Model deteksi di-*training* dan diestimasi berdasarkan arsitektur *SSD-Mobilenet*. Menurut hasil, total mAP mencapai 96,04%, dan kesalahan terendah dihitung melalui kehilangan klasifikasi dan lokalisasi dan itu kurang dari 0,5.

Penelitian-penelitian di atas merupakan penelitian terdahulu yang menggunakan metode *Convolutional Neural Network* (CNN), *Tenserflow*, dan atau model SSD. Penelitian yang dilakukan oleh I wayan Suartika (2016) merupakan penerapan CNN untuk klasifikasi gambar. Sedangkan Muhammad Zuffar dan Budi Setiyono menggunakan algoritma CNN untuk mendeteksi wajah. Fadwa Al-Azzoa, dkk (2018) menerapkan CNN dengan membandingkan model *Faster-R-CNN* dan model SSD, didapatkan hasil kategori terbaik yaitu menggunakan *SSD-mobilenet*. Penelitian Ervin Milos dkk, dengan enam model arsitektur CNN untuk membandingkan nilai akurasi dari klasifikasi *training* model. Syarifah Rosita Dewi, Rizky Dwi Novyantika, Arwa Mohammed Taqi dkk melakukan penelitian dengan menerapkan algoritma CNN dengan *Tensorflow* dan *SSD-mobilenet*. Dari ketujuh penelitian terdahulu yang telah menggunakan algoritma CNN dengan model *SSD-mobilenet* hampir sama dengan penelitian yang akan dilakukan oleh peneliti. Namun, objek rambu-rambu K3 belum terdapat pada penelitian sebelumnya.

BAB III

LANDASAN TEORI

3.1 Kesehatan dan Keselamatan Kerja (K3)

Menurut OHSAS 18001:2007 mendefinisikan Keselamatan dan Kesehatan Kerja sebagai kondisi dan faktor yang mempengaruhi atau akan mempengaruhi keselamatan dan kesehatan pekerja dan juga tamu atau orang lain berada di tempat kerja. K3 adalah suatu sistem yang dirancang untuk menjamin keselamatan yang baik pada semua personel di tempat kerja agar tidak menderita luka maupun menyebabkan penyakit di tempat kerja dengan mematuhi atau taat pada hukum dan aturan keselamatan dan kesehatan kerja, yang tercermin pada perubahan sikap menuju keselamatan di tempat kerja (G, Diah, & Zen, 2017). K3 adalah kegiatan yang menjamin terciptanya kondisi kerja yang aman, terhindar dari gangguan fisik dan mental melalui pembinaan dan pelatihan, pengarahan, dan kontrol terhadap pelaksanaan tugas dari para karyawan dan pemberian bantuan sesuai dengan aturan yang berlaku, baik dari lembaga pemerintah maupun perusahaan dimana mereka bekerja (Yuli, 2005).

K3 dijadikan sebagai aspek perlindungan tenaga kerja sekaligus melindungi aset perusahaan yang bertujuan sebaik mungkin memberikan jaminan kondisi yang aman dan sehat kepada setiap karyawan dan untuk melindungi Sumber Daya Manusia (SDM). Kesehatan dan Keselamatan Kerja bertujuan untuk mengurangi angka kecelakaan kerja khususnya di Indonesia. Hal ini dapat tercapai apabila perusahaan selalu memperhatikan faktor keselamatan dan kesehatan kerja karena hal ini akan meningkatkan kinerja karyawan. Perhatian terhadap kesehatan pekerja pada mulanya lebih menekankan pada masalah keselamatan kerja yaitu perlindungan pekerjaan dari kerugian atau luka yang disebabkan oleh kecelakaan berkaitan dengan kerja. Kemudian seiring dengan perkembangan industri, perusahaan mulai memperhatikan kesehatan pekerja dalam arti luas yaitu terbebasnya pekerjaan dari kesakitan fisik maupun psikis (Mondy, Wayne, & Robert, 2005).

3.2 Rambu Kesehatan dan Keselamatan Kerja (K3)

Rambu-rambu K3 merupakan tanda-tanda yang dipasang di tempat kerja dan laboratorium, guna mengingatkan atau mengidentifikasi pada semua pelaksana kegiatan disekeliling tempat tersebut terhadap kondisi, resiko, yang terkait dengan keselamatan dan kesehatan kerja. Rambu K3 merupakan salah satu cara untuk menginformasikan kepada para pekerja tentang bahaya-bahaya keselamatan dan kesehatan kerja dari sesuatu aktivitas, area atau peralatan kerja tertentu. Sehingga, dengan adanya rambu K3 tersebut setiap orang baik pekerja, tamu, dan kontraktor dapat mengantisipasi sedini mungkin tentang bahaya-bahaya di area tersebut dan juga meminimalisir risiko. Rambu-rambu K3 dikelompokkan menjadi beberapa kategori berdasarkan warnanya.

Warna Keselamatan	Warna Kontras (Simbol atau Tulisan)	Makna
MERAH	PUTIH	Larangan Pemadam Api
KUNING	HITAM	Perhatian / Waspada Potensi Beresiko Bahaya
HIJAU	PUTIH	Zona Aman Pertolongan Pertama
BIRU	PUTIH	Wajib Ditaati
PUTIH	HITAM	Informasi Umum

Gambar 3.1 warna rambu K3

Sumber : (Rambu - rambu K3, 2015)

3.3 Manfaat Rambu K3

Beberapa manfaat dari pemasangan rambu-rambu K3 bagi tenaga kerja maupun pengunjung suatu kantor atau gedung :

1. Menyediakan kejelasan informasi dan memberikan pengarahan secara umum
2. Memberikan penjelasan mengenai kesehatan dan keselamatan kerja

3. Menunjukkan adanya potensi bahaya yang mungkin tidak terlihat
4. Mengingatkan para tenaga kerja dimana harus menggunakan peralatan perlindungan diri sebelum memulai aktifitas di tempat kerja
5. Menunjukkan dimana peralatan darurat keselamatan berada
6. Memberikan peringatan waspada terhadap beberapa tindakan atau perilaku yang tidak diperbolehkan

3.4 Kategori Rambu K3

Rambu-rambu K3 penting untuk ditaati dan dipatuhi agar kita semua terhindar dari kecelakaan. Berikut ini beberapa kategori dari rambu K3.

1. Rambu Larangan

Rambu ini adalah rambu yang memberikan larangan yang wajib ditaati kepada siapa saja yang ada di lingkungan itu, tanpa ada pengecualian. Adapun larangan yang harus ditaati adalah sesuai dengan rambu gambar atau informasi yang terpasang.

2. Rambu Peringatan

Rambu ini adalah rambu yang memberikan peringatan yang perlu diperhatikan kepada siapa saja yang ada di lingkungan itu karena dapat mengakibatkan kejadian yang tidak diinginkan. Adapun Peringatan yang perlu diikuti adalah sesuai dengan rambu gambar atau informasi yang terpasang.

3. Rambu Prasyarat

Rambu ini adalah rambu yang memberikan persyaratan dilaksanakan kepada siapa saja yang ada di lingkungan itu karena prasyarat tersebut merupakan kewajiban yang harus dilaksanakan. Adapun Prasyarat yang perlu dilaksanakan adalah sesuai dengan rambu tergambar atau informasi yang terpasang.

4. Rambu Pertolongan

Rambu ini adalah rambu yang memberikan bantuan/pertolongan serta arah yang ada di lingkungan itu karena arah pertolongan tersebut merupakan petunjuk arah yang harus diikuti siapa saja terutama bila terjadi kondisi

darurat. Adapun rambu pertolongan atau petunjuk arah tersebut dipasang pada tempat yang strategis dan mudah terlihat dengan jelas.

Berdasarkan kategori yang telah disebutkan di atas, berikut ini adalah bentuk dari setiap kategori rambu.

BENTUK DASAR (KELOMPOK)	ARTI	PENJELASAN
	Bentuk Bulat, dasar warna putih, lingkaran merah, dengan garis 45° miring dari kiri atas ke bawah, logo hitam	Tanda Larangan Contoh: 
	Bentuk Bulat, dasar warna Biru, lingkaran putih, logo atau keterangan gambar warna putih	Tanda Wajib / prasyarat Contoh : 
	Bentuk segitiga, dasar warna kuning garis hitam, dengan logo / gambar warna hitam	Tanda Waspada / Contoh : peringatan 
	Bentuk segi empat, dasar warna hijau, garis luar putih, logo / gambar putih	Tanda pertolongan / Contoh : Arah penyelamatan 

Gambar 3.2 Kategori Rambu K3

Sumber : (Rambu - rambu K3, 2015)

3.5 Contoh objek rambu K3

3.5.1 Rambu Jalur Evakuasi

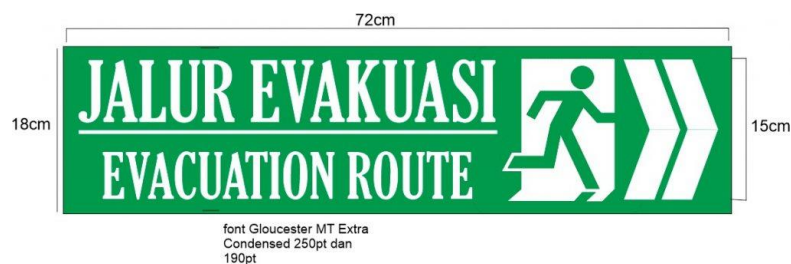
Salah satu peraturan yang menjadi dasar kewajiban pengadaan rambu jalur evakuasi tertuang pada Undang-undang No. 28 tahun 2002 tentang Bangunan Gedung dan juga Peraturan Pemerintah No. 36 tahun 2005 tentang Bangunan Gedung. PP No 36 tahun 2005 tentang bangunan gedung menyatakan bahwa “Setiap bangunan gedung, kecuali rumah tinggal tunggal dan rumah deret sederhana, harus menyediakan sarana evakuasi yang meliputi sistem peringatan bahaya bagi pengguna, pintu keluar darurat, dan jalur evakuasi yang dapat menjamin kemudahan pengguna bangunan gedung untuk melakukan evakuasi

dari dalam bangunan gedung secara aman apabila terjadi bencana atau keadaan darurat”.

Adapun beberapa kriteria dari pemasangan jalur evakuasi yaitu sebagai berikut.

- Jalur Evakuasi harus memiliki akses langsung ke jalan atau ruang terbuka yang aman, dilengkapi penanda yang jelas dan mudah terlihat.
- Jalur Evakuasi dilengkapi penerangan yang cukup.
- Jalur Evakuasi bebas dari benda yang mudah terbakar atau benda yang dapat membahayakan.
- Jalur Evakuasi bersih dari orang atau barang yang dapat menghalangi gerak, tidak melewati ruang yang dapat dikunci.
- Jalur Evakuasi memiliki lebar minimal 71.1 cm dan tinggi langit-langit minimal 230 cm.
- Pintu Darurat dapat dibuka ke luar, searah Jalur Evakuasi menuju Titik Kumpul, bisa dibuka dengan mudah, bahkan dalam keadaan panik.
- Pintu Darurat dilengkapi dengan penutup pintu otomatis.
- Pintu Darurat dicat dengan warna mencolok dan berbeda dengan bagian bangunan yang lain

Pada gambar 3.3 di bawah ini merupakan contoh rambu jalur evakuasi beserta dengan ukurannya.



Gambar 3.3 Rambu Jalur Evakuasi

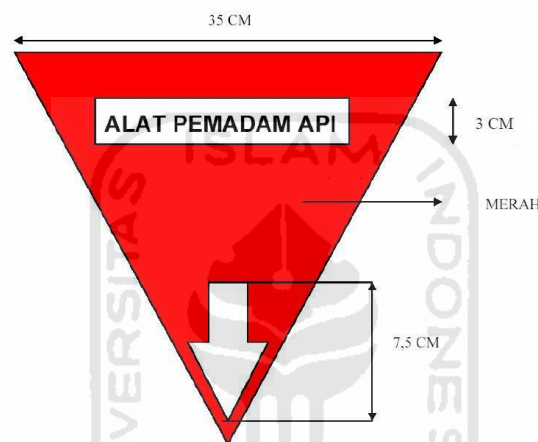
3.5.2 Rambu Alat Pemadam Api

Rambu Alat Pemadam Api adalah salah satu rambu keselamatan yang bertujuan untuk menunjukkan dimana letak alat pemadam api ringan. Dalam

Permenakertrans RI No 4/MEN/1980, ada beberapa poin yang ditekankan terkait tanda APAR (Alat Pemadam Api Ringan):

- Tanda APAR harus berbentuk segitiga sama sisi dengan warna dasar merah
- Ukuran tiap sisi dari segitiga tersebut adalah 35 cm
- Tinggi huruf yang ada pada tanda APAR setinggi 3 cm dan berwarna putih
- Ukuran tanda panah pada tanda APAR setinggi 7,5 cm dengan warna putih

Gambar 3.4 di bawah ini merupakan contoh rambu alat pemadam api beserta ukurannya.



Gambar 3.4 Rambu Alat Pemadam Api

3.6 Citra

Menurut Sutojo (2004) citra adalah sebuah representasi, atau bayangan visual dari suatu objek. Berdasarkan sifatnya, citra dapat dibagi menjadi dua kategori, yaitu citra yang bersifat analog dan digital. Citra analog adalah citra yang bersifat kontinu seperti gambar pada monitor televisi, foto sinar X, hasil *CT Scan*, dan lain-lain. Sedangkan pada citra digital adalah citra yang dapat diolah oleh komputer (Sutoyo & dkk, 2009).

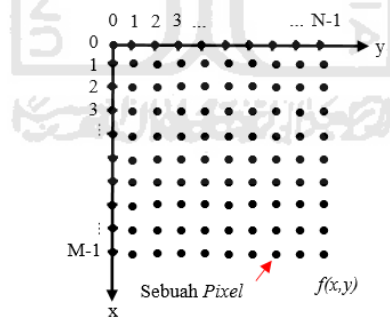
Citra merupakan gambar pada dua dimensi. Citra tersusun dari jumlah pixel yang sangat banyak yang merupakan bagian terkecil dari citra. Secara garis besar citra terbentuk dari kotak-kotak berbentuk persegi empat yang disebut pixel, jarak horizontal dan vertikal antara pixel adalah sama. Citra atau *image* biasanya ditulis dalam koordinat cartesian (x,y) dan setiap kordinat merepresentasikan satu bagian

terkecil dari objek yaitu pixel. Nilai yang terdapat pada koordinat (x,y) adalah $f(x,y)$, yaitu besar intensitas atau warna dari pixel di titik itu (Munir, 2004).

3.6.1 Citra Digital

Citra digital adalah citra yang dapat diolah oleh komputer. Umumnya citra digital berbentuk persegi panjang atau bujur sangkar yang memiliki lebar dan tinggi tertentu (Fadlisyah, 2007). Ukuran ini biasanya dinyatakan dalam banyaknya titik atau pixel sehingga ukuran citra selalu bernilai bulat. Setiap titik memiliki koordinat sesuai posisinya dalam citra. Setiap titik juga memiliki nilai berupa angka digital yang merepresentasikan informasi yang diwakili oleh titik tersebut.

Suatu citra dapat didefinisikan sebagai fungsi $f(x,y)$ berukuran M baris dan N kolom, dengan x dan y adalah koordinat spasial, dan amplitudo f di titik koordinat (x,y) dinamakan intensitas atau tingkat keabuan dari citra pada titik tersebut. Apabila nilai x , y , dan nilai amplitudo f secara keseluruhan berhingga dan bernilai diskrit maka dapat dikatakan bahwa citra tersebut citra digital. Gambar 3.5 menunjukkan posisi koordinat citra digital (Nafi'iyah, 2015).



Gambar 3.5 Koordinat Citra Digital (Putra D. , 2010)

Citra digital adalah suatu matriks dimana indeks baris dan kolom menyatakan suatu titik pada suatu citra. Citra digital terdiri atas beberapa elemen dimana setiap elemen merupakan elemen citra atau pixel yang menyatakan nilai derajat keabuan pada titik tersebut. Citra digital berukuran $M \times N$ (baris = M , kolom = N) dinyatakan dengan matriks $M \times N$. Adapun bentuk matriks citra digital seperti dibawah ini (Basuki, 2016).

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,M-1) \\ f(1,0) & f(1,1) & \cdots & f(1,M-1) \\ \vdots & \vdots & \vdots & \vdots \\ f(N-1,0) & f(N-1,1) & \cdots & f(N-1,M-1) \end{bmatrix} \quad (3.1)$$

Berdasarkan rumus di atas, suatu citra $f(x,y)$ dapat dituliskan kedalam fungsi matematis seperti berikut ini :

$$0 \leq x \leq M-1$$

$$0 \leq y \leq N-1$$

$$0 \leq f(x,y) \leq G-1$$

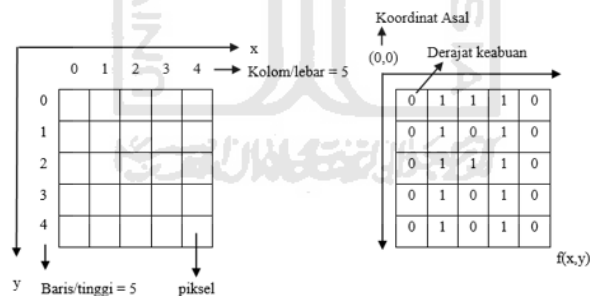
Keterangan :

N : Jumlah pixel kolom pada array citra

M : Jumlah pixel baris pada array citra

G : Nilai skala keabuan (*grayscale*)

Besarnya nilai G tergantung pada proses digitalisasinya. Biasanya keabuan 0 (nol) menyatakan intensitas hitam dan 1 (satu) menyatakan intensitas putih. Untuk citra 8 bit, nilai G sama dengan $2^8 = 256$ warna (derajat keabuan). Berikut gambar 3.6 di bawah ini yang merupakan representasi citra digital 2 dimensi.



Gambar 3.6 Representasi Citra Digital

Dari gambar 3.6 di atas, terdapat nilai pada suatu irisan antara baris dan kolom (x,y) disebut dengan pixel (*picture element*), *image elements* atau pels. Citra digital pada umumnya memiliki bentuk persegi panjang dengan lebar dan panjang tertentu. Adapun ukuran citra diukur dalam jumlah titik atau pixel yang mana setiap pixel memiliki koordinat menurut letaknya didalam citra digital. Dalam masing-masing pixel memiliki nilai yang dapat mewakili informasi yang ada didalam pixel.

3.6.2 Jenis Citra

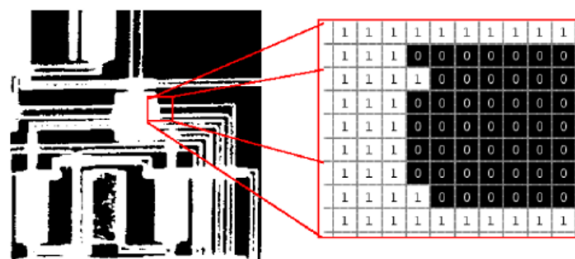
Suatu pixel memiliki nilai dalam rentang tertentu yaitu mulai dari nilai minimum hingga nilai maksimum. Jangkauan yang digunakan berbeda, tergantung dari jenis warnanya. Namun secara umum untuk citra 8-bit memiliki jangkauan warna 0 – 255. Citra dengan penggambaran seperti ini digolongkan ke dalam citra integer. Berikut kategori jenis-jenis citra berdasarkan nilai pixel-nya (Darma, 2010).

1. Citra Biner

Citra Biner atau *Binary Image* merupakan citra yang pada masing-masing pixel hanya terdiri atas warna hitam dan warna putih. Pada citra biner hanya memerlukan satu bit per pixel yaitu 0 dan 1, hal ini dikarenakan hanya terdapat dua warna untuk setiap pixel-nya. Nilai 0 merupakan nilai yang melambangkan warna hitam sedangkan nilai 1 adalah nilai yang melambangkan warna putih. Citra yang direpresentasikan dengan biner sangat tepat untuk teks yang dicetak atau tulisan tangan (Kusumanto & Tompunu, 2011). Citra biner dinyatakan dalam suatu fungsi sebagai berikut:

$$f(x, y) = \{0, 1\} \quad (3.2)$$

Berikut ini merupakan contoh citra biner beserta dengan representasi nilai tiap pixel (Irianto, 2016) :



Gambar 3.7 Pixel Citra Biner

Gambar 3.8 di bawah ini merupakan salah satu contoh ilustrasi citra biner.



Gambar 3.8 Ilustrasi Citra Biner

2. Citra *Grayscale*

Citra keabuan adalah citra *grayscale* biasa dikenal sebagai citra : *intensity* atau *gray level*. Citra yang terdiri dari satu layer warna dengan derajat keabuan tertentu. Untuk kebanyakan citra digital 8 bit, maka sistem *grayscale* diukur berdasarkan skala intensitas kecerahan, yang bernilai 0 – 255, dimana yang berwarna hitam adalah 0 dan yang terputih adalah 255 (Munir, 2004). Citra *Grayscale* dinyatakan dalam fungsi berikut:

$$f(x, y) = \{0, \dots, 255\} \quad (3.3)$$

Tingkatan keabuan disini merupakan warna abu dengan berbagai tingkatan dari hitam hingga mendekati putih pada setiap nilai bagian *red*, *green*, dan *blue*. Pada gambar 3.9 di bawah ini menunjukkan warna *grayscale* pada citra 8 bit, yaitu dari warna hitam, keabuan dan putih pada setiap nilai bagian *red*, *green*, dan *blue*.



Gambar 3.9 Palet *grayscale* pada nilai bagian *Red*, *Green*, dan *Blue*

Citra *grayscale* mempunyai kemungkinan warna antara hitam (minimum) dan putih (maksimum). Contoh untuk skala *grayscale* 4 bit maka jumlah kemungkinan nilainya adalah $2^4 = 16$ (memiliki 16 warna), dan memiliki nilai maksimum $2^4 - 1 = 15$, sehingga memiliki kemungkinan warna 0 (minimum) sampai 15 (maksimum). Sedangkan untuk skala 8 bit,

maka jumlah kemungkinan warnanya adalah $2^8 = 256$, yaitu memiliki 256 warna dengan nilai maksimum $2^8 - 1 = 255$, sehingga memiliki kemungkinan warna 0 (minimum) sampai 255 (maksimum) (Balza & Kartika, 2005). Berikut merupakan ilustrasi dari citra *grayscale* :



Gambar 3.10 Ilustrasi Citra Grayscale

3. Citra Warna

Citra berwarna atau biasa dinamakan citra RGB, merupakan jenis citra yang menyajikan warna dalam bentuk komponen R(*Red*), G(*Green*), dan B(*Blue*). RGB adalah suatu model warna yang terdiri atas warna merah, hijau, dan biru. Apabila ketiga warna tersebut digabungkan maka akan membentuk suatu susunan warna yang luas (Kusumanto & Tompunu, 2011).

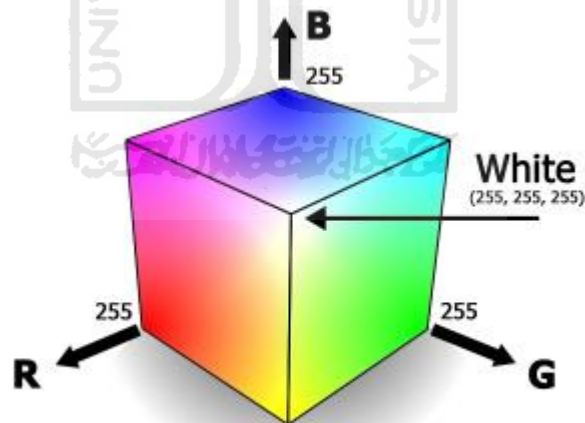
Setiap warna dasar mempunyai intensitas sendiri dengan nilai maksimum 255 (8 bit), dan warna minimum adalah putih. *Red* memiliki warna minimum putih dan warna maksimum merah. *Green* memiliki warna minimum putih dan warna maksimum hijau. *Blue* memiliki warna minimum putih dan warna maksimum biru. Misalnya warna kuning merupakan kombinasi warna merah dan hijau sehingga nilai RGB-nya adalah (255, 255, 0). Dengan demikian setiap titik (pixel) pada citra warna membutuhkan data 3 *byte* (Balza & Kartika, 2005).

Pemberian rentang nilai dapat diberikan pada setiap warna dasar. Dalam monitor komputer nilai rentang yang paling kecil adalah 0 dan yang paling besar adalah 255. Pemilihan skala 256 ini didasarkan pada cara

menampilkan 8 digit bilangan biner yang digunakan oleh mesin komputer. Dengan demikian, kemungkinan warna yang dapat disajikan mencapai $256 \times 256 \times 256$ atau 16,777,216 warna. Dalam prosesnya, dimisalkan pada satu jenis warna terdapat sebuah vektor dalam ruang tiga dimensi dalam matematika, yaitu komponen x, y dan z yang dituliskan dengan $r = (x, y, z)$, kemudian komponen tersebut diubah menjadi komponen RGB (Red, Green, Blue) sehingga satu jenis warna dapat dituliskan menjadi “warna = RGB (30, 255, 255)”, untuk warna putih yaitu “putih = (255, 255, 255)” sedangkan untuk warna hitam yaitu “hitam = RGB(0, 0, 0)”. Adapun representasi dalam citra digital dapat dinyatakan dalam persamaan sebagai berikut:

$$\begin{aligned} fR(x,y) &= \{0, \dots, 255\} \\ fG(x,y) &= \{0, \dots, 255\} \\ fB(x,y) &= \{0, \dots, 255\} \end{aligned} \quad (3.4)$$

Adapun koordinat dalam warna RGB beserta contoh ilustrasi citra warna adalah sebagai berikut :



Gambar 3.11 Koordinat warna RGB (Pamungkas, 2016)

Gambar 3.12 di bawah ini merupakan salah satu contoh ilustrasi citra berwarna.

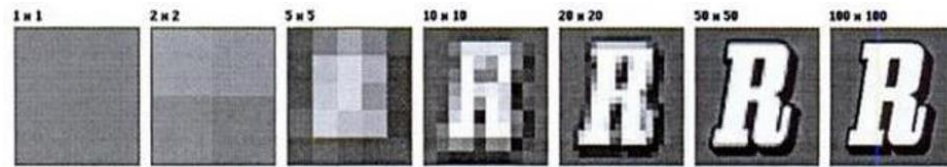


Gambar 3.12 Ilustrasi Citra Warna

3.6.3 Resolusi Citra

Resolusi citra adalah tingkatan detail dari suatu gambar. Semakin tinggi resolusi gambar maka akan membuat semakin tinggi juga tingkat detail dari gambar tersebut. Adapun satuan dalam pengukuran suatu resolusi citra dapat seperti ukuran fisik yaitu jumlah garis per millimeter atau jumlah garis per inch dan pengukuran juga dapat berupa ukuran citra secara menyeluruh yaitu dengan jumlah garis tertinggi citra. Resolusi pixel merupakan salah satu cara untuk mengukur resolusi sebuah citra (Putra D. , 2010).

Resolusi pixel adalah ukuran untuk perhitungan jumlah pixel dalam sebuah citra digital. Citra yang memiliki lebar M pixel dan tinggi N pixel akan memiliki ukuran resolusi sebesar $M \times N$. Resolusi pixel dapat juga didefinisikan dengan hasil perkalian antara jumlah pixel lebar dan tinggi, kemudian dibagi dengan angka satu juta. Sebagai contoh, sebuah citra yang memiliki lebar 3.508 pixel serta tinggi 2.480 pixel akan memiliki resolusi sebesar $3.508 \times 2.480 = 8.699.840$ pixel atau 8.7 megapixel. Berikut pada gambar 3.13 merupakan ilustrasi dari pixel dari sebuah objek 2 dimensi, dimana semakin besar nilai pixel-nya maka kualitas gambar semakin baik.



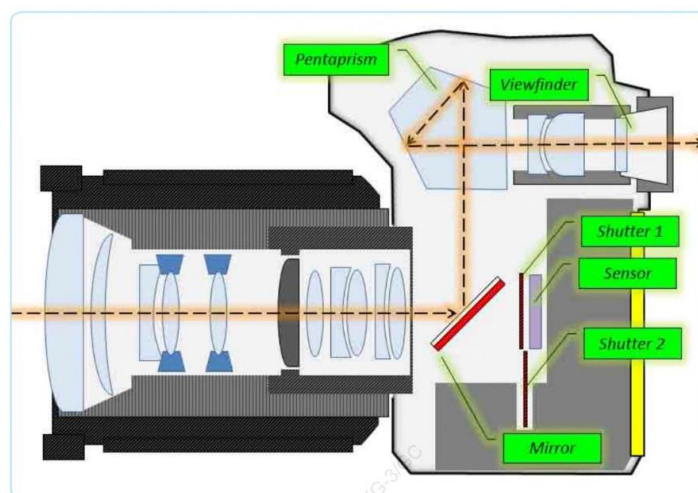
Gambar 3.13 Resolusi Pixel (Putra D. , 2010)

3.7 Cara Kerja Kamera

Kamera adalah alat yang digunakan untuk memotret atau mengambil gambar suatu benda atau objek dalam bentuk foto. Pada kamera juga dilengkapi dengan lensa yang berfungsi untuk mengumpulkan cahaya, maka dari itu kamera juga sering disebut sebagai alat lukis cahaya.

Pada dasarnya mekanisme kerja kamera ada 2 (dua) macam yaitu kamera film (ANALOG) dan kamera digital (DSLR). Perbedaan antara keduanya terletak pada media penyimpanan yang digunakan. Pada kamera analog saat tombol *shutter* dipencet pantulan cahaya dari objek akan masuk ke kamera melalui lensa. Lensa akan memfokuskan cahaya yang diterima berupa bayangan terbalik dan akan meneruskannya ke suatu media yang sangat peka cahaya (film). Sedangkan pada kamera digital terdapat sebuah sensor yang akan merekam semua informasi cahaya yang dikirim melalui lensa. Berikut merupakan langkah dasar cara kerja kamera DSLR:

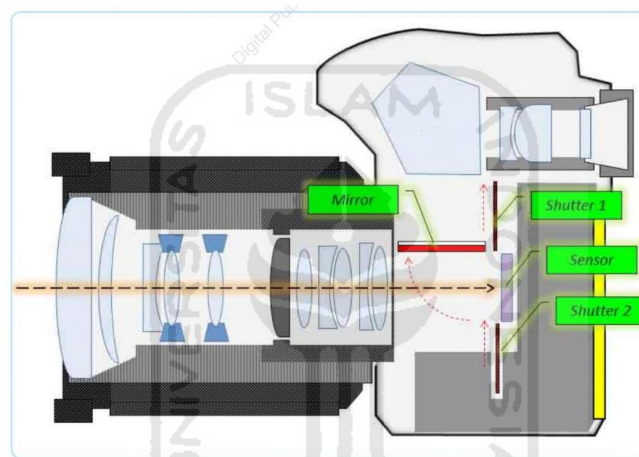
1. Tahap *Preview*



Gambar 3.14 *Preview* kamera DSLR

Pada tahap ini, cahaya yang berasal dari objek akan masuk melalui sisi depan lensa, melalui elemen-elemen optik dan pengaturan *aperture* (diafragma), untuk kemudian dipantulkan oleh cermin ke arah atas. Pantulan dari cermin tersebut mengarah menuju prisma atau cermin segilima (pentaprism atau pentamirror), untuk kemudian diteruskan ke *Viewfinder* (lubang incar). Tahap *Preview* digunakan untuk memperoleh gambaran sudut pandang yang akan dipotret, berikut pengaturan teknis lainnya seperti *exposure*, *metering*, *focus*, *white balance*, dan sebagainya.

2. Tahap *Capture*



Gambar 3.15 *Capture* Kamera DSLR

Tahap *Capture* sangat erat hubungannya dengan kerja mekanisme shutter (rana, atau juga penutup). Jenis yang paling umum dipergunakan adalah *Focal Plane Shutter* yang berbentuk bilah-bilah penutup horizontal (atau vertical, tergantung merek dan desain) yang membuka dan menutup sesuai pengaturan *Shutter Speed*. Kamera melalui tahap *Capture* dengan mekanisme yang menggerakkan cermin keatas, bersamaan dengan membukanya bilah penutup yang pertama (sering disebut sebagai *First Curtain*). Sebagai hasilnya, cahaya dari elemen lensa diteruskan langsung ke sensor selama beberapa saat, sampai *Shutter* kedua bergerak menutup sensor dan mengakhiri *exposure*. Citra yang ditangkap sensor kemudian diproses oleh *chip* menjadi data digital, dan akhirnya disimpan sebagai file foto pada media simpan.

Cara kerja kamera pada *smartphone* pada intinya mirip dengan kamera digital SLR. Tugas lensa merekam obyek bidikan dengan memanfaatkan intensitas cahaya yang ada. Karenanya cahaya amat penting untuk menghasilkan gambar yang prima. Cahaya dapat diatur, tetapi hal ini membutuhkan keahlian yang cukup. Perbedaan kamera DSLR dan kamera *smartphone* yaitu spesifikasi lensa pada *smartphone* tidak setinggi kamera DSLR, maka sebagian besar masalah akan muncul pada lensa, apertur, dan sensor sangat kecil. Akibatnya kurang bisa mendapatkan cahaya yang dibutuhkan untuk mendapatkan foto yang diinginkan. Berikut enam proses kerja sebuah kamera pada *smartphone*:

1. Mem-fokuskan lensa ke obyek
2. Cahaya memasuki lensa
3. Bukaan (*aperture*) menentukan intensitas cahaya yang memasuki sensor
4. Rana (*shutter*) menentukan berapa lama sensor terkena cahaya
5. Sensor menangkap gambar
6. *Hardware* memproses dan merekam gambar

3.8 Computer Vision

Vision secara bahasa dapat diartikan sebagai penglihatan. *Vision* juga dapat diartikan sebagai suatu proses pengamatan apa yang ada pada dunia nyata melalui panca indra penglihatan manusia. Adapun *computer vision* adalah suatu pembelajaran menganalisis gambar dan video untuk memperoleh hasil sebagaimana yang bisa dilakukan manusia. Pada hakikatnya, *computer vision* mencoba meniru cara kerja sistem visual manusia (*Human Vision*). Manusia melihat objek dengan indra penglihatan (mata), lalu citra objek diteruskan ke otak untuk diinterpretasi sehingga manusia mengerti objek apa yang tampak dalam pandangan matanya. Hasil interpretasi ini mungkin digunakan untuk pengambilan keputusan (misalnya menghindar kalau melihat mobil melaju di depan atau menghindar ketika ada pejalan kaki ketika sedang mengendarai sebuah mobil).

Secara garis besar, *computer vision* adalah sebuah teknologi yang digunakan oleh sebuah komputer untuk dapat melihat. Kemampuan untuk mengenali ini merupakan kombinasi dari pengolahan citra dan pengenalan pola. Pengolahan citra adalah sebuah proses dalam *computer vision* untuk

menghasilkan citra yang lebih baik dan atau mudah diinterpretasikan, sedangkan pengenalan pola adalah proses identifikasi objek pada sebuah citra.

Computer vision merupakan kombinasi antara *image processing* dan *pattern recognition*. *Computer vision* adalah pembangunan deskripsi objek fisik yang eksplisit dan gamblang dari sebuah gambar. *Output* dari *computer vision* adalah deskripsi atau interpretasi atau beberapa pengukuran kuantitatif struktur dalam adegan 3D (Le, 2015).

3.8.1 *Image Processing* (Pengolahan Citra)

Pengolahan citra digital adalah suatu disiplin ilmu yang mempelajari hal-hal yang berkaitan dengan perbaikan kualitas gambar (peningkatan kontras, transformasi warna, restorasi citra), transformasi gambar (rotasi, translasi, skala, transformasi geometrik), melakukan pemilihan citra ciri (*feature images*) yang optimal untuk tujuan analisis, melakukan proses penarikan informasi atau deskripsi objek atau pengenalan objek yang terkandung pada citra, melakukan kompresi atau reduksi data untuk tujuan penyimpanan data, transmisi data, dan waktu proses data (Sutoyo & dkk, 2009).

Pemrosesan gambar adalah metode untuk mengubah gambar menjadi bentuk digital dan melakukan beberapa proses operasi untuk mendapatkan beberapa informasi yang berguna dari gambar tersebut. Dimana input pada proses ini adalah gambar dan output mungkin gambar atau karakteristik yang terkait dengan gambar itu. (Jalled & Voronkov, 2016).

3.8.2 *Pattern Recognition* (Pengenalan Pola)

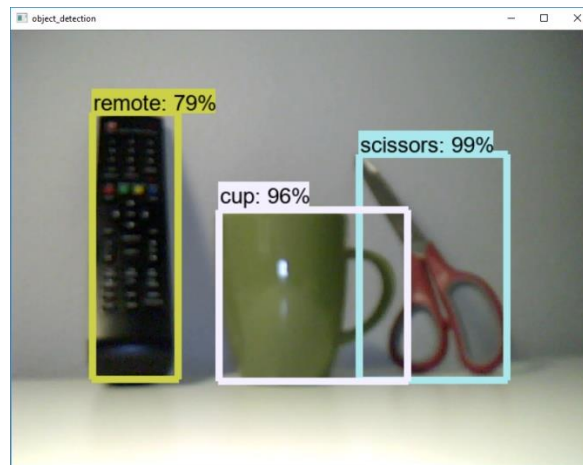
Pola adalah entitas yang terdefinisikan dan dapat diidentifikasi melalui ciri-cirinya (*features*). Ciri-ciri tersebut digunakan untuk membedakan suatu pola dengan pola lainnya. Ciri yang bagus adalah mempunyai daya pembeda yang tinggi, sehingga pengelompokan pola berdasarkan ciri yang dimiliki dapat dilakukan dengan keakuratan yang tinggi. Ciri pada suatu pola diperoleh dari hasil pengukuran terhadap objek uji. Pengenalan pola bertujuan menentukan kelompok atau kategori pola berdasarkan ciri-ciri yang dimiliki oleh pola tersebut. Dengan kata lain, pengenalan pola membedakan suatu objek dengan objek lain. Ada dua

fase dalam sistem pengenalan pola, yaitu fase pelatihan dan fase pengenalan. Pada fase pelatihan dipelajari untuk menentukan ciri yang akan digunakan dalam proses pengenalan serta prosedur klasifikasinya. Pada fase pengenalan, citra diambil cirinya kemudian ditentukan kelas kelompoknya (Syafitri, 2011).

Dalam mengklasifikasikan atau menggambarkan suatu citra dilakukan dengan proses pengukuran ciri atau sifat utama dari suatu objek. Bagian terpenting dari teknik pengenalan pola adalah bagaimana memperoleh informasi atau ciri penting yang terdapat dalam sinyal. Contoh dari pengenalan pola adalah sidik jari, raut wajah, gelombang suara, tulisan tangan dan lain sebagainya (Ardiyansyah, 2014).

3.9 *Object Detection*

Object Detection atau deteksi objek adalah prosedur menentukan keberadaan suatu objek dan atau ruang lingkungannya, serta lokasi pada gambar. Hal ini dapat diperlakukan sebagai pengenalan objek kelas dua, dimana satu kelas mewakili kelas objek dan kelas lain mewakili kelas non-objek. Deteksi objek dapat dibagi lagi menjadi *soft detection* dan *hard detection*. *Soft detection* hanya mendeteksi adanya objek sedangkan *hard detection* mendeteksi keberadaan dan lokasi objek. Bidang deteksi objek biasanya dilakukan dengan mencari setiap bagian dari gambar untuk melokalisasi bagian, yang fotometrik atau sifat geometrisnya cocok dengan objek target dalam *database* pelatihan. Hal ini dapat dilakukan dengan memindai *template* objek melintasi gambar di lokasi, skala, dan rotasi yang berbeda, dan deteksi dinyatakan jika kesamaan antara *template* dan gambar cukup tinggi. Kesamaan antara *template* dan wilayah gambar dapat diukur sesuai dengan korelasinya (Jalled & Voronkov, 2016). Berikut ini merupakan visualisasi hasil *object detection* pada beberapa objek.



Gambar 3.16 *Object Detection* (Tanner, 2019)

3.10 Artificial Intelligence

Artificial Intelligence (Kecerdasan Buatan) merupakan cabang pengetahuan dalam ilmu komputer yang dikenal dengan nama AI. Kecerdasan buatan yang biasa disingkat AI (*Artificial Intelligence*) merupakan ilmu tentang bagaimana membangun suatu sistem komputer yang menunjukkan kecerdasan dalam berbagai cara. Masalah inti kecerdasan buatan meliputi pemrograman komputer untuk sifat-sifat tertentu seperti pengetahuan, pemikiran, penyelesaian masalah, persepsi, pembelajaran, perencanaan, dan kemampuan dalam memanipulasi serta memindahkan objek. AI merupakan area penelitian yang dinamis dalam topik riset ilmu komputer. Sampai saat ini, telah banyak penelitian mengenai perkembangan AI diantaranya *neural network*, *evolutionary computing*, *machine learning*, *natural language processing*, pemrograman otomatis, robotika, dan *object oriented programming* (Yunanto Andhik Ampuh, 2016).

Artificial Intelligence (AI) menurut John McCarthy (1956) yang dikutip dari jurnal penelitian (Pannu, 2015) mengatakan bahwa AI bertujuan untuk mengetahui atau memodelkan proses berpikir manusia dan mendesain mesin sehingga bisa menirukan perilaku manusia. Cerdas, berarti memiliki pengetahuan ditambah pengalaman, penalaran, dan moral yang baik.

Manusia cerdas (pandai) dalam menyelesaikan permasalahan karena manusia mempunyai pengetahuan dan pengalaman. Pengetahuan diperoleh dari belajar. Semakin banyak bekal pengetahuan yang dimiliki tentu akan lebih

mampu menyelesaikan permasalahan. Tapi bekal pengetahuan saja belum cukup, manusia juga diberi akal untuk melakukan penalaran, mengambil kesimpulan berdasarkan pengetahuan dan pengalaman yang dimiliki. Tanpa memiliki kemampuan untuk menalar dengan baik, manusia dengan segudang pengalaman dan pengetahuan tidak akan dapat menyelesaikan masalah dengan baik. Demikian pula dengan kemampuan menalar yang sangat baik, namun tanpa pengetahuan dan pengalaman yang memadai, maka manusia juga tidak dapat menyelesaikan masalah (Dahria, 2008).

3.11 *Machine Learning*

Machine Learning atau pembelajaran mesin merupakan pendekatan dalam AI yang banyak digunakan untuk menggantikan atau menirukan perilaku manusia untuk menyelesaikan masalah atau melakukan otomatisasi. *Machine Learning* mencoba menirukan bagaimana proses manusia atau makhluk cerdas belajar dan menggeneralisasi (Tanaka, 2014).

Dalam *machine learning* terdapat dua aplikasi utama yaitu klasifikasi dan prediksi. Ciri khas dari *machine learning* adalah adanya proses pelatihan, pembelajaran, atau *training*. Oleh karena itu, *machine learning* membutuhkan data untuk dipelajari yang disebut sebagai data training. Klasifikasi adalah metode dalam *machine learning* yang digunakan oleh mesin untuk memilah atau mengklasifikasikan obyek berdasarkan ciri tertentu sebagaimana manusia mencoba membedakan benda satu dengan yang lain. Sedangkan prediksi atau regresi digunakan oleh mesin untuk menerka keluaran dari suatu data masukan berdasarkan data yang sudah dipelajari dalam training. Metode *machine learning* yang paling populer yaitu Sistem Pengambil Keputusan, *Support Vector Machine* (SVM), dan *Neural Network* (Ahmad, 2017).

Pada pembelajaran *machine learning* terdapat beberapa kategori yaitu sebagai berikut :

1. *Supervised Learning*

Supervised learning adalah sistem pembelajaran mesin yang dilatih di bawah pengawasan. Algoritma dilatih menggunakan data berlabel dan *output* yang diinginkan. Algoritma *Supervised learning* membangun

model berdasarkan data berlabel dan mencoba memprediksi pada dataset baru. Untuk variabel *input* yaitu $(x = x_1, x_2, \dots, x_n)$ dan *output* variabel atau label $(y_1 = y_1, y_2, \dots, y_n)$ algoritma menggunakan fungsi pemetaan dari input ke output sebagai $y = f(x)$. Identya adalah untuk memperkirakan fungsi pemetaan $f(x)$ dengan cukup baik sehingga ketika ada input data baru x' , model dapat memprediksi label keluaran y' dari data tersebut.

2. *Unsupervised Learning*

Sistem *machine learning* yang dilatih tanpa pengawasan atau menggunakan data tanpa label dikenal sebagai *unsupervised learning*. *Unsupervised learning* hanya memiliki data input tanpa variabel output yang sesuai (label). Tujuannya adalah untuk merancang atau menemukan pola data untuk mempelajari lebih lanjut tentang data tersebut. Clustering, visualisasi dan pengurangan dimensi adalah contoh umum dari algoritma pembelajaran mesin *unsupervised machine learning* (Geron, 2017).

3.12 *Deep Learning*

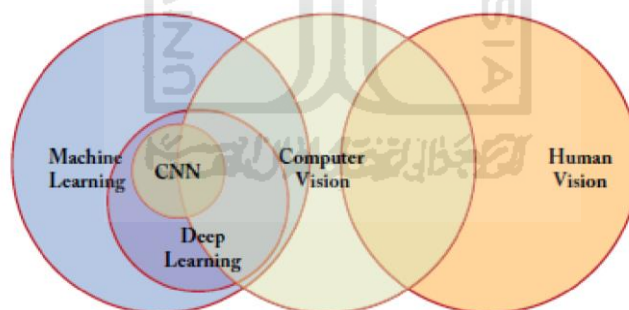
Secara definisi *deep learning* merupakan bagian dari *machine learning* yang digunakan untuk pemodelan abstraksi tingkat tinggi pada suatu data berdasarkan algoritma dengan menggunakan lapisan implementasi dan menggunakan struktur yang kompleks atau sebaliknya, terdiri dari beberapa transformasi non-linear (Fikrieabdillah, 2016).

Deep learning mempunyai sebuah fitur untuk mengekstraksi pola yang didapatkan dari data yang membantu model untuk membedakan kelas sehingga fitur ini juga berperan untuk pencapaian hasil prediksi yang baik, fitur ini disebut dengan *Feature Engineering*. *Deep learning* merupakan cabang dari *Machine learning* yang terinspirasi dari korteks manusia dengan menerapkan jaringan syaraf buatan yang memiliki banyak hidden layer (Santoso & Ariyanto, 2018).

Machine learning menggunakan pembelajaran representasi tingkat tunggal sementara *deep learning* menggunakan kombinasi beberapa lapisan pemrosesan

untuk mempelajari fitur terbaik yang diperlukan untuk mewakili data. Pembelajaran representasi tingkat yang lebih tinggi membuat *deep learning* mampu menyelesaikan tugas kompleks pada data dimensi tinggi (Goodfellow, 2016). Sebagai contoh, dengan kekuatan komputasi yang lebih besar dan memori yang cukup besar, seseorang dapat membuat jaringan saraf dari banyak lapisan, yang disebut *deep neural network*.

Karena keberhasilan luar biasa dalam mempelajari *deep neural network*, teknik *deep learning* saat ini berkembang lebih canggih untuk deteksi, segmentasi, klasifikasi dan *recognition* pada identifikasi dan verifikasi objek dalam gambar. Para peneliti sekarang bekerja untuk menerapkan keberhasilan ini dalam pengenalan pola untuk tugas-tugas yang lebih kompleks seperti diagnosa medis dan terjemahan bahasa otomatis. *Convolutional Neural Networks* adalah salah satu metode populer pada kategori *deep neural network* yang telah terbukti sangat efektif di berbagai bidang seperti pengenalan dan klasifikasi gambar. Gambar 3.15 menggambarkan hubungan antara *Computer Vision*, *Machine Learning*, *Human Vision*, *Deep Learning*, dan CNN.



Gambar 3.17 Hubungan antara *Human Vision*, *Computer Vision*, *Machine Learning*, *Deep Learning*, dan CNN (Khan, Rahmani, Shah, & Bennamoun, 2018)

3.13 Artificial Neural Network

Artificial Neural Network (ANN) merupakan sebuah sistem pembelajaran terhadap penerimaan informasi yang memiliki kinerja layaknya sebuah jaringan syaraf pada manusia. ANN diimplementasikan dengan menggunakan program komputer sehingga mampu menyelesaikan sejumlah proses perhitungan. *Neural*

Network adalah prosesor yang terdistribusi paralel, terbuat dari unit-unit yang sederhana, dan memiliki kemampuan untuk menyimpan pengetahuan yang diperoleh secara eksperimental dan siap pakai untuk berbagai tujuan (Nurmila, Sugiharto, & Sarwoko, 2010).

ANN adalah suatu metode pengelompokan dan pemisahan data yang prinsip kerjanya sama seperti *neural network* pada manusia. Elemen mendasar dari paradigma tersebut adalah struktur yang baru dari sistem pemrosesan informasi. ANN dibentuk untuk memecahkan suatu masalah tertentu seperti pengenalan pola atau klasifikasi dari proses pembelajaran (Puspitaningrum, 2006).

Lapisan *Artificial Neural Network* biasanya terdiri atas 3 layer. Pertama adalah Input layer terdiri dari neuron-neuron berfungsi untuk menerima input. Kedua adalah *Hidden layer* berfungsi meneruskan informasi dari hasil input untuk diteruskan ke layer berikutnya. Ketiga adalah *Output layer* berfungsi untuk menerima hasil dari *output Hidden layer* (Dharma, Putera, & Ardana, 2011). Namun, ada beberapa arsitektur *neural network* yang hanya memiliki lapisan *input* dan lapisan *output*.

Menurut Pham dalam jurnal Hermantoro (Pham, 1994) mengatakan bahwa ANN bersifat fleksibel terhadap inputan data dan menghasilkan *output* respon konsisten. ANN telah banyak digunakan dalam area yang luas. Menurut Kumar & Haynes (Kumar & Haynes, 2003) dalam jurnal (Hamida, 2014) menjelaskan, penerapan ANN dapat mengidentifikasi beberapa aplikasi yaitu :

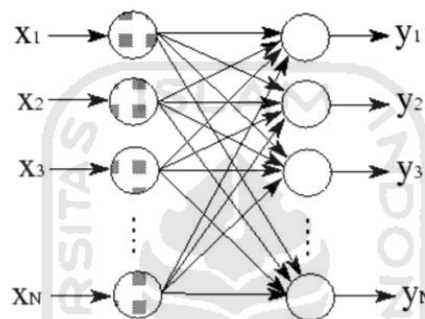
1. Estimasi/prediksi (aproksimasi fungsi, peramalan)
2. Pengenalan pola (klasifikasi, diagnosis, dan analisis diskriminan)
3. Klustering (Pengelompokan tanpa adanya pengetahuan sebelumnya)

3.13.1 Arsitektur Jaringan Artificial Neural Network

Arsitektur yang dapat dibentuk oleh ANN bermacam-macam. Dari yang paling sederhana terdiri satu neuron (*single neuron*) sampai yang paling rumit menjadi multi neuron (*multiple neuron*) dalam satu lapis (*single layer*), sampai jaringan multiple neuron dalam *multiple layers*. Beberapa jaringan tersebut memiliki kemampuan yang berbeda-beda. Semakin rumit suatu jaringan, maka

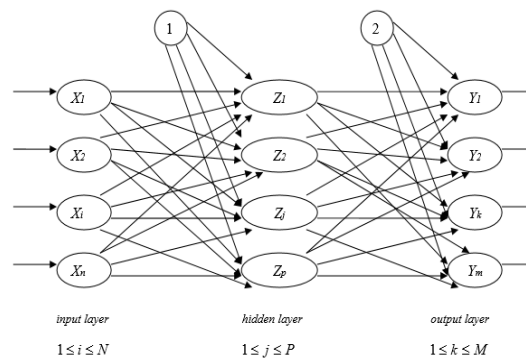
persoalan yang dapat diselesaikan menjadi lebih luas. Menurut Hermawan (2006), *Arsitektur neural network* dapat dibagi berdasarkan jumlah lapisannya, diantaranya sebagai berikut:

1. *Single Layer Neural Network* : Jaringan dengan lapisan tunggal terdiri dari 1 lapisan input dan 1 lapisan output. Setiap neuron yang terdapat di dalam lapisan input selalu terhubung dengan setiap neuron yang terdapat pada lapisan output. Jaringan ini hanya menerima input kemudian secara langsung akan mengolahnya menjadi output tanpa harus melalui lapisan tersembunyi (*hidden layer*) :



Gambar 3.18 *Single Layer Neural Network*

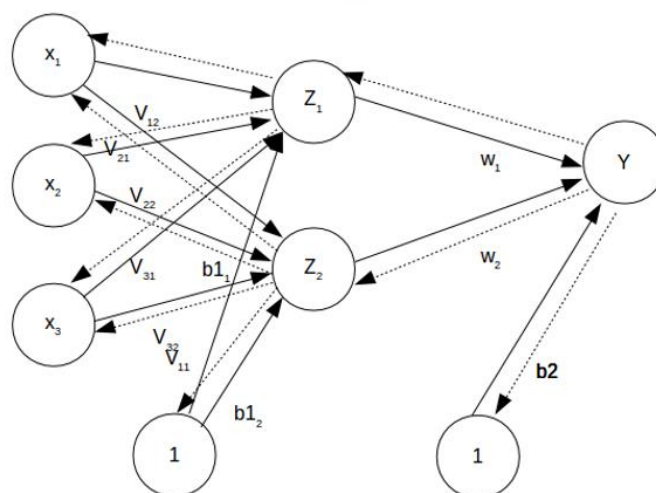
2. *Multi Layers Neural Network* : Jaringan dengan lapisan jamak memiliki ciri khas tertentu yaitu memiliki 3 jenis lapisan yakni lapisan input, lapisan output, dan lapisan tersembunyi. *Multi layer neural network* merupakan model yang paling sering digunakan dalam pengembangan sistem neural karena memiliki kinerja yang baik dalam sisi keakuratan. Jaringan dengan banyak lapisan ini dapat menyelesaikan permasalahan yang lebih kompleks dibandingkan jaringan dengan lapisan tunggal. Akan tetapi, proses pelatihan sering membutuhkan waktu yang cenderung lama. Cara kerja jaringan ini adalah, setelah input masuk ke input layer maka data akan diolah dan diteruskan ke masing-masing bagian di depannya sampai pada output layer (Prabaningrum, 2019). Gambar 3.17 di bawah ini merupakan arsitektur *multi layer neural network*.



Gambar 3.19 Multi Layer Neural Network

3.13.2 Backpropagation

Backpropagation merupakan algoritma pembelajaran yang terawasi atau *supervised learning* dan biasanya digunakan oleh perceptron dengan banyak lapisan untuk mengubah bobot-bobot yang terhubung dengan neuron-neuron yang ada pada lapisan tersembunyi. Pada algoritma ini, *error* yang digunakan adalah *error output* untuk mengubah nilai bobot-bobotnya dalam arah mundur (backward). Jika ingin mendapatkan *error* ini, maka tahap perambatan maju (*forward propagation*) harus dikerjakan terlebih dahulu. Pada saat perambatan maju, neuron-neuron diaktifkan dengan menggunakan fungsi aktivasi yang dapat diturunkan, seperti fungsi sigmoid (Kusumadewi, 2004). Berikut gambar 3.18 di bawah ini merupakan arsitektur jaringan untuk *backpropagation*:

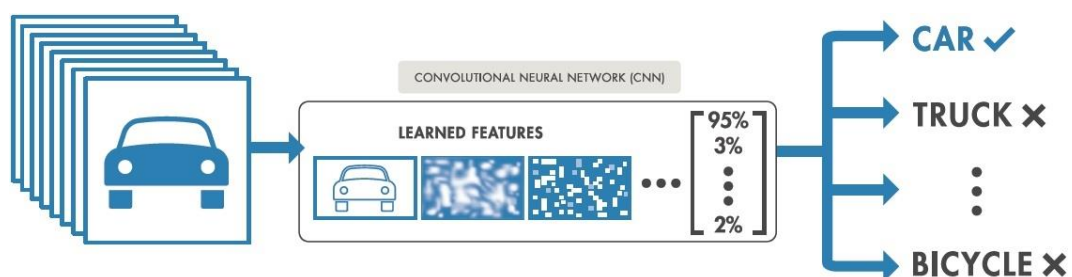


Gambar 3.20 Arsitektur Jaringan Backpropagation

Gambar 3.18 di atas merupakan arsitektur jaringan *backpropagation*. Pada gambar tersebut terlihat bahwa *neural network* terdiri dari tiga unit neuron pada lapisan *input* (X_1 , X_2 , dan X_3), dua neuron pada *hidden layer* (Z_1 dan Z_2) serta 1 unit neuron pada lapisan *output* (Y). Bobot yang menghubungkan lapisan *input* dengan neuron pertama pada *hidden layer* adalah V_{11} , V_{21} , dan V_{31} . Bobot bias yang menuju neuron pertama dan kedua pada *hidden layer* ditunjukkan dengan b_{11} dan b_{12} . Bobot yang menghubungkan Z_1 dan Z_2 dengan neuron pada lapisan *output* adalah W_1 dan W_2 . Bobot bias b_2 menghubungkan *hidden layer* dengan lapisan *output*.

3.14 Convolutional Neural Network

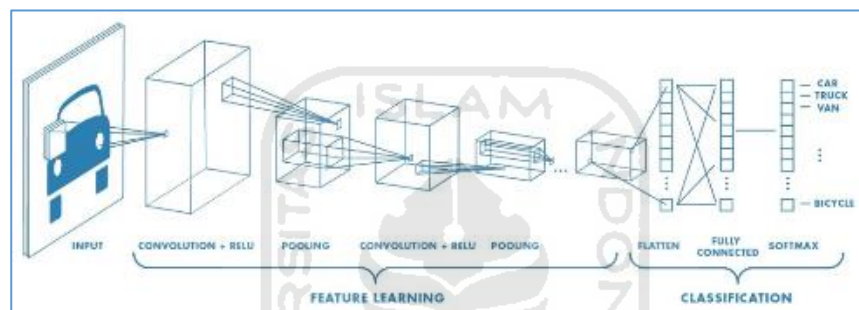
Convolutional network atau yang dikenal juga dengan nama *Convolutional Neural Network* (CNN) adalah tipe khusus dari *neural network* untuk memproses data yang mempunyai topologi jala atau *grid-like topology*. Pemberian nama *convolutional neural network* mengindikasikan bahwa jaringan tersebut menggunakan operasi matematika yang disebut konvolusi. Konvolusi sendiri adalah sebuah operasi linear. Jadi *Convolutional Network* adalah *neural network* yang menggunakan konvolusi minimal satu pada lapisannya (Lecun, Bengio, & Geoffrey, 2015). CNN digunakan untuk melakukan klasifikasi data yang berlabel dengan menggunakan metode *supervised learning* yang cara kerjanya adalah terdapat data yang dilatih dan terdapat variabel yang ditargetkan sehingga tujuan dari metode ini yaitu mengelompokkan suatu data ke data yang sudah ada. *Convolutional neural network* (ConvNets) merupakan *special case* dari *artificial neural network* (ANN) yang saat ini diklaim sebagai model terbaik untuk memecahkan masalah *object recognition* dan *detection*.



Gambar 3.21 Ilustrasi Arsitektur CNN (Matlab)

CNN akan belajar memprediksikan label gambar yang diberikan sesuai dengan representasi *feature* mulai dari yang sederhana hingga yang lebih kompleks. *Feature* ini yang kemudian digunakan dalam jaringan untuk memprediksi kategori dengan benar.

Convolutional Neural Network merupakan metode yang tercakup di dalam kelas *Feed Forward Neural Network* yang terinspirasi dari visual cortex dari otak dan dikhususkan untuk memproses data yang memiliki struktur grid. CNN mempunyai beberapa jenis layer yang digunakan. Berikut arsitektur CNN pada gambar 3.20 di bawah ini.



Gambar 3.22 Arsitektur *Convolutional Neural Network*

Arsitektur dari CNN dibagi menjadi 2 bagian besar yaitu *Feature Extraction Layer* dan *Fully-Connected Layer*.

a. *Feature Extraction Layer*

Proses yang terjadi pada bagian ini adalah “encoding” dari sebuah image menjadi features yang berupa angka-angka yang merepresentasikan image tersebut (*Feature Extraction*). *Feature extraction layer* terdiri dari dua bagian. Adapun layer pertama yaitu *Convolutional Layer* dan layer kedua merupakan *pooling layer*. Pada setiap layer terdapat fungsi aktivasi. Layer ini menerima input gambar secara langsung dan memprosesnya sehingga menghasilkan *output* berupa vektor untuk diolah pada layer berikutnya.

b. *Fully-Connected Layer*

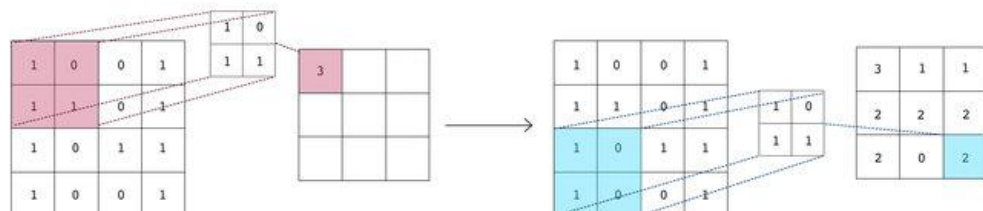
Tersusun dari beberapa layer dan pada setiap layer tersusun atas neuron yang terkoneksi secara penuh (*fully connected*) dengan layer lainnya. Layer ini menerima input dari hasil *output Feature Extraction Layer* (layer ekstraksi fitur) gambar berupa vektor dan kemudian ditransformasikan dengan

tambahan beberapa *hidden layer*. Hasil output yang dihasilkan adalah berupa skoring kelas untuk klasifikasi (Zufar & Setiyono, 2016).

3.14.1 Convolutional Layer

Convolutional Layer merupakan layer pertama yang menerima *input* gambar langsung pada arsitektur. Operasi pada layer ini sama dengan operasi konvolusi yaitu melakukan operasi kombinasi linier filter terhadap daerah lokal. Filter merupakan representasi bidang reseptif dari neuron yang terhubung ke dalam daerah lokal pada input gambar. *Convolutional Layer* melakukan operasi konvolusi pada output dari layer sebelumnya. Layer tersebut adalah proses utama yang mendasari sebuah CNN (Dewi, 2018).

Konvolusi adalah suatu istilah matematis yang berarti mengaplikasikan sebuah fungsi pada output fungsi lain secara berulang. Dalam pengolahan citra, konvolusi berarti mengaplikasikan sebuah kernel pada citra disemua *offset* yang memungkinkan seperti yang ditunjukkan pada Gambar 3.21. Kotak dengan ukuran 2×2 secara keseluruhan adalah citra yang akan dikonvolusi. Kernel bergerak dari sudut kiri atas ke kanan bawah. Dalam tahap ini, pergerakan kernel juga diatur oleh pendefinisian *stride*, *stride* yang digunakan pada gambar di bawah ini adalah 1. Hasil konvolusi dari citra tersebut dapat dilihat pada gambar disebelah kanannya. Tujuan dilakukannya konvolusi pada data citra adalah untuk mengekstraksi fitur dari citra input. Konvolusi akan menghasilkan transformasi linear dari data input sesuai informasi spasial pada data. Bobot pada layer tersebut menspesifikasikan kernel konvolusi yang digunakan, sehingga kernel konvolusi dapat dilatih berdasarkan input pada CNN.



Gambar 3.23 Proses konvolusi

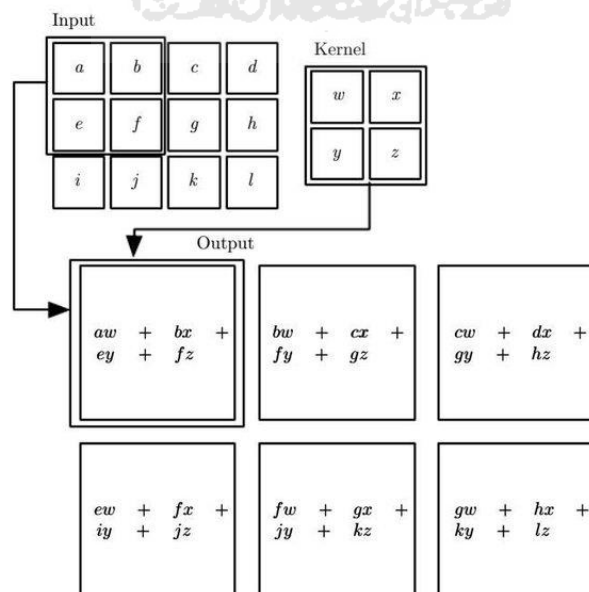
Operasi konvolusi yang dikenakan pada fungsi $s(t)$ dengan bobot (atau sering disebut kernel) $w(t)$, $x*w$, didefinisikan secara matematis pada persamaan (3.5).

$$s(t) = (x*w)(t) \quad (3.5)$$

Fungsi $s(t)$ memberikan *output* tunggal berupa *feature Map*. Argumen pertama adalah *input* yang merupakan x dan argumen kedua w sebagai kernel atau filter. Apabila dilihat *input* sebagai citra dua dimensi, maka bisa dikatakan t sebagai piksel dan menggantinya dengan i dan j . Maka dari itu, operasi untuk konvolusi ke *input* dengan lebih dari satu dimensi dapat menulis sebagai berikut :

$$s_{(i,j)} = (K * I)_{(i,j)} = \sum_m \sum_n I_{(i+m,j+n)} * K_{(m,n)} \quad (3.6)$$

Berdasarkan kedua persamaan diatas merupakan perhitungan dasar dalam operasi konvolusi, dengan i dan j adalah sebuah piksel dari citra. Sedangkan m dan n adalah ukuran dimensi suatu kernel. Perhitungan tersebut bersifat kumulatif dan muncul saat K sebagai kernel, kemudian I sebagai *input*. Sebagai alternatif, operasi konvolusi dapat disebut sebagai perkalian-perkalian matriks antara citra *input* dan kernel dimana keluarannya dihitung dengan *dot product*. Proses perhitungan berdasarkan rumus di atas dapat lebih mudah dipahami dengan melihat gambar 3.22 di bawah ini.



Gambar 3.24 Proses konvolusi

3.14.2 Pooling Layer

Pooling Layer atau subsampling merupakan lapisan yang menggunakan fungsi dengan *Feature Map* sebagai masukan dan mengolahnya dengan berbagai macam operasi statistik berdasarkan nilai pixel terdekat. Lapisan *Pooling* pada model *CNN* biasanya disisipkan secara teratur setelah beberapa lapisan konvolusi. Lapisan *Pooling* yang dimasukkan diantara lapisan konvolusi secara berturut-turut dalam arsitektur model *CNN* dapat secara progresif mengurangi ukuran volume *output* pada *Feature Map*, sehingga jumlah parameter dan perhitungan di jaringan berkurang, serta untuk mengendalikan *overfitting* (Shafira, 2018).

Pooling layer terdapat dua jenis yaitu *Max pooling* dan *Average pooling*. *Max pooling* mengekstrak fitur yang paling penting seperti tepi sedangkan *average pooling* lebih halus dibandingkan dengan *max pooling*. Meskipun keduanya digunakan untuk alasan yang sama, *max pooling* lebih baik untuk mengekstrak fitur ekstrem (Rahman, 2017). Sebagai contoh, apabila kita menggunakan *max pooling* 2×2 dengan *stride* 2, maka pada setiap pergeseran filter, nilai yang diambil adalah nilai yang terbesar pada area 2×2 tersebut seperti pada gambar 3.23, pada setiap kotak yang memiliki warna yang berbeda kemudian diambil satu angka yang merupakan nilai maksimalnya. Sedangkan *Average Pooling* akan mengambil nilai rata-rata.



Gambar 3.25 Proses *Max Pooling Layer*

3.14.3 Fungsi Aktivasi

Fungsi aktivasi adalah fungsi yang menggambarkan hubungan antara tingkat aktivitas internal atau *summation function* yang dapat berupa linear ataupun non-linear. Fungsi aktivasi memiliki tujuan untuk menentukan aktif tidaknya suatu neuron (Nurhikmat, 2018).

Terdapat berbagai fungsi aktivasi yang dapat digunakan dalam jaringan syaraf, namun yang paling umum digunakan adalah *nonlinear function*. Fungsi aktivasi mengambil input bernilai riil dan menekannya dalam kisaran kecil seperti $[0,1]$ dan $[-1,1]$. Fungsi nonlinear juga dapat dipahami sebagai *switching* atau mekanisme seleksi, yang memutuskan apakah neuron akan diaktifkan atau tidak diberikan semua inputnya. Fungsi aktivasi yang biasa digunakan dalam *deep neural network* dapat digunakan untuk mengaktifkan *error backpropagation*. Di bawah ini adalah fungsi aktivasi yang paling umum digunakan dalam *deep neural network* (Khan, Rahmani, Shah, & Bennamoun, 2018) :

1. ReLu (*Rectified Linier Unit*) merupakan fungsi aktivasi pada *Artificial Neural Network* yang saat ini banyak digunakan, berikut rumus pada ReLu.

$$f(x) = \max(x,0) \quad (3.7)$$

Dimana x merupakan input neuron yang dikenal sebagai fungsi ramp dan analog dengan rektifikasi *half-wave* pada teknik elektro. Jika input lebih besar 0, outputnya sama dengan input. Fungsi ReLu lebih mirip neuron seperti pada tubuh manusia. Aktivasi ReLu pada dasarnya sebuah fungsi aktivasi non-linier yang paling sederhana. Bila mendapatkan input positif, turunannya hanya 1, dengan kata lain, aktivasi hanya men-threshold pada nilai nol. Penelitian menunjukkan bahwa ReLu menghasilkan pelatihan yang lebih cepat untuk jaringan besar (Dewi, 2018).

2. Fungsi aktivasi sigmoid merupakan fungsi non-linear yang mana masukannya adalah nilai real dan output yang dihasilkan adalah nilai antara 0 dan 1, yang mana jika nilai input yang dimasukkan adalah nilai negative yang besar maka output yang dihasilkan adalah 0 kemudian apabila nilai input adalah nilai positif yang sangat besar maka akan menghasilkan output angka 1.

$$f(x) = \frac{1}{1 + e^{-x}} \quad (3.8)$$

3.14.4 *Stride*

Stride adalah parameter yang menentukan berapa jumlah pergeseran filter. Jika nilai *stride* adalah 1, maka *convolution filter* akan bergeser sebanyak 1 pixels secara horizontal lalu vertical. Dalam prosesnya, semakin kecil nilai *stride* maka akan membuat semakin detail informasi yang didapatkan dari sebuah *input*, tetapi akan membutuhkan komputasi yang lebih dibandingkan dengan *stride* yang besar (Dewi, 2018).

3.14.5 *Padding*

Padding atau *Zero Padding* adalah parameter yang menentukan jumlah pixel (berisi nilai 0) yang akan ditambahkan disetiap sisi dari *input*. Hal ini digunakan dengan tujuan untuk memanipulasi dimensi *output* dari *convolution layer* (*Feature Map*). Tujuan dari padding antara lain :

1. Dimensi *output* dari *convolution layer* selalu lebih kecil dari inputnya (kecuali penggunaan 1x1 filter dengan *stride* 1). *Output* ini akan digunakan kembali sebagai *input* dari *convolution layer* selanjutnya, sehingga makin banyak informasi yang terbuang. Dengan menggunakan *padding*, kita dapat mengatur dimensi *output* agar tetap sama seperti dimensi *input* atau setidaknya tidak berkurang secara drastis. Sehingga kita bisa menggunakan *convolution layer* yang lebih dalam sehingga lebih banyak *features* yang berhasil di-extract.
2. Meningkatkan performa dari model karena *convolution filter* akan fokus pada informasi yang sebenarnya yaitu yang berada diantara *zero padding* tersebut. Untuk menghitung dimensi dari *feature map* kita bisa digunakan rumus seperti di bawah ini:

$$Output = \frac{W-N+2P}{S} + 1 \quad (3.9)$$

Keterangan :

W = Panjang/Tinggi Input

N = Panjang/Tinggi Filter

P = *Zero Padding*

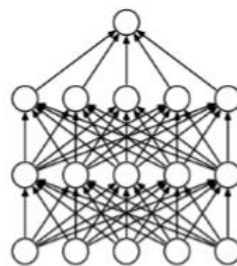
S = *Stride*

3.14.6 Fully Connected Layer

Fully connected layer merupakan lapisan dimana seluruh neuron aktivasi dari lapisan sebelumnya terhubung dengan neuron pada lapisan selanjutnya seperti halnya jaringan syaraf tiruan biasa. *Activation map* yang dihasilkan dari *feature extraction layer* masih berbentuk multidimensional array, sehingga harus melakukan *reshape activation map* menjadi sebuah vector agar bisa digunakan sebagai input dari *fully connected layer*. Layer ini memiliki *hidden layer*, *activation function*, *output layer*, dan *loss function*. Layer ini adalah layer yang biasanya digunakan dalam penerapan *multi layer perceptron* dan bertujuan untuk melakukan transformasi pada dimensi data agar data dapat diklasifikasikan secara linear. Setiap neuron pada *convolution layer* perlu ditransformasi menjadi data satu dimensi terlebih dahulu sebelum dapat dimasukkan ke dalam sebuah *fully connected layer*. Karena hal tersebut menyebabkan data kehilangan informasi spasialnya dan tidak reversibel, sedangkan *fully connected layer* hanya dapat diimplementasikan di akhir jaringan (Sholihah, 2018).

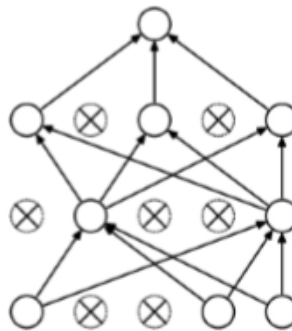
3.14.7 Dropout

Dropout regularization, merupakan teknik regulasi pada sebuah jaringan syaraf tiruan dimana beberapa neuron akan dipilih untuk tidak aktif agar sebuah jaringan syaraf tiruan tidak mengalami *overfitting*. *Dropout* ini merupakan sebuah teknik yang digunakan untuk memilih secara acak neuron mana yang tidak digunakan pada saat *training*. Neuron ini di “*Drop-out*” secara acak. Artinya kontribusi neuron yang dibuang ini akan diberhentikan sementara pada saat melakukan proses *feedforward* dan tidak akan diberikan bobot baru pada neuron pada saat melakukan *backpropagation*. Berikut contoh arsitektur *Neural Network* sebelum adanya proses *dropout*.



Gambar 3.26 Neural Network sebelum dropout

Berikut contoh arsitektur Neural Network yang sudah dilakukan proses dropout.



Gambar 3.27 Neural Network setelah dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014)

3.14.8 *Softmax Classifier*

Softmax Classifier adalah sebuah fungsi aktivasi yang digunakan untuk permasalahan klasifikasi, biasanya fungsi aktivasi ini digunakan pada *output layer*. Pada dasarnya fungsi ini adalah probabilitas eksponensial yang dinormalisasi dari pengamatan kelas yang diwakili sebagai aktivasi neuron. Fungsi eksponensial akan meningkatkan probabilitas nilai maksimum lapisan sebelumnya dibandingkan dengan nilai lainnya.

3.14.9 *Loss Function*

Loss Function atau *Cost Function* merupakan fungsi yang menggambarkan nilai kerugian yang terkait dengan semua kemungkinan yang dihasilkan oleh model. *Loss Function* bekerja ketika model pembelajaran memberikan kesalahan yang harus diperhatikan. *Loss Function* yang baik adalah fungsi yang menghasilkan *error* yang diharapkan paling kecil atau paling rendah.

3.15 *Integral Image*

Integral Image atau *Summed Area Table*, pertama kali diperkenalkan kepada kami pada tahun 1984, tetapi tidak secara tepat diperkenalkan ke dunia *computer vision* hingga tahun 2001 oleh Viola dan Jones dengan kerangka deteksi objek Viola-Jones. Integral image sering digunakan pada algoritma untuk pendeteksian wajah. Dengan menggunakan integral image proses perhitungan bisa dilakukan hanya dengan satu kali *scan* dan memakan waktu yang cepat dan akurat. *Integral*

image digunakan untuk menghitung hasil penjumlahan intensitas piksel pada daerah tertentu, dalam hal ini daerah yang berbentuk *rectangular* atau persegi panjang. Sehingga hasil dari *Integral Image* dikatakan sebagai representasi baru dari citra asli. Apabila ada sebuah citra masukan yang dilalui oleh fitur haar dapat dilihat pada gambar 3.28.



Gambar 3.28 Visualisasi Citra Masukan

Dari nilai-nilai pixel yang didapatkan pada fitur tersebut, maka akan dihitung nilai *integral image* pada fitur tersebut dengan rumus 3.10.

$$s(x, y) = i(x, y) + s(x, y - 1) + s(x - 1, y) - s(x - 1, y - 1) \quad (3.10)$$

Keterangan :

- $s(x, y)$ = nilai hasil penjumlahan dari tiap-tiap pixel
- $i(x, y)$ = nilai intensitas diperoleh dari nilai pixel citra masukan
- $s(x - 1, y)$ = nilai pixel pada sumbu x
- $s(x, y - 1)$ = nilai pixel pada sumbu y
- $s(x - 1, y - 1)$ = nilai pixel diagonal

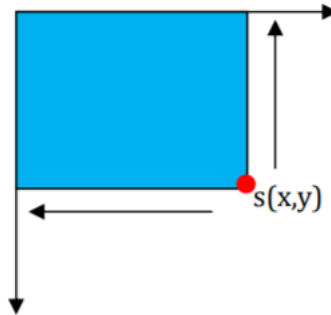
Berikut ini adalah contoh original matrix berukuran 4x4, Misalkan kita ingin menghitung jumlah seluruh piksel (intensitas) atau nilai dari matriks di atas pada daerah yang diberi kotak warna merah.

Original Matrix

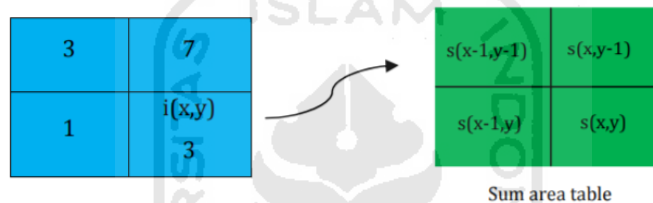
3	7	7	3
1	3	3	1
5	9	9	5
3	6	6	3

Gambar 3.29 Matriks Original

Langkah pertama yang dilakukan yaitu membuat *summed area table* yang merupakan *integral image* dari matriks original dengan ukuran yang sama persis dengan menggunakan rumus 3.10. Berikut merupakan ilustrasi perhitungannya.



Gambar 3.30 Matriks Original



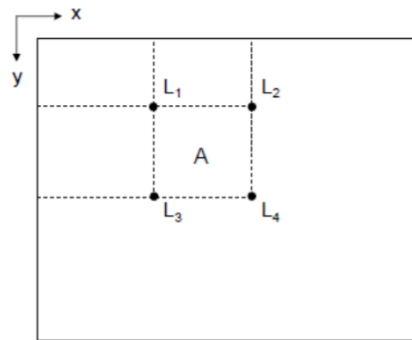
Gambar 3.31 Matriks Original

$s(x,y)$ merupakan nilai *summed area* pada (x,y) , nilai $i(x,y)$ merupakan intensitas dari matriks/citra asli, $s(x,y-1)$ merupakan nilai *summed area* dari nilai piksel tetangga y , dan $s(x-1,y)$ merupakan nilai *summed area* dari nilai piksel tetangga x serta $s(x-1,y-1)$ merupakan nilai *summed area* dari nilai piksel tetangga diagonalnya. Apabila dilakukan perhitungan untuk semua pixel yang terdapat dalam kotak-kotak fitur, maka akan didapatkan hasil perhitungan dari *integral image* dapat dilihat pada gambar 3.10.

Summed Area Table			
3	10	17	20
4	14	24	28
9	28	47	56
12	37	62	74

Gambar 3.32 Integral Image

Hasil dari *Integral Image* akan digunakan untuk menghitung jumlah piksel atau intensitas suatu citra untuk bagian tertentu saja. Bagian ini merupakan sebuah area yang berbentuk *rectangle*.



Gambar 3.33 Area Rectangle

Untuk menghitung jumlah piksel dari daerah A, maka rumusnya adalah,

$$A = L_1 + L_4 - (L_2 + L_3)$$

Memetakan kembali simbol L_1 sampai L_4 ke dalam tabel *integral image*:

Summed Area Table			
3	10	17	20
4	14	24	28
9	28	47	56
12	37	62	74

Gambar 3.34 Integral

$L_1 = 14$, $L_2 = 28$, $L_3 = 37$, $L_4 = 74$, maka jumlah pixel pada daerah A adalah :

$$\begin{aligned} A &= 14 + 74 - (28 + 37) \\ &= 23 \end{aligned}$$

Apabila sudah didapatkan nilai integral image dari sebuah citra masukan dan nilai jumlah pixel pada daerah tertentu, maka hasil tersebut akan dibandingkan antara nilai pixel pada daerah terang dan daerah gelap. Jika selisih nilai pixel pada daerah terang dengan nilai pixel pada daerah gelap di atas nilai ambang (*threshold*) maka daerah tersebut dinyatakan memiliki fitur.

3.16 Confusion Matrix

Confusion matrix adalah sebuah tabel yang berisi informasi hasil klasifikasi yang dikerjakan *classifier* berupa kelas sebenarnya dan hasil prediksi. *Confusion matrix* merupakan sebuah metode untuk menganalisis seberapa baik kinerja sebuah model yang telah dibuat dalam mengidentifikasi data dari kelas yang berbeda-beda (Santra, 2012).

		Nilai sebenarnya	
		TRUE	FALSE
Nilai prediksi	TRUE	TP (True Positive) <i>Correct result</i>	FP (False Positive) <i>Unexpected result</i>
	FALSE	FN (False Negative) <i>Missing result</i>	TN (True Negative) <i>Correct absence of result</i>

Gambar 3.35 Tabel *Confusion Matrix*

Dengan keterangan sebagai berikut:

1. TP (True Positive): keluaran dimana model dapat dengan benar memprediksi kelas positif.
2. TN (True Negative): keluaran dimana model dapat dengan benar memprediksi kelas negatif.
3. FP (False Positive): keluaran dimana model tidak dapat memprediksi kelas positif.
4. FN (False Negative): keluaran dimana model tidak dapat memprediksi kelas negatif.

Dari hasil tabel 3.35 di atas dapat digunakan untuk menghitung nilai akurasi. Akurasi untuk menggambarkan seberapa akurat sistem dapat mengklasifikasikan data secara tepat dan benar. Dapat dikatakan sebagai presentase dari total data yang diidentifikasi dan dinilai benar. Tujuan dari fungsi ini ialah agar akurasi model dapat dipertanggungjawabkan kebenarannya dengan melihat akurasi model dari sisi lainnya. Akurasi didefinisikan sebagai tingkat kedekatan antara nilai prediksi dengan nilai aktual.

$$\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN} \quad (3.11)$$

3.17 Tingkat Akurasi

Tingkat akurasi adalah pengukuran yang dilakukan untuk mengetahui seberapa besar kemampuan sistem mesin dalam mengenali objek yang ada dan kemampuan menghasilkan tingkat kemiripan dengan objek yang sebenarnya. ROC (*Receiver Operating Characteristic*) merupakan grafik dua dimensi dengan *false positive* sebagai garis horizontal dan *true positive* sebagai garis vertikal. Adapaun proses perhitungan untuk mengukur perbedaan performa metode yang

ada adalah dengan menggunakan AUC (*the area under curve*). Adapun tingkat nilai diagnosa dalam ROC sebagai berikut (Andriani, 2013):

1. Akurasi bernilai $0,90 - 1,00 = \text{excellent classification}$
2. Akurasi bernilai $0,80 - 0,90 = \text{good classification}$
3. Akurasi bernilai $0,70 - 0,80 = \text{fair classification}$
4. Akurasi bernilai $0,60 - 0,70 = \text{poor classification}$
5. Akurasi bernilai $0,50 - 0,60 = \text{failure}$

3.18 Python

Python merupakan bahasa pemrograman dengan tujuan umum yang dikembangkan secara khusus untuk membuat *source code* mudah dibaca. *Python* juga memiliki *library* yang lengkap sehingga memungkinkan programmer untuk membuat aplikasi yang mutakhir dengan menggunakan *source code* yang tampak sederhana (Perkovic, 2012).

Python termasuk bahasa pemrograman yang mudah dipelajari karena sintaks yang jelas, dapat dikombinasikan dengan penggunaan modul-modul siap pakai, dan struktur data tingkat tinggi yang efisien. Distribusi *Python* dilengkapi dengan suatu fasilitas seperti *shell* di *Linux*. Lokasi penginstalan *Python* biasa terletak di “/usr/bin/Python”, dan bisa berbeda. Menjalankan *Python*, cukup dengan mengetikkan “*Python*”, tunggu sebentar lalu muncul tampilan “>>>”, berarti *Python* telah siap menerima perintah. Ada juga tanda “...” yang berarti baris berikutnya dalam suatu blok prompt '>>>'. Text editor digunakan untuk modus skrip (Prasetya, 2012).

3.19 Tensorflow

Tensorflow adalah perpustakaan perangkat lunak yang dikembangkan oleh Tim *Google Brain* dalam organisasi penelitian Mesin Cerdas *Google* untuk tujuan melakukan pembelajaran mesin dan penelitian jaringan syaraf dalam. *Tensorflow* kemudian menggabungkan aljabar komputasi teknik pengoptimalan kompilasi, mempermudah penghitungan banyak ekspresi matematis dimana masalahnya adalah waktu yang dibutuhkan untuk melakukan perhitungan. Fitur utamanya meliputi (Taufiq, 2018):

1. Mendefinisikan, mengoptimalkan, dan menghitung secara efisien ekspresi matematis yang melibatkan *array multi dimensi* (tensors).
2. Pemrograman pendukung jaringan syaraf dalam dan teknik pembelajaran mesin.
3. Penggunaan GPU yang transparan, mengotomatisasi manajemen dan optimalisasi memori yang sama dan data yang digunakan. Tensorflow bisa menulis kode yang sama dan menjalankannya baik di CPU atau GPU. Lebih khusus lagi, *tensorflow* akan mengetahui bagian perhitungan mana yang harus dipindahkan ke GPU.
4. Skalabilitas komputasi yang tinggi di seluruh mesin dan kumpulan data yang besar.

Tensorflow menyediakan *Object Detection API* yaitu merupakan salah satu *application programming interface*. *Object Detection API* merupakan salah satu antar muka yang dikembangkan oleh beberapa peneliti yang berkontribusi pada perkembangan *Tensorflow* sehingga *library Tensorflow* ini dapat digunakan oleh kalangan luas dan dapat terus berkembang.

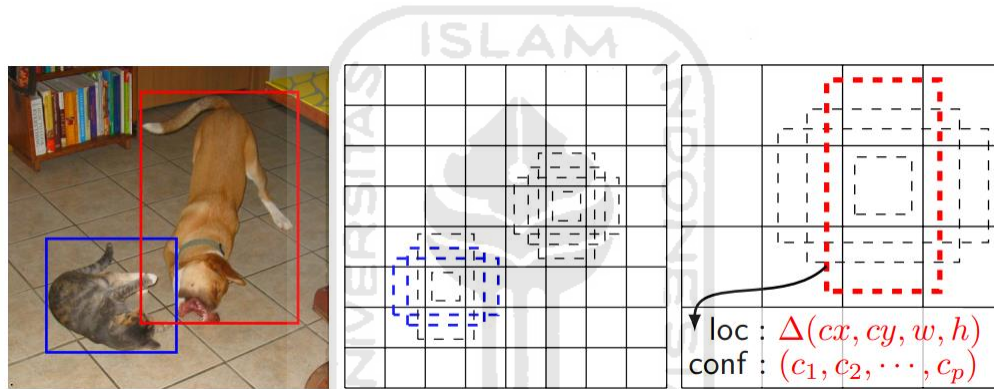
Dalam penggunaannya *application programming interface* (API) *tensorflow* ini sangat mudah digunakan untuk melakukan penelitian terhadap kegunaan *tensorflow* yang dapat di gunakan dalam segala bidang, misalnya dalam bidang klasifikasi objek, membuat *word-embedding*, *vordvector*, mendeteksi sebuah objek, dan banyak lagi. Berbagai macam API dikembangkan melalui *github* yang merupakan perangkat berbagi yang di khususkan untuk para pengembang aplikasi.

Object Detection API, salah satu API yang disediakan oleh beberapa peneliti yang dapat digunakan untuk melakukan *training* sebuah model pendeteksi objek yang didefinisikan oleh peneliti sendiri. Sehingga hasil yang didapat adalah dapat melakukan pendeteksian objek yang sudah ditentukan pada sebuah citra digital.

3.20 Single Shot Multibox Detector (SSD)

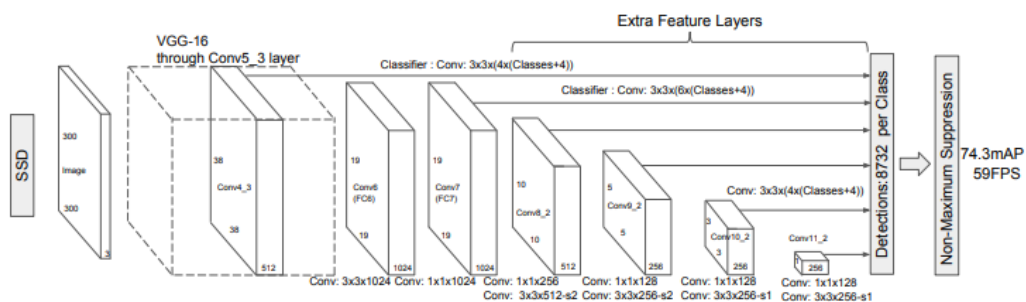
Berdasarkan hasil penelitian yang telah dilakukan oleh Liu dkk (2016), *Single Shot MultiBox Detector* atau yang lebih dikenal dengan SSD merupakan sebuah metode untuk mendeteksi objek dalam gambar menggunakan *single deep*

neural network. SSD, mendiskritisasi ruang *output* dari kotak pembatas menjadi satu set kotak standar pada berbagai rasio dan skala aspek per lokasi *feature map*. Pada saat melakukan prediksi, jaringan menghasilkan skor untuk keberadaan setiap kategori objek disetiap kotak default dan menghasilkan penyesuaian kotak sehingga lebih cocok dengan bentuk objek. Selain itu, jaringan menggabungkan prediksi dari beberapa *feature map* dengan resolusi berbeda untuk secara alami menangani objek dari berbagai ukuran. Hasil eksperimen Liu dkk menunjukkan bahwa dibandingkan dengan metode satu tahap lainnya, SSD memiliki akurasi yang jauh lebih baik bahkan dengan ukuran gambar input yang lebih kecil. *Framework SSD* dapat didefinisikan dengan Gambar 3.26 sebagai berikut:



Gambar 3.36 (a) Input yang dibutuhkan untuk *training* pada SSD berupa kotak untuk masing-masing objek (b) Visualisasi *feature map* berukuran 8 x 8 (c) *Feature map* 4 x 4 berupa detail dari *feature map* 8 x 8 (Liu, 2016)

SSD menggunakan model VGG-16 sebagai jaringan dasar dalam pembuatannya, adapun arsitektur SSD dapat divisualisasikan dalam gambar 3.27 sebagai berikut.



Gambar 3.37 Arsitektur SSD

VGG-16 Network adalah kepanjangan dari “*Visual Geometry Group-16 Network*”, merupakan arsitektur jaringan *convolutional neural network (CNN)* yang dibuat oleh grup peneliti dalam kompetisi *Image Net Large Scale Visual Recognition Challenge (ILSVRC)* pada tahun 2014 untuk melakukan klasifikasi dan lokalisasi pada gambar. Arsitektur *CNN* ini memiliki 16 layer dimana terdiri dari:

1. *Convolution using 64 filters*
2. *Convolution using 64 filters + Max pooling*
3. *Convolution using 128 filters*
4. *Convolution using 128 filters + Max pooling*
5. *Convolution using 256 filters*
6. *Convolution using 256 filters*
7. *Convolution using 256 filters + Max pooling*
8. *Convolution using 512 filters*
9. *Convolution using 512 filters*
10. *Convolution using 512 filters + Max pooling*
11. *Convolution using 512 filters*
12. *Convolution using 512 filters*
13. *Convolution using 512 filters + Max pooling*
14. *Fully connected with 4096 nodes*
15. *Fully connected with 4096 nodes*
16. *Output layer with Softmax activation*

3.21 Mobilenet V1

MobileNets merupakan sebuah model yang didasarkan pada arsitektur konvolusi efisien yang dapat dipisahkan secara mendalam untuk membangun *deep neural networks*. Dalam penelitian yang dilakukan oleh Howard dkk pada tahun (2017) menemukan bahwa kecenderungan umum *MobileNets* adalah untuk membuat jaringan yang lebih dalam dan lebih rumit untuk mencapai akurasi yang lebih tinggi.

dengan menggunakan kamera *smartphone Iphone 6s Plus* dengan resolusi 12 megapixel pada tanggal 30 Januari – 1 Februari 2020 pukul 09.00 -12.00 WIB di gedung FMIPA UII. Estimasi jarak pengambilan gambar antara 3 – 10 meter dari objek.

Metode pengambilan sampel untuk data citra rambu K3 menggunakan metode quota sampling. Quota Sampling merupakan sebuah metode teknik sampling *non-probability* yang dapat digunakan oleh karena banyaknya jumlah populasi mencapai nilai tak berhingga atau ketidakpastian jumlah populasi dari data penelitian tersebut. Dalam mengambil sampel, peneliti terlebih dahulu menentukan banyaknya sampel yang akan diambil untuk sampel penelitian.

Data citra dibagi kedalam 2 kelompok yaitu data train dan data test, dimana perbandingan proporsi 80% untuk data train dan 20% untuk data test. Banyaknya data yang digunakan untuk komputasi berjumlah 600 data training dan 150 data testing pada masing-masing kategori atau dengan keseluruhan berjumlah 1200 data training dan 300 data testing.

4.4 Perangkat Penelitian

Pengambilan dan pengumpulan data menggunakan kamera *smartphone Iphone 6S plus* dengan resolusi 12 megapixel. Berdasarkan beberapa penelitian hasil gambar yang digunakan untuk penelitian adalah gambar dengan ukuran pixel yang tidak terlalu besar sehingga ukuran pixel kamera 12 megapixel sudah layak digunakan untuk mengambil gambar. Model didapatkan dengan melakukan proses *training (Machine learning)* sehingga membutuhkan perangkat *hardware* seperti laptop/komputer dengan spesifikasi tinggi untuk mendapatkan suatu model yang dapat memprediksi objek. Berikut ini adalah spesifikasi perangkat yang digunakan.

Tabel 4.2 Spesifikasi *Iphone 6S plus*

No.	Komponen	Spesifikasi
1.	Kamera	12 Megapixel
2.	RAM	2 GB
3.	Kapasitas	64 GB

Tabel 4.3 Spesifikasi Lenovo Ideapad C340

No.	Komponen	Spesifikasi
1.	<i>Processor</i>	Intel® Core™ i7-8565U Processor (1.80 GHz. up to 4.60 GHz.)
2.	<i>RAM</i>	16GB DDR4
3.	<i>OS Type</i>	Windows 10 Home Single Language 64-bit
4.	<i>GPU</i>	NVIDIA GeForce MX230 2GB

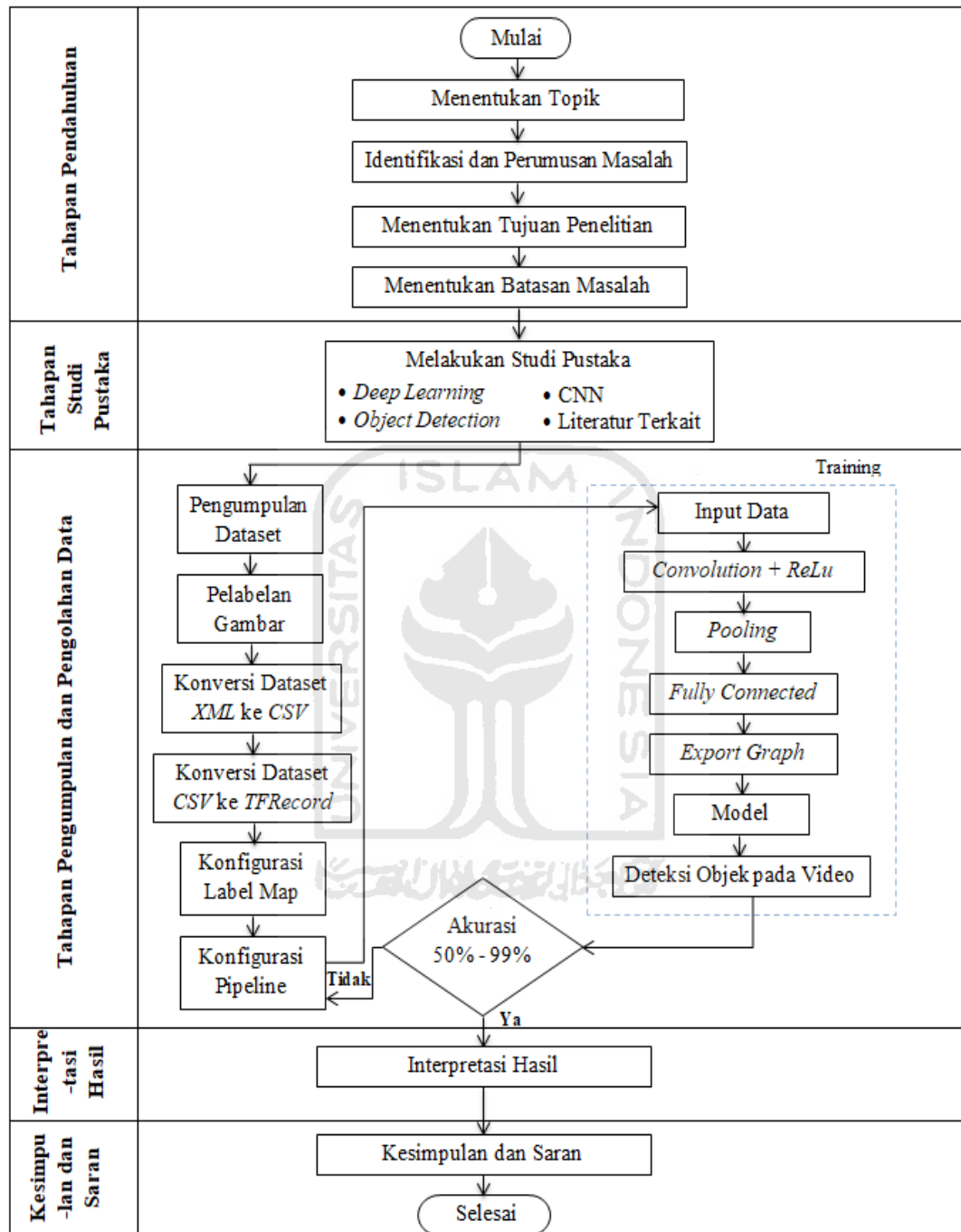
Tabel 4.4 Perangkat Lunak yang Digunakan

No.	Perangkat Lunak	Kegunaan
1.	<i>Anaconda dan Python 3.7.0</i>	Bahasa pemrograman untuk mengembangkan sistem
2.	<i>Tensorflow CPU 2.0</i>	<i>Library</i> untuk keperluan <i>deep learning</i>
3.	<i>OpenCV</i>	<i>Library</i> untuk mengolah gambar
4.	<i>LabelImg</i>	<i>Library</i> yang digunakan untuk melabeli <i>class</i> data

4.5 Metode Penelitian

Metode yang digunakan pada penelitian ini adalah metode *Convolutional neural network* (CNN) pada *deep learning* dimana *deep learning* merupakan suatu metode bagian dari ANN (*Artificial Neural Network*). Metode *Convolutional neural network* (CNN) digunakan untuk mendeteksi dan mengklasifikasi objek rambu K3 .

4.6 Diagram Alir Penelitian



Gambar 4.1 Flowchart Alur Penelitian

Pada penelitian ini adapun tahap-tahap yang di lakukan adalah sebagai berikut :

1. Langkah pertama yaitu menentukan topik penelitian yang akan dilakukan pada penelitian ini.
2. Selanjutnya peneliti mengidentifikasi dan merumuskan masalah yang dapat diteliti atau dikaji dalam penelitian ini, yang bermaksud akan tercapainya tujuan dari penelitian. Dalam penelitian ini juga ditentukan batasan penelitian sehingga tujuan dan manfaat penelitian dapat tercapai dengan baik.
3. Kemudian peneliti melakukan studi putaka pada beberapa literatur mengenai topik yang telah ditentukan
4. Proses awal dari pengolahan data penelitian ini ialah pengumpulan data gambar atau citra berupa citra digital rambu K3 yaitu rambu Jalur Evakuasi dan rambu Alat Pemadam Api yang berada di gedung FMIPA UII dengan cara mengambil gambar objek dengan kamera *smartphone*.
5. Langkah selanjutnya setelah semua data terkumpul adalah pelabelan data gambar. Proses ini bertujuan untuk memberi label pada objek yang akan dideteksi secara satu per satu dengan nama yang sama. Pada penelitian ini nama atau label yang diberikan pada gambar objek rambu Jalur Evakuasi adalah “Jalur Evakuasi” sedangkan untuk rambu Alat Pemadam Api adalah “Alat Pemadam Api”. Pemberian label pada gambar menggunakan aplikasi LabelImg. Setelah pelabelan, file yang disimpan akan tersimpan dalam format file “.xml” dengan format *PASCAL VOC*.
6. Kemudian agar file *XML* tersebut dapat terbaca pada *framework Tensorflow* perlu adanya konversi menjadi file berekstensi *CSV*.
7. Setelah itu peneliti melakukan konversi berkas *CSV* tersebut menjadi format *TFRecord* atau *Tensorflow Record* yang merupakan format penyimpanan dari *Tensorflow*. File *TFRecord* tersebut digunakan dalam proses *feeding data* atau membaca data input sehingga informasi dataset dapat diambil secara langsung dengan fungsi *feed_dictionary*.
8. Langkah selanjutnya adalah proses membuat label map. Label map dibuat untuk mendefinisikan numerikal kategori Rambu K3 dalam kalkulasi. Pada penelitian ini menggunakan dua objek yang akan dideteksi sehingga label map

yang akan dibuat berjumlah dua item yang mana terdiri dari id dan nama (*name*) yang harus sesuai dengan urutan dan nama saat dilakukannya proses labeling.

9. Kemudian tahapan selanjutnya adalah konfigurasi pipeline hal ini dilakukan karena *tensorflow* menggunakan *ProtoBuf* sehingga perlu dilakukannya konfigurasi pipeline yang berguna untuk mengkonfigurasi proses training dan evaluasi. Pada penelitian ini menggunakan konfigurasi model *SSD Mobilenet V1*.
10. Setelah konfigurasi pipeline dipastikan telah sesuai dengan rancangan penelitian, maka penulis dapat melakukan training model. Adapun proses dalam training model adalah sebagai berikut:
 - a. Pada saat pengumpulan data, penulis telah membagi citra menjadi training dan testing dengan perbandingan 80:20. Tahap awal, *TFRecord* digunakan sebagai input data dalam pelatihan model.
 - b. Tahap selanjutnya adalah tahap *convolution* yaitu melakukan operasi konvolusi dari layer sebelumnya. Pada proses operasi konvolusi diperlukannya sebuah kernel dengan ukuran tertentu. Kemudian dilanjutkan kepada tahap fungsi aktivasi *Rectifier Linier Unit* (ReLU) yang berfungsi untuk mengubah nilai negatif menjadi nol pada matriks hasil konvolusi.
 - c. Kemudian tahapan selanjutnya adalah *pooling*, pada tahapan ini proses yang dilakukan adalah pengurangan dimensi dari *feature map* dengan menggunakan metode *max pooling*.
 - d. Setelah tahapan *pooling* selesai, tahapan selanjutnya adalah *flattening* yaitu dimana *feature map* dari hasil *pooling layer* diubah menjadi bentuk vector, yang mana vektor tersebut akan menjadi *input* dalam tahapan *fully connected layer*.
 - e. Setelah proses *training* selesai, kemudian dilakukan *export graph* untuk mengeksport model yang akan digunakan untuk mendeteksi objek Rambu K3.

11. Langkah selanjutnya setelah mendapatkan model yaitu mendeteksi objek rambu K3 pada video. Jika tingkat akurasi hasil deteksi bernilai 50%-99% maka model layak digunakan. Namun, jika tidak dapat mendeteksi dan nilai tingkat akurasi dibawah 50% maka dilakukan kembali konfigurasi pipeline dan mengulang training model hingga mendapatkan model terbaik.
12. Setelah mendapatkan model terbaik, langkah selanjutnya melakukan interpretasi pada hasil pendeteksian yang kemudian akan didapatkannya kesimpulan dan saran dari penelitian ini.

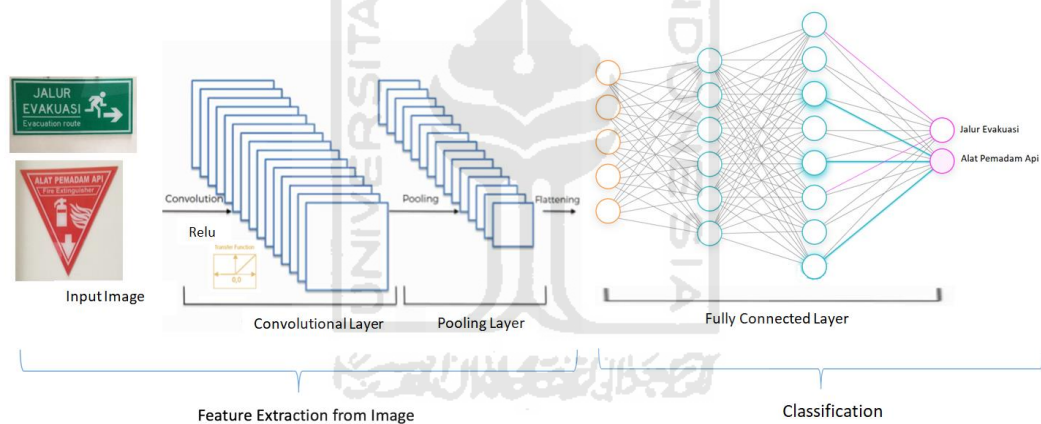


BAB V

PEMBAHASAN

5.1 Rancangan Sistem

Rancangan sistem dalam penelitian ini dimulai dari pengumpulan data berupa dataset gambar rambu K3 yang berada di gedung FMIPA UII. Pengambilan data menggunakan kamera *smartphone Iphone 6s Plus* dengan resolusi 12 *megapixel*. Dataset yang dikumpulkan yaitu berupa gambar rambu K3 jalur evakuasi dan alat pemadam api sebanyak 1500 data citra, terdapat 750 data citra untuk masing-masing *class*. Data citra yang telah terkumpul akan menjadi input data dalam sistem ini. Berikut di bawah ini merupakan visualisasi arsitektur jaringan dalam penelitian ini.



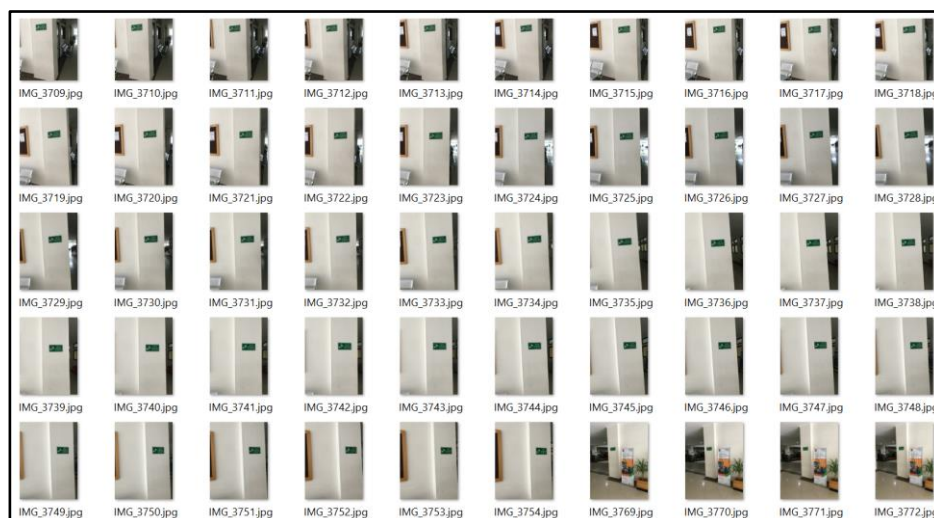
Gambar 5.1 Visualisasi Arsitektur Jaringan

Pada penelitian ini input data yang akan digunakan dalam proses *training* yaitu berupa data gambar dengan ukuran pixel 300x300x3. Ukuran tersebut menunjukkan bahwa gambar memiliki lebar dan tinggi sebesar 300 pixel dan memiliki 3 channel antara lain warna *RGB (Red, Green, Blue)*. Dengan ukuran tersebut maka input kode warna dari semua pixel adalah sebesar 270.000 yang merupakan hasil perkalian 300x300x3. Rancangan sistem dalam penelitian ini seperti pada gambar 5.1 menggunakan algoritma *Convolutional Neural Network (CNN)* yang terdiri atas beberapa bagian. Bagian pertama yaitu input data berupa kode warna pixel dari gambar yang telah dikumpulkan, kemudian dilakukan *Convolutional Layer* beserta dengan Fungsi Aktivasi (*ReLU*). Setelah itu

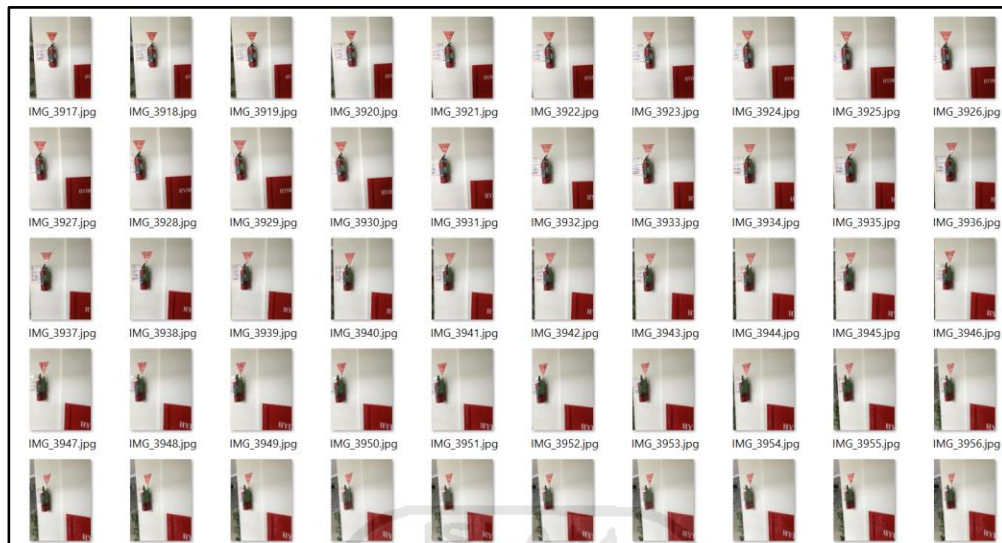
dilakukan *Pooling Layer* pada *output* dari *Convolutional Layer* yaitu *feature map*. Langkah selanjutnya *output* dari *Pooling Layer* yaitu hasil ekstraksi gambar akan masuk pada *Fully Connected Layer*. Dalam *Fully Connected Layer* akan terjadi proses *backpropagation* untuk menentukan klasifikasi kelas objek sehingga sistem dapat mendeteksi objek dengan tingkat akurasi yang tinggi. Pada pengolahan data gambar pada sistem terdapat distribusi gambar yang disebut *integral image*, sistem akan mengolah data sehingga dapat membedakan jumlah intensitas pixel pada setiap bagian. Hal tersebut akan menghasilkan pembagian daerah mana pada gambar tersebut yang dinyatakan memiliki fitur. Sehingga sistem dapat mendeteksi objek pada bagian yang tepat.

5.2 Pembuatan Dataset

Penelitian ini menggunakan dataset pada metode *CNN* yang berupa data gambar. Pembuatan dan pengumpulan dataset dilakukan dengan mengambil gambar rambu-rambu K3 yaitu Jalur Evakuasi dan Alat Pemadam Api menggunakan kamera smartphone *Iphone 6S Plus* dengan resolusi kamera 12 *megapixel*. Pengumpulan data dilakukan pada tanggal 30 Januari – 1 Februari 2020 pukul 09.00 -12.00 WIB di gedung FMIPA UII. Estimasi jarak pengambilan gambar antara 3 – 10 meter dari objek. Semua data gambar yang telah diambil yaitu berjumlah 1500 gambar. Gambar 5.2 dan gambar 5.3 di bawah ini merupakan dataset Jalur Evakuasi dan Alat Pemadam Api.



Gambar 5.2 Dataset Jalur Evakuasi



Gambar 5.3 Dataset Alat Pemadam Api

Setelah semua data terkumpul sesuai dengan quota sampling yang telah ditentukan peneliti maka data gambar dibagi menjadi dua bagian yaitu data *train* dan data *test*. Proses *training* akan berjalan dengan baik apabila menggunakan data *train* gambar dengan jumlah yang banyak, hal ini dikarenakan dengan banyaknya gambar maka model dapat belajar untuk mengenali gambar tersebut dengan lebih baik. Pembagian data *train* dan data *test* menggunakan kuota dengan perbandingan 80% untuk data *training* dan 20% untuk data *testing* data sehingga pembagian datanya menjadi seperti tabel 5.1 di bawah ini :

Tabel 5.1. Pembagian Data *Train* dan Data *Test*

Label	Data Train	Data Test	Jumlah
Jalur Evakuasi	600	150	750
Alat Pemadam Api	600	150	750
Total	1200	300	1500

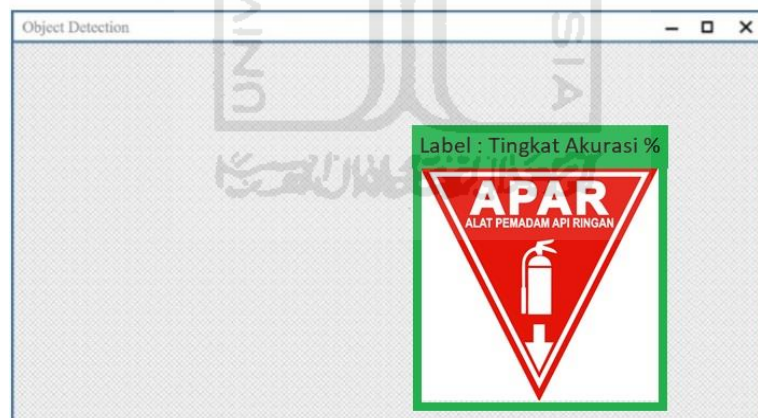
Tabel 5.1 di atas merupakan tabel pembagian dataset, dimana dalam penelitian ini digunakan 2 label yaitu Jalur Evakuasi dan Alat Pemadam Api. Total dataset yang berupa data gambar masing-masing adalah 750 dengan pembagian 600 data *train* dan 150 data *test*. Total keseluruhan dataset yang digunakan adalah 1500 data.

5.3 Rancangan Output

Hasil akhir atau *output* dari penelitian ini yaitu sebuah sistem yang dapat mendeteksi rambu K3 Jalur Evakuasi dan Alat Pemadam Api. *Output* tersebut akan berupa sebuah *frame object detection* yang dapat mendeteksi objek. *Frame* tersebut akan dapat mendeteksi jika terdapat rambu K3 Jalur Evakuasi dan Alat Pemadam Api seperti pada gambar 5.4 dan 5.5.



Gambar 5.4 Rancangan *Output* Rambu Jalur Evakuasi



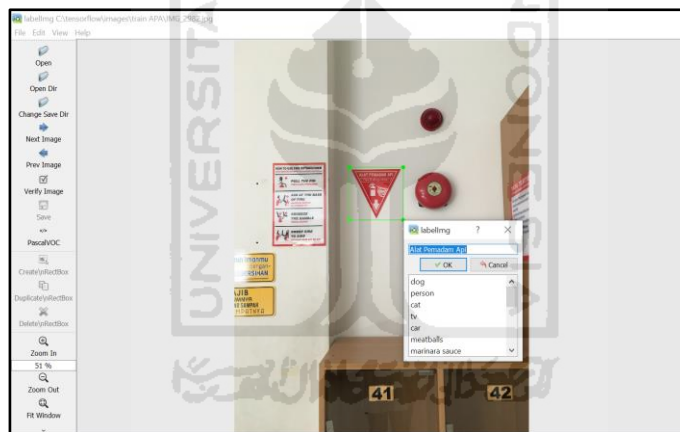
Gambar 5.5 Rancangan *Output* Rambu Alat Pemadam Api

Gambar 5.4 dan 5.5 merupakan rancangan dari *output* sistem yang akan dibuat oleh peneliti. Dimana visualisasi rancangan *output* tersebut peneliti buat dengan kotak yang mana merupakan/menandakan sebuah objek yang disertai dengan label disertai dengan tingkat akurasi dari pendeteksian objek tersebut.

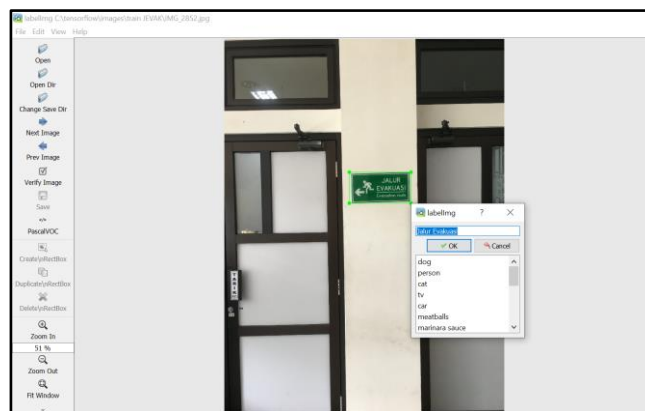
5.4 Preprocessing Data

5.4.1 Pelabelan Gambar

Pelabelan gambar merupakan langkah awal dalam proses *preprocessing* data yang mana dataset akan diberi label sesuai dengan kategori atau *class* objek. Data gambar yang telah terkumpul akan diberikan label satu per satu dengan menggunakan aplikasi *LabelImg*, dengan label sesuai dengan dua objek yaitu “Jalur Evakuasi” dan “Alat Pemadam Api”. Tujuan dari pelabelan gambar yaitu untuk menyimpan informasi gambar tersebut yang selanjutnya disimpan dalam file “.xml” dengan format *PASCAL VOC*. Data gambar rambu jalur evakuasi dan alat pemadam api yang diberi label sejumlah 1500 akan dibagi menjadi data *train* dan data *test* dengan perbandingan 80:20. Berikut gambar di bawah ini merupakan visualisasi pemberian label pada aplikasi *LabelImg*.



Gambar 5.6 Pemberian label rambu alat pemadam api



Gambar 5.7 Pemberian label rambu jalur evakuasi

5.4.2 Konversi Dataset XML ke csv

Hasil *output* pelabelan berupa file dengan format *XML* (*Xtensible Markup Language*), selanjutnya akan dikonversi dari *.xml* menjadi *.csv*. Adapun *script* yang digunakan untuk mengkonversi dataset *XML* menjadi *csv* akan disimpan dengan format file *python* dan akan disimpan dengan nama *xml_to_csv.py*. Setelah peneliti melakukan konversi, file yang telah dikonversi menjadi file dengan format *.csv* akan tersimpan. Berikut di bawah ini adalah perintah yang akan digunakan untuk menjalankan *script* pada *command prompt*:

```
python xml_to_csv.py
```

Gambar 5.8 Perintah menjalankan konversi XML ke csv

Data yang telah tersimpan dalam format *XML* (*Xtensible Markup Language*), selanjutnya dikonversi dalam bentuk file *csv* (*comma separated value*) yang mana hal tersebut dilakukan untuk konversi dataset menjadi berkas *TFRecord*. *Script* yang digunakan untuk mengkonversi *XML* menjadi *csv* terdapat pada lampiran 4. Berikut adalah hasil konversi berupa file data train dan data test dalam format *csv*.

	A	B	C	D	E	F	G	H
1	filename	width	height	class	xmin	ymin	xmax	ymax
2	IMG_2851.jpg	1350	1800	Jalur Evakuasi	508	558	957	771
3	IMG_2852.jpg	1350	1800	Jalur Evakuasi	602	642	899	791
4	IMG_2873.jpg	1350	1800	Jalur Evakuasi	528	462	816	642
5	IMG_2874.jpg	1350	1800	Jalur Evakuasi	446	477	726	654
6	IMG_2875.jpg	1350	1800	Jalur Evakuasi	267	399	599	583
7	IMG_2876.jpg	1350	1800	Jalur Evakuasi	579	679	838	836
8	IMG_2877.jpg	1350	1800	Jalur Evakuasi	426	660	579	813
9	IMG_2878.jpg	1350	1800	Jalur Evakuasi	748	693	991	864
10	IMG_2879.jpg	1350	1800	Jalur Evakuasi	589	322	861	509
11	IMG_2880.jpg	1350	1800	Jalur Evakuasi	575	626	834	752
12	IMG_2881.jpg	1350	1800	Jalur Evakuasi	487	752	681	834
13	IMG_2882.jpg	1350	1800	Jalur Evakuasi	702	648	930	762
14	IMG_2883.jpg	1350	1800	Jalur Evakuasi	624	599	793	734
15	IMG_2884.jpg	1350	1800	Jalur Evakuasi	616	703	857	864
16	IMG_2885.jpg	1350	1800	Jalur Evakuasi	740	728	942	862
17	IMG_2886.jpg	1350	1800	Jalur Evakuasi	699	703	914	838
18	IMG_2887.jpg	1350	1800	Jalur Evakuasi	534	597	732	705
19	IMG_2888.jpg	1350	1800	Jalur Evakuasi	546	577	753	711
20	IMG_2889.jpg	1350	1800	Jalur Evakuasi	546	552	763	699
21	IMG_2890.jpg	1350	1800	Jalur Evakuasi	555	705	824	836
22	IMG_2891.jpg	1350	1800	Jalur Evakuasi	563	668	814	817
23	IMG_2892.jpg	1350	1800	Jalur Evakuasi	573	660	832	809
24	IMG_2893.jpg	1350	1800	Jalur Evakuasi	593	581	857	728

Gambar 5.9 Hasil konversi file csv

Pada konversi *XML* ke *csv* menghasilkan *output* data *csv* seperti pada gambar 5.9. Output tersebut berisi kolom *filename* yaitu nama file gambar, kemudian *width* dan *height* adalah lebar dan tinggi atau ukuran dimensi dari gambar tersebut yaitu detail informasi pada data gambar IMG_2851 memiliki dimensi gambar 1350×1800 , sesuai dengan kolom *width* dan *height* pada data

csv. Kolom *class* merupakan hasil label klasifikasi pada objek dalam hal ini yaitu Jalur Evakuasi dan Alat Pemadam Api. Kolom *xmin* dan *ymin* adalah nilai minimal pixel pada *width* dan *height*, sedangkan *xmax* dan *ymax* merupakan nilai maksimal pixel pada *width* dan *height* dari kotak/box yang yang digunakan untuk menandai sebuah objek sehingga untuk pelabelannya sesuai dengan objek

5.4.3 Konversi Dataset csv ke TFRecord

Pada tahap ini peneliti melakukan konversi dari format file csv menjadi *TFRecord* atau *Tensorflow Record*, hal ini dilakukan karena dalam tahap awala proses training, *tensorflow* akan membaca data yang dimasukkan (data *input*) atau yang disebut dengan *feeding* data dengan fungsi *feed_dictionary*. Dalam proses *feeding* data, informasi dataset yang tersimpan dalam format *TFRecord* akan diambil secara langsung. *Script* yang digunakan untuk meng-*generate* file csv ke *TFRecord* pada data train dan data test terdapat pada lampiran 5. Gambar 5.10 merupakan perintah untuk menjalankan *script* tersebut.

<i>Train</i> →	<code>python generate_tfrecord.py --type=train -- csv_input=data/train_labels.csv -- output_path=data/train.record</code>
<i>Test</i> →	<code>python generate_tfrecord.py --type=test -- csv_input=data/test_labels.csv -- output_path=data/test.record</code>

Gambar 5.10 Perintah menjalankan *script* konversi csv ke *TFRecord*

5.4.4 Konfigurasi Label Map

Konfigurasi *label map* adalah proses pemetaan label yang digunakan dalam pemberian nama objek yang akan dideteksi. Dalam penelitian ini terdapat dua objek yang akan dideteksi sehingga pada *label map* juga terdapat dua item. Konfigurasi *label map* yang digunakan dalam penelitian ini seperti pada gambar 5.11 berisi nama dan urutan id pada masing-masing item sesuai dengan nama dan urutan saat dilakukannya proses *labeling*. Penelitian ini menggunakan 2 item label dengan nama item Jalur Evakuasi dengan id= 1 dan Alat Pemadam Api dengan id=2. *Label Map* tersebut disimpan pada berkas dengan format “.pbtxt” yang selanjutnya dibutuhkan pada saat konfigurasi pipeline. Berikut merupakan *script label map* yang juga terdapat pada lampiran 6.

```

item {
  id: 1
  name: 'Alat Pemadam Api'
}
item {
  id: 2
  name: 'Jalur Evakuasi'
}

```

Gambar 5.11 Konfigurasi Label Map

5.5 Konfigurasi *Object Detection Training Pipeline*

Penelitian ini menggunakan konfigurasi model *SSD with Mobilenet v1*. SSD tidak melalui tahapan proses pembuatan proposal dan *feature resampling* tetapi dengan melalui tahapan merangkum semua perhitungan dalam satu jaringan sehingga hal ini yang membuat SSD bekerja dengan algoritma yang relatif sederhana. Metode SSD mudah dilatih bahkan dapat langsung diintegrasikan ke sistem. Pada penelitian ini konfigurasi *pipeline* dilakukan pada beberapa parameter.

Konfigurasi *pipeline* perlu dilakukan karena *tensorflow* menggunakan *protobuf* sehingga berfungsi untuk mengkonfigurasi proses *training* dan evaluasi. Peneliti mendefinisikan jumlah kelas pada *script num_classes: 2* yang berarti bahwa jumlah kelas berjumlah dua yaitu rambu jalur evakuasi dan rambu alat pemadam api. Konfigurasi pada penelitian ini menggunakan *batch_size: 2*, yang berarti jumlah sampel dari data yang disebarkan ke dalam *Neural Network* adalah berjumlah 2 dimana sampel tersebut akan diambil secara random dari semua sampel dataset. Pada penelitian ini banyaknya *batch_size* disesuaikan dengan kemampuan perangkat peneliti. Penelitian ini menggunakan *num_steps: 100.000* yang berarti jumlah langkah maksimal dalam proses pelatihan adalah sebesar 100.000 *step*. File konfigurasi pipeline tersimpan dengan nama *rambu_v1.config* yang secara keseluruhan dapat dilihat pada lampiran 7. File konfigurasi pipeline tersebut selanjutnya akan digunakan pada proses *training*.

5.6 Arsitektur Jaringan

Pada penelitian ini peneliti menggunakan input gambar berukuran 300x300 pixel. Sistem jaringan yang digunakan dalam penelitian ini yaitu VGG-16, jaringan syaraf *convolutional* yang dibuat oleh grup peneliti dalam kompetisi

Image Net Large Scale Visual Recognition Challenge (ILSVRC) pada tahun 2014 yang dilatih pada lebih dari satu juta gambar. Jaringan ini memiliki kedalaman 16 lapis atau *layer* yang dapat mengklasifikasikan gambar ke dalam 1000 kategori objek. Sehingga jaringan ini telah mempelajari berbagai representasi fitur dari banyak gambar. Jaringan VGG-16 ini tersedia secara default oleh API yang disediakan oleh *Tensorflow*. Pada gambar 5.6 di bawah ini merupakan arsitektur jaringan CNN yang digunakan pada penelitian ini. Sedangkan untuk visualisasi arsitektur jaringan pada penelitian ini terdapat pada gambar 5.12.



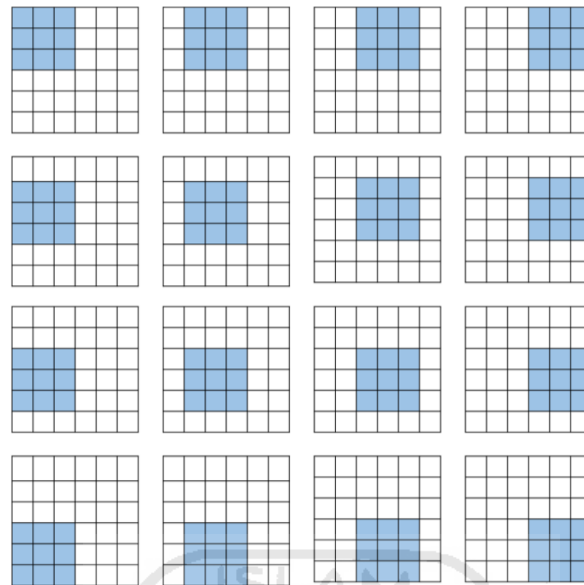
Gambar 5.12 Jaringan VGG-16 Net

Di bawah ini adalah beberapa proses dalam arsitektur jaringan pada penelitian ini. Oleh karena input berukuran 300×300 pixel sehingga terbatasnya kemampuan peneliti dalam memvisualisasikan proses kalkulasi, maka peneliti hanya menampilkan proses yang bersifat ilustrasi. Ukuran pixel ilustrasi yang digunakan yaitu 6×6 pixel dengan nilai sampel kode warna secara acak. Gambar 5.13 merupakan sampel matriks berukuran 6×6 yang digunakan sebagai ilustrasi *input feature map*.

0	1	2	9	8	5
5	8	9	0	2	0
7	2	5	8	5	4
6	2	8	2	7	4
3	3	1	9	3	1
1	7	8	5	1	3

Gambar 5.13 Matriks Sampel Kode Warna





Gambar 5.15 Ilustrasi Pergerakan Kernel

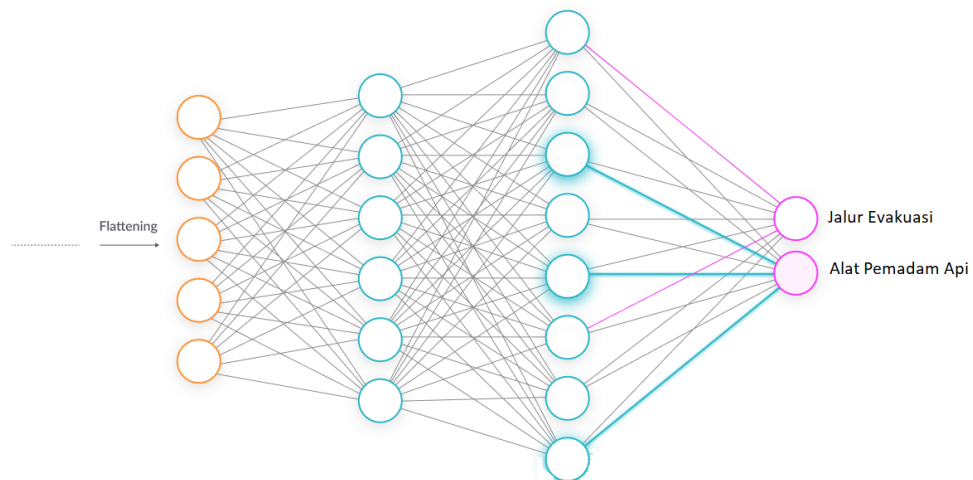
Proses pergerakan pada gambar 5.15 di atas ini disebut dengan *sliding window*, yaitu dalam proses perhitungan pada konvolusi diawali dari sudut kiri atas dan terus berjalan hingga sudut kanan bawah. Jumlah pergerakan kernel sebanyak 16 kali, dimulai dari sudut kiri merupakan posisi ke-1 sampai dengan sudut kanan bawah merupakan posisi ke-16. Perkalian input dan kernel akan menghasilkan matriks output berukuran 4×4 . Pada masing-masing posisi perhitungan konvolusi tersebut menggunakan rumus 3.5. Berbagai tipe kernel yang telah disebutkan di atas digunakan untuk melakukan variasi konvolusi sehingga menghasilkan output beberapa activation map.

5.6.2 Fungsi Aktivasi

Tahap ini merupakan perhitungan nilai yang digunakan untuk mencari nilai non-linier pada nilai hasil konvolusi. Fungsi Aktivasi yang digunakan dalam mendeteksi rambu adalah ReLu (*Rectified Linier Unit*). Fungsi aktivasi ReLU bekerja dengan cara mencari nilai maksimum pada sebuah sel dari hasil konvolusi yang terbentuk. Adapun Rumus ReLu yaitu $f(x) = \max(x, 0)$. Pada rumus tersebut x merupakan input neuron atau node, angka 0 berugas sebagai unit linier yang akan dikoreksi jika input bernilai kurang dari 0. Sehingga fungsi aktivasi ReLU digunakan untuk menghilangkan nilai negatif dari hasil proses konvolusi.







Gambar 5.19 Ilustrasi *Fully Connected Layer*

Pada gambar 5.19 di atas merupakan proses *fully connected layer* yang menggunakan 2 *hidden layer*. Proses dimulai pada input hasil *flattening* lalu melewati 2 *hidden layer* dengan beberapa node yang kemudian menghasilkan klasifikasi objek. Metode atau dalam hal ini fungsi aktivasi dalam *fully connected layer* yang digunakan untuk melakukan klasifikasi ialah *Softmax*. *Softmax* akan menghasilkan *output* berupa peluang prediksi masing-masing kelas untuk sebuah objek dan nilai peluang tertinggi merupakan kelas yang digunakan sebagai output prediksi.

Proses *backpropagation* terjadi pada *fully connected layer* ketika proses training jaringan syaraf tiruan. Langkah pertama pada *backpropagation* yaitu melakukan inisiasi pada masing-masing edge, kemudian dijalankan secara maju (*forward*). Pada proses *forward pass*, input menuju *output layer* yang akan menghasilkan prediksi *output*. Hasil tersebut akan dibandingkan dengan target yang biasa disebut *loss function*. *Backpropagation* dimulai dari *feed forward* kemudian hasil dari *feed forward* akan mendapatkan nilai total *error* yang kemudian proses berbalik ke belakang sehingga disebut proses *backward pass* dimana pada proses ini melakukan perhitungan untuk mendapatkan nilai *error* yang lebih kecil dari hasil *feed forward*, sehingga pada hasil akhir jaringan syaraf tiruan bisa memiliki *error* yang terkecil dan dapat melakukan klasifikasi secara tepat.

5.7 Training Model

Tahapan utama dari *neural network* adalah training model, dimana semua dataset akan dilatih atau di-*training* untuk mengenali dan mempelajari pola rambu jalur evakuasi dan rambu alat pemadam api. Semua proses yang ada pada jaringan *CNN* akan bekerja pada proses training model. Peneliti menggunakan *pre-trained model* *ssd_mobilenet_v1_coco* dengan kecepatan mencapai 30/millisecond. Tujuan akhir dari proses *training* yaitu didapatkan model yang dapat mendeteksi objek dengan tingkat akurasi yang tinggi. Proses training dimulai dengan menjalankan perintah sebagai berikut.

```
pythontrain.py --logtostderr --train_dir=training --
pipeline_config_path=training/rambu_v1.config
```

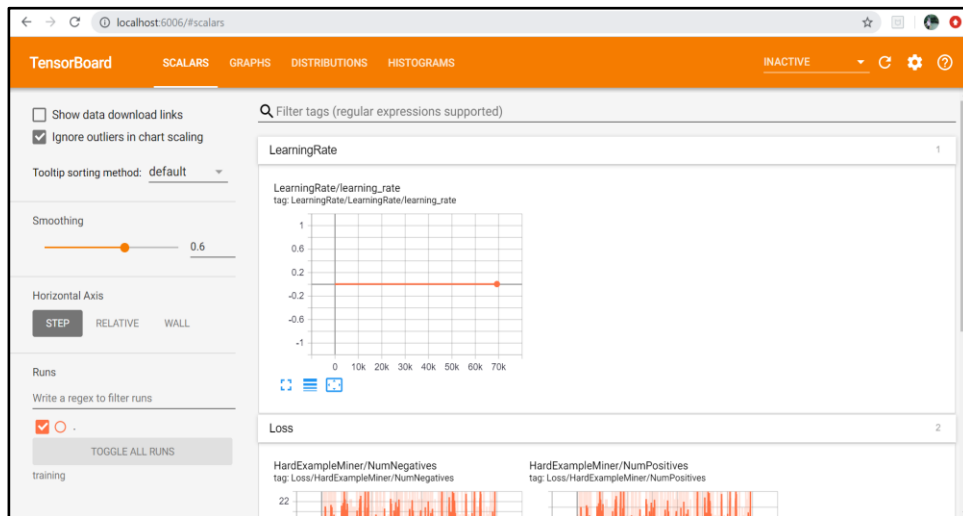
Gambar 5.20 Perintah *Training Model*

Gambar 5.20 di atas adalah perintah untuk menjalankan proses *training* yaitu pada *script train.py* yang terdapat pada lampiran 8. Model hasil *training* akan tersimpan pada folder *training* sesuai dengan *syntax train_dir=training*. Konfigurasi pipeline yang digunakan berada pada folder *training* dengan nama file *rambu_v1.config*. Setelah perintah tersebut dijalankan maka akan muncul proses *training* dimulai dari step 0. Pemantauan proses *training* dapat menggunakan module yang telah tersedia pada *Tensorflow* yaitu *Tensorboard*. Berikut ini adalah perintah untuk menjalankan *Tensorboard* melalui *command prompt* :

```
#Train tensorboard
tensorboard --logdir=training
```

Gambar 5.21 Perintah untuk melihat *Tensorboard*

Setelah menjalankan perintah pada gambar 5.21 di atas, untuk memunculkan *tensorboard* maka harus membuka alamat *localhost:6006* pada *browser*. *Tensorboard* memanggil *file checkpoint* yang di hasilkan selama proses *training* di folder *training*. Pada *Tensorboard* peneliti dapat memantau grafik-grafik yang ada ketika proses *training* berjalan seperti grafik *total_loss* dan grafik *global_step*. Berikut gambar 5.22 merupakan tampilan awal dari *tensorboard*.



Gambar 5.22 Tampilan Awal Tensorboard

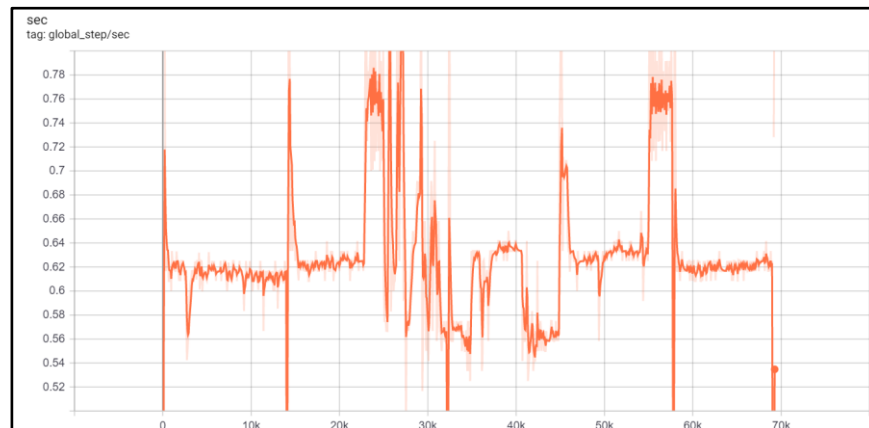
5.8 Model Hasil Training

5.8.1 Hasil Training Steps

Training model dengan menggunakan jumlah maksimal step sebanyak 100.000 dan ukuran *batch* 2 diestimasikan akan membutuhkan waktu 48 jam. Karena pada step 100.000 belum tentu merupakan model yang terbaik maka peneliti melakukan *trial and error* dengan menggunakan model checkpoint. Berikut ini merupakan proses training beserta grafik *step/sec* pada training.

```
C:\Windows\System32\cmd.exe
INFO:tensorflow:Saving checkpoint to path training\model.ckpt
I0419 06:32:45.173136 3032 supervisor.py:1117] Saving checkpoint to path training\model.ckpt
INFO:tensorflow:global step 69276: loss = 3.9196 (1.277 sec/step)
I0419 06:32:45.223003 1000 learning.py:507] global step 69276: loss = 3.9196 (1.277 sec/step)
INFO:tensorflow:global step 69277: loss = 1.7712 (1.405 sec/step)
I0419 06:32:46.637224 1000 learning.py:507] global step 69277: loss = 1.7712 (1.405 sec/step)
INFO:tensorflow:Saving checkpoint to path training\model.ckpt
I0419 06:32:47.471993 3032 supervisor.py:1117] Saving checkpoint to path training\model.ckpt
INFO:tensorflow:global step 69278: loss = 3.0150 (1.339 sec/step)
I0419 06:32:47.997588 1000 learning.py:507] global step 69278: loss = 3.0150 (1.339 sec/step)
INFO:tensorflow:global step 69279: loss = 3.4005 (1.383 sec/step)
I0419 06:32:49.389867 1000 learning.py:507] global step 69279: loss = 3.4005 (1.383 sec/step)
INFO:tensorflow:Saving checkpoint to path training\model.ckpt
I0419 06:32:50.346312 3032 supervisor.py:1117] Saving checkpoint to path training\model.ckpt
INFO:tensorflow:global step 69280: loss = 1.4415 (1.542 sec/step)
I0419 06:32:50.978621 1000 learning.py:507] global step 69280: loss = 1.4415 (1.542 sec/step)
INFO:tensorflow:global step 69281: loss = 3.0198 (1.251 sec/step)
I0419 06:32:52.248229 1000 learning.py:507] global step 69281: loss = 3.0198 (1.251 sec/step)
INFO:tensorflow:Saving checkpoint to path training\model.ckpt
I0419 06:32:52.483599 3032 supervisor.py:1117] Saving checkpoint to path training\model.ckpt
INFO:tensorflow:global step 69282: loss = 2.2486 (1.255 sec/step)
I0419 06:32:53.512848 1000 learning.py:507] global step 69282: loss = 2.2486 (1.255 sec/step)
INFO:tensorflow:Saving checkpoint to path training\model.ckpt
I0419 06:32:54.573015 3032 supervisor.py:1117] Saving checkpoint to path training\model.ckpt
INFO:tensorflow:global step 69283: loss = 0.9185 (1.291 sec/step)
I0419 06:32:54.822349 1000 learning.py:507] global step 69283: loss = 0.9185 (1.291 sec/step)
INFO:tensorflow:global step 69284: loss = 1.5198 (1.539 sec/step)
I0419 06:32:56.364227 1000 learning.py:507] global step 69284: loss = 1.5198 (1.539 sec/step)
INFO:tensorflow:Saving checkpoint to path training\model.ckpt
I0419 06:32:56.618547 3032 supervisor.py:1117] Saving checkpoint to path training\model.ckpt
```

Gambar 5.23 Log Training Step Proses



Gambar 5.24 Grafik *Global Training Step*

Waktu dalam setiap step dapat dilihat pada gambar 5.24 yang mana dapat diketahui bahwa rata-rata waktu training setiap step adalah sekitar 0,62 detik. Gambar 5.23 merupakan hasil pencatatan step saat proses training beserta dengan nilai *loss*-nya. Peneliti telah melakukan percobaan pada beberapa step yang telah tersimpan dalam bentuk *checkpoint*, namun model tersebut belum dapat mendeteksi kedua objek pada suatu video. Step 69.284 memiliki jumlah step yang cukup banyak serta nilai *loss function* yang sangat kecil sehingga peneliti mengambil model tersebut sebagai model yang digunakan dalam mendeteksi rambu jalur evakuasi dan rambu alat pemadam api. Step 69.284 telah tersimpan pada folder *training* sehingga model tersebut dapat diexport yang kemudian digunakan untuk mendeteksi objek. Model tersebut memiliki nilai *loss* yang rendah yaitu 1,5198.

5.8.2 *Export Model*

Dalam folder training terdapat beberapa *file checkpoint* yang tersimpan, maka peneliti akan meng-ekspor file *checkpoint* sesuai dengan step yang memiliki nilai *loss* rendah. Proses meng-ekspor model merupakan tujuan utama dalam training neural network ini karena dibutuhkan sebuah model untuk mendeteksi objek pada sebuah video. Adapun perintah untuk meng-ekspor model adalah sebagai berikut.

```

Python export_inference_graph.py --input_type
image_tensor --pipeline_config_path
training/rambu_v1.config --
trained_checkpoint_prefix training/model.ckpt-
69284 --output_directory rambu_baru

```

Gambar 5.25 Perintah *Export Model*

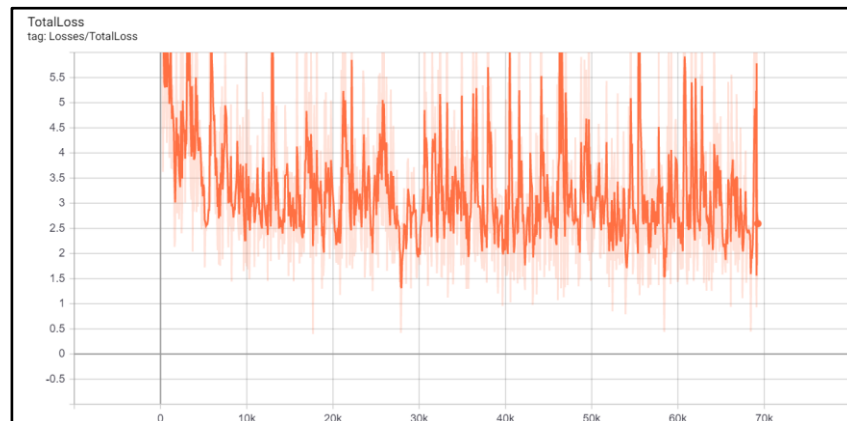
Perintah pada gambar 5.25 berfungsi untuk menjalankan *script export_inference_graph.py* yang terdapat pada lampiran 9 dengan konfigurasi pipeline *rambu_v1.config* yang terdapat pada lampiran 7 serta *model checkpoint* 69284 yang akan menghasilkan folder *rambu_baru*. Setelah *export* selesai maka hasil model training akan muncul pada folder *rambu_baru* seperti pada gambar 5.26. Model yang dimaksudkan pada *Tensorflow API* adalah berupa file *checkpoint* hasil training/pelatihan dan data tensor graph yang dimuat pada berkas berekstensi *protobuf* “.pb”. Setelah dilakukannya beberapa proses dalam algoritma *Convolutional Neural Network* (CNN) dihasilkan sebuah model yang dapat digunakan dalam pendeteksian objek.

Name	Date modified	Type	Size
saved_model	4/20/2020 12:44 PM	File folder	1 KB
checkpoint	4/20/2020 12:44 PM	File	22,196 KB
frozen_inference_graph.pb	4/20/2020 12:44 PM	PB File	21,712 KB
model.ckpt.data-00000-of-00001	4/20/2020 12:44 PM	DATA-00000-OF-000...	9 KB
model.ckpt.index	4/20/2020 12:44 PM	INDEX File	1,046 KB
model.ckpt.meta	4/20/2020 12:44 PM	META File	4 KB
pipeline.config	4/20/2020 12:44 PM	XML Configuration File	4 KB

Gambar 5.26 Model Hasil Training

5.8.3 Total Loss

Pada saat proses training berlangsung, semua proses terekam pada *tensorboard*. Seluruh proses *training* seperti *training loss/accuracy*, *validation loss/accuracy*, *training step/second*, dan lain-lain akan tersimpan dan akan tervisualisasikan dalam sebuah grafik. Hasil nilai total loss merupakan nilai *error* yang berasal dari sisa nilai akurasi pada setiap step. Pada hasil pendeteksian menampilkan nilai tingkat akurasi dimana nilai *error*-nya merupakan hasil pengurangan *probability* 100% dikurangi dengan tingkat akurasi setiap hasil deteksinya. Berikut ini adalah grafik *Total Loss* yang ditampilkan dalam *tensorBoard*.



Gambar 5.27 Grafik Total Loss

Total Loss merupakan jumlah *error* yang terhitung saat *training*. Gambar 5.27 adalah grafik *total loss* yang dihasilkan dalam proses *training*. Grafik *total loss* yang ditampilkan dalam *tensorboard* Gambar 5.27 ini merupakan visualisasi *loss function* selama melakukan *training* model. Grafik tersebut menunjukkan adanya penurunan nilai *total loss* namun tidak secara signifikan. Dari awal proses *training* dapat diketahui bahwa nilai *total loss* sangat tinggi. Namun, semakin banyak *step* membuat nilai *total loss* mulai menurun. Hal ini terlihat dari nilai *total loss* pada *step* 69.284 yang sangat rendah yaitu 1,5198. Dalam hal ini, semakin kecil nilai *loss* maka model yang digunakan akan mendeteksi objek dengan baik.

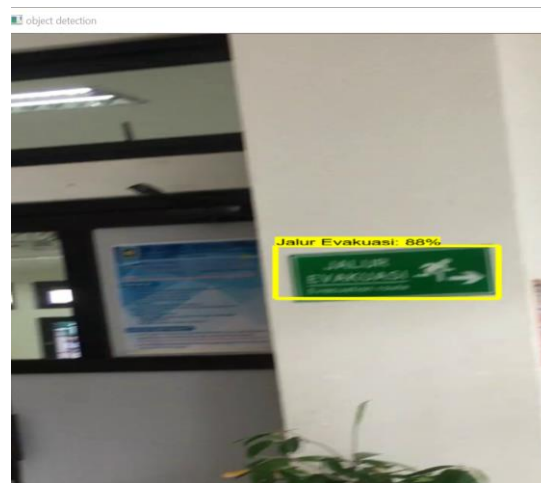
5.9 Hasil Pendeteksian Objek pada Video

Sistem akan dicoba untuk mendeteksi objek rambu K3 jalur evakuasi dan alat pemadam api dengan menggunakan video. Ketika mendeteksi objek berupa video, model akan mendeteksi frame setiap detiknya dan menampilkan hasil klasifikasi berupa kotak beserta dengan tingkat akurasi dari hasil klasifikasi objek tersebut. Uji coba model akan menggunakan python dengan *script* *rambu_video.py* yang terdapat pada lampiran 10. Berikut ini adalah perintah pada *command prompt* untuk menjalankan *script* tersebut.

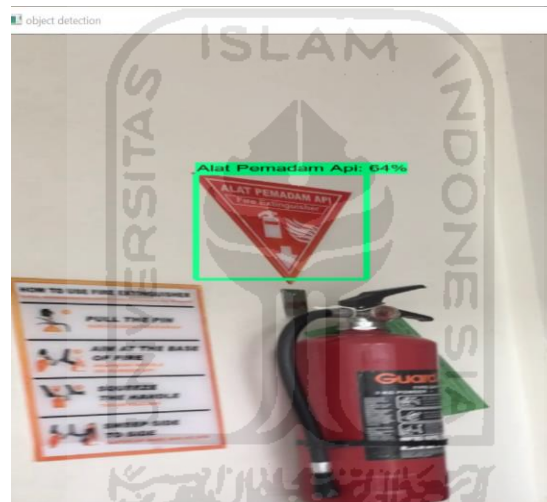
```
C:\Windows\System32\cmd.exe
(tensorflow_cpu) C:\tensorflow\models\research\object_detection>python rambu_video.py
```

Gambar 5.28 Perintah uji coba model





Gambar 5.31 Hasil Deteksi Rambu Jalur Evakuasi pada Video Kedua



Gambar 5.32 Hasil Deteksi Rambu Alat Pemadam Api pada Video Kedua

Gambar 5.31 dan 5.32 merupakan hasil deteksi pada video kedua yang memiliki tingkat akurasi 88% pada Rambu K3 Jalur Evakuasi dan 64% pada Rambu K3 Alat Pemadam Api. Hasil deteksi pada gambar di atas memperlihatkan bahwa tingkat akurasi akan berubah sesuai dengan tingkat kualitas gambar pada *frame* tersebut terdeteksi. Secara keseluruhan model ini menghasilkan tingkat akurasi mulai dari 50% hingga 97%.

BAB VI

PENUTUP

6.1 Kesimpulan

Setelah melalui tahap pembahasan, peneliti dapat mengambil beberapa kesimpulan pada penelitian ini, yaitu :

1. Implementasi *deep learning* untuk mendeteksi objek pada citra rambu K3 yaitu menggunakan sistem jaringan *VGG16-Net*, jaringan tersebut merupakan jaringan *Convolutional Neural Network* (CNN) yang telah dikembangkan. Arsitektur jaringan yang digunakan secara umum terbagi menjadi *layer input*, *convolutional*, *pooling*, *fully connected*, dan *output*. Pada proses *training* dilakukan perulangan pada *layer convolutional*, *pooling*, dan *fully connected* sehingga mendapatkan model yang optimal. Proses training menghasilkan model yang dapat digunakan untuk mendeteksi rambu K3 pada video.
2. Model yang digunakan dalam penelitian ini ialah *pre-trained* model *SSD-mobilenet V1*. Sehingga model yang terbentuk pada hasil training memiliki jumlah *step* 69.284 dengan 2 *batch size* serta memiliki nilai *loss* yang cukup rendah yaitu 1,5198.
3. Hasil dari pendeteksian rambu K3 jalur evakuasi dan alat pemadam api dengan menggunakan jaringan *Convolutional Neural Network* (CNN) memiliki tingkat akurasi yang cukup tinggi yaitu berkisar antara 50-97%.

6.2 Saran

Berdasarkan hasil penelitian maka peneliti akan memberikan beberapa saran untuk penelitian selanjutnya, diantaranya sebagai berikut:

1. Mengembangkan sistem dengan menambahkan dataset agar dapat mendeteksi seluruh Rambu Kesehatan dan Keselamatan Kerja (K3) serta membuat sebuah aplikasi yang dapat memberi petunjuk keberadaan rambu-rambu K3 untuk memudahkan para pekerja yang memiliki kebutuhan khusus.

2. Dalam pengumpulan dataset, peneliti selanjutnya perlu mengumpulkan dan menggunakan gambar dari segala arah yang mana akan digunakan dalam proses training, sehingga dapat menghasilkan akurasi yang tinggi.
3. Meningkatkan spesifikasi perangkat keras seperti *Random Access Memory (RAM)* yang tinggi dan menggunakan *Graphics Processing Unit (GPU)* sehingga dapat mempercepat kalkulasi serta mampu menjalankan kalkulasi dengan jumlah input citra yang lebih banyak dan resolusi yang tinggi.
4. Menggunakan foto-foto yang memiliki diversifikasi jarak yg berbeda-beda



DAFTAR PUSTAKA

- Rambu - rambu K3. (2015, Juli 30). Retrieved from dukuhjayamandiri.wordpress.com:
<https://dukuhjayamandiri.wordpress.com/2015/07/30/rambu-rambu-k3/>
- BPPT: Mitigasi Bencana Dengan Teknologi. (2018, November 27). Retrieved from <https://www.bppt.go.id>.
- Ahmad, A. (2017). Mengenal Artificial Intelligence, Machine Learning, Neural Network, dan Deep Learning. *Yayasan Cahaya Islam, Jurnal Teknologi Indonesia*.
- Al-Azzoa, F., Taqia, A. M., & Milanovab, M. (2018). Human Related-Health Actions Detection using Android Camera based on TensorFlow Object Detection API. *International Journal Of Advanced Computer Science And Applications*, 9(10), 9 -23.
- Ardiyansyah, F. (2014). Implementasi Pattern Recognition Pada Pengenalan Monumen-Monumen Bersejarah Di Kota Bandung Menggunakan Augmented Reality Berbasis Android. *Jurnal Ilmiah Komputer dan Informatika (KOMPUTA)*, 2-3.
- Balza, A., & Kartika, F. (2005). *Teknik Pengolahan Citra Digital menggunakan Delphi*. Yogyakarta: Ardi Publishing.
- Dahria, M. (2008). Kecerdasan Buatan (Artificial Intelligence). *Jurnal SAINTIKOM Vol. 5*, 185-196.
- Darma, P. (2010). *Citra Digital dan Ekstraksi Fitur*. Yogyakarta: Graha ilmu.
- Dewi, S. R. (2018). *Deep Learning Object Detection Pada Video Menggunakan Tensorflow dan Convolutional Neural Network*. Yogyakarta: Universitas Islam Indonesia.
- Dharma, S., Putera, A., & Ardana, P. (2011). Artificial Neural Networks Untuk Pemodelan Curah Hujan-Limpasan Pada Daerah Aliran Sungai (Das) Di Pulau Bali. *Jurnal Bumi Lestari, Volume 11 No. 1*, 9-22.
- Fadlisyah, S. (2007). *Computer Vision dan Pengolahan Citra*. Yogyakarta: ANDI.

- Fikrieabdillah. (2016). *Penggunaan Deep Learning untuk Prediksi Churn pada Jaringan Telekomunikasi Mobile*. Bandung: Telkom Bandung 40257.
- G, E. E., Diah, Y. M., & Zen, M. K. (2017). Pengaruh Keselamatan Dan Kesehatan Kerja Terhadap Kinerja Karyawan Pt. Pertamina Ep Asset 2 Prabumulih. *Jurnal Ilmiah Manajemen Bisnis Dan Terapan Tahun XIV No 2, Oktober 2017*, 103 - 118.
- Geron, A. (2017). *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. Sebastopol: O'Reilly Media, Inc, USA.
- Goodfellow, I. B. (2016). *Deep Learning*. The MIT Press.
- Hamida, U. (2014). Penggunaan Artificial Neural Network (Ann) Untuk Memodelkan Kebutuhan Energi Untuk Transportasi. *Jurnal Teknologi Manajemen, Vol. 12, No.2*.
- Hermawan, A. (2006). *Jaringan Syaraf Tiruan dan Aplikasinya*. Yogyakarta: Andi.
- Irianto, S. Y. (2016). *Analisa Citra Digital Dan Content Bases Image Retrieval*. Lampung: CV. Anugerah Utama Raharja.
- Jalled, F., & Voronkov, I. (2016). *Object Detection Using Image Processing*. Ithaca, New York: Cornell University Library.
- Johnson, R. A., & Bhattacharyya, G. K. (2010). *Statistics Principles & Methods*. USA: John Wiley & Sons.
- Khan, S., Rahmani, H., Shah, S. A., & Bennamoun, M. (2018). *A Guide to Convolutional Neural Networks for Computer Vision*. DOI 10.2200/S00822ED1V01Y201712COV015: Morgan & Claypool Publishers.
- Kumar, K., & Haynes, J. D. (2003). Forecasting Credit ratings Using an ANN and Statistitcal Techniques. *International journal of Business Studies*, 91-108.
- Kusumadewi, S. (2004). *Membangun Jaringan Syaraf Tiruan*. Yogyakarta: Graha Ilmu.

- Kusumanto, R. D., & Tompunu, A. N. (2011). Pengolahan Citra Digital Untuk Mendeteksi Obyek Menggunakan Pengolahan Model Normalisasi RGB. *Semantik*, 1(1).
- Le, H. H. (2015). *Division Intro*. Retrieved from Academia: https://www.academia.edu/27143145/Division_intro
- Lecun, Y., Bengio, Y., & Geoffrey, H. (2015). *Deep Learning*. <https://doi.org/10.1038/nature14539>.
- Liu, W. A.-Y. (2016). *SSD: Single Shot MultiBox Detector*. Retrieved from arXiv: <https://arxiv.org/pdf/1512.02325.pdf>
- Matlab. (n.d.). *Convolutional Neural Network: 3 Things You Need to Know*. Retrieved April 25, 2020, from MathWorks: Deep Learning Solution: <https://www.mathworks.com/solutions/deep-learning/convolutionalneural-network.html>
- Milos, E., Kolesao, A., & Sesok, D. (2018). Traffic Sign Recognition using Convolutional Neural Networks. *Mokslas – Lietuvos ateitis / Science – Future of Lithuania 2018, Volume 10 ISSN 2029-2341.*, 1-5.
- Mondy, R., Wayne, & Robert, M. N. (2005). *Human Resource Management Edisi ke Sepuluh*. Prentice Hall: Cornell University.
- Munir, R. (2004). *Pengolahan citra digital dengan pendekatan algoritmik*. Bandung: Informatika.
- Munir, R. (2004). *Pengolahan citra digital dengan pendekatan algoritmik*. Bandung: Institut Teknologi Bandung Press.
- Nafi'iyah, N. (2015). Algoritma Kohonen dalam Mengubah Citra Graylevel Menjadi Citra Biner. *Jurnal Ilmiah Teknologi dan Informasia ASIA (JITIKA) Vol.9, No.2, Agustus 2015 ISSN: 0852-730X*, 49-55.
- Novyantika, R. (2018). *Deteksi Tanda Nomor Kendaraan Bermotor Pada Media Streaming Dengan Algoritma Convolutional Neural Network Menggunakan Tensorflow*. Yogyakarta: Universitas Islam Indonesia.

- Nurhikmat, T. (2018). *Implementasi Deep Learning Untuk Image Classification Menggunakan Algoritma Convolutional Neural Network (CNN) Pada Citra Wayang Golek*. Yogyakarta: Universitas Islam Indonesia.
- Nurmila, N., Sugiharto, A., & Sarwoko, E. A. (2010). Algoritma Back Propagation Neural Network Untuk Pengenalan Pola Karakter Huruf Jawa. *Jurnal Masyarakat Informatika*, 1.
- Pamungkas, A. (2016, Juni 08). *Model Ruang Warna Pengolahan Citra*. Retrieved from Pemrograman Matlab: <https://pemrogramanmatlab.com>
- Pannu, A. &. (2015). Artificial Intelligence and its Application in Different Areas. *International Journal of Engineering and Innovative Technology (IJEIT)*, Volume 4, ISSN, 2277-3754.
- Pathak, A. R., Pandey, M., & Rautaray, S. (2018). Application of Deep Learning for Object Detection. *International Conference on Computational Intelligence and Data Science (ICCIDS 2018)* (pp. 1706-1717). India: Procedia Computer Science 132 (2018).
- Perkovic, L. (2012). *Introduction to Computing Using Python : An Application Development Focus*.
- Pham, D. (1994). *Neural Network for Chemical Engineers*. Amsterdam: Elsevier Press.
- Prabaningrum, A. (2019). *Implementasi Deep Learning Object Detection Pada Citra Sepatu Merek Adidas Dan Nike Dengan Menggunakan Algoritma Convolutional Neural Network Dan Tensorflow*. Skripsi jurusan Statistika Universitas Islam Indonesia.
- Prasetya, D. A. (2012). Deteksi Wajah Metode Viola Jones Pada OpenCV Menggunakan Pemrograman Python. *Simposium Nasional RAPI XI FT UMS, E18-E23*.
- Puspitaningrum, D. (2006). *Pengantar Jaringan Syaraf Tiruan*. Yogyakarta: Penerbit Andi.
- Putra, D. (2010). *Pengolahan Citra Digital*. Yogyakarta: Andi Offset.

- Putra, I. (2016). Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) pada Caltech 101. *Jurnal Teknik ITS Vol. 5, No. 1, (2016) ISSN: 2337-3539*, A65 - A69.
- Rahman, N. (2017, Mei 7). *What is the benefit of using average pooling rather than max pooling*. Retrieved from Quora: <https://www.quora.com/What-is-the-benefit-of-using-average-poolingrather-than-max-pooling>
- Rizka, M. (2020, Februari 13). *Tanggulangi Bencana, Save The Chidren Gandeng Google dan LPBI NU*. Retrieved from <https://jabarnews.com>.
- Santra, B. K. (2012). Genetic Algorithm and Confusion Matrix for Document Clustering. *International Journal of Computer Science Issues*.
- Shafira, T. (2018). *Implementasi Convolutional Neural Networks Untuk Klasifikasi Citra Tomat Menggunakan Keras*. Yogyakarta: Universitas Islam Indonesia.
- Sholihah, N. (2018, Juni 25). *Fully-Connected Layer CNN dan Implementasinya*. Retrieved from Universitas Gadjah Mada Menara Ilmu Machine Learning: <http://machinelearning.mipa.ugm.ac.id/2018/06/25/fully-connected-layer-cnn-dan-implementasinya/>
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from. *Journal of Machine Learning Research 15 (2014)*, 1929-1958.
- Stutz, D. (2014). *Seminar Report: Understanding Convolutional Neural Networks*. Aachen, Jerman: Fakultät für Mathematik, Informatik und Naturwissenschaften Lehr- und Forschungsgebiet Informatik VIII.
- Sugandi, D. (2003). *Keselamatan Kerja dan Pencegahan Kecelakaan Kerja (Bunga*. Semarang: Badan Penerbit Universitas Diponegoro.
- Sugiyono. (2007). *Statistika Untuk Penelitian*. Bandung: Alfabeta.
- Sutojo, S. (2004). *Membangun Citra Perusahaan*. Jakarta: Damar Mulia Pustaka.
- Sutoyo, T. D., & dkk. (2009). *Teori pengolahan citra digital*. Yogyakarta: Andi.
- Syafitri, N. (2011). Pengenalan Pola untuk Deteksi Uang Koin. *SNTIKI III 2011, ISSN : 2085-9902* , 18-24.

- Tanaka, M. O. (2014). Pattern Recognition (ICPR). *2014 22nd International Conference*, 1526-1531.
- Tanjung, C. A. (2020, Februari 14). *Ada 77.295 Kasus Kecelakaan Kerja di 2019*. Retrieved from <https://finance.detik.com>.
- Tanner, G. (2019, February 01). *Live Object Detection*. Retrieved July 03, 2020, from Towards Data Science: <https://towardsdatascience.com/live-object-detection-26cd50cceffd>
- Taqi, A. M., Al-Azzo, F., Awad, A., & Milanova, M. (2019). Skin Lesion Detection by Android Camera based on SSD- Mobilenet and TensorFlow Object Detection API . *American Journal of Advanced Research*, 5-11.
- Taufiq, I. (2018). *Deep Learning for Detection Vehicle Number Signs Using Convolutional Neural Network Algoritm Using Python and Tensorflow*. Yogyakarta: STMIK AKAKOM.
- Yuli, S. B. (2005). *Manajemen Sumber Daya Manusia*. UMM Press: Malang.
- Yunanto Andhik Ampuh, D. (2016). Kecerdasan Buatan Pada Game Edukasi Untuk Pembelajaran Bahasa Inggris Berbasis Pendekatan Heuristik Similaritas. *Jurnal Sistem Dan Informatika*.
- Zufar, M., & Setiyono, B. (2016). Convolutional neural networks untuk pengenalan wajah secara real-time. *Jurnal Sains Dan Seni Its Vol. 5 No. 2 (2016) 2337-3520 (2301-928X Print)*, A72-A77.

LAMPIRAN

Lampiran 1 Hasil Perhitungan pada *Convolutional Layer*

Posisi Kernel Ke -	Perhitungan pada setiap Sel Output	Output
1	$(0 \times 0) + (1 \times (-1)) + (2 \times 0) + (5 \times (-1)) + (8 \times 5) + (9 \times (-1)) + (7 \times 0) + (2 \times (-1)) + (5 \times 0)$	23
2	$(1 \times 0) + (2 \times (-1)) + (9 \times 0) + (8 \times (-1)) + (9 \times 5) + (0 \times (-1)) + (2 \times 0) + (5 \times (-1)) + (8 \times 0)$	30
3	$(2 \times 0) + (9 \times (-1)) + (8 \times 0) + (9 \times (-1)) + (0 \times 5) + (2 \times (-1)) + (5 \times 0) + (8 \times (-1)) + (5 \times 0)$	-28
4	$(9 \times 0) + (8 \times (-1)) + (5 \times 0) + (0 \times (-1)) + (2 \times 5) + (0 \times (-1)) + (8 \times 0) + (5 \times (-1)) + (4 \times 0)$	-3
5	$(5 \times 0) + (8 \times (-1)) + (9 \times 0) + (7 \times (-1)) + (2 \times 5) + (5 \times (-1)) + (6 \times 0) + (2 \times (-1)) + (8 \times 0)$	-12
6	$(8 \times 0) + (9 \times (-1)) + (0 \times 0) + (2 \times (-1)) + (5 \times 5) + (8 \times (-1)) + (2 \times 0) + (8 \times (-1)) + (2 \times 0)$	-2
7	$(9 \times 0) + (0 \times (-1)) + (2 \times 0) + (5 \times (-1)) + (8 \times 5) + (5 \times (-1)) + (8 \times 0) + (2 \times (-1)) + (7 \times 0)$	28
8	$(0 \times 0) + (2 \times (-1)) + (0 \times 0) + (8 \times (-1)) + (5 \times 5) + (4 \times (-1)) + (2 \times 0) + (7 \times (-1)) + (4 \times 0)$	4

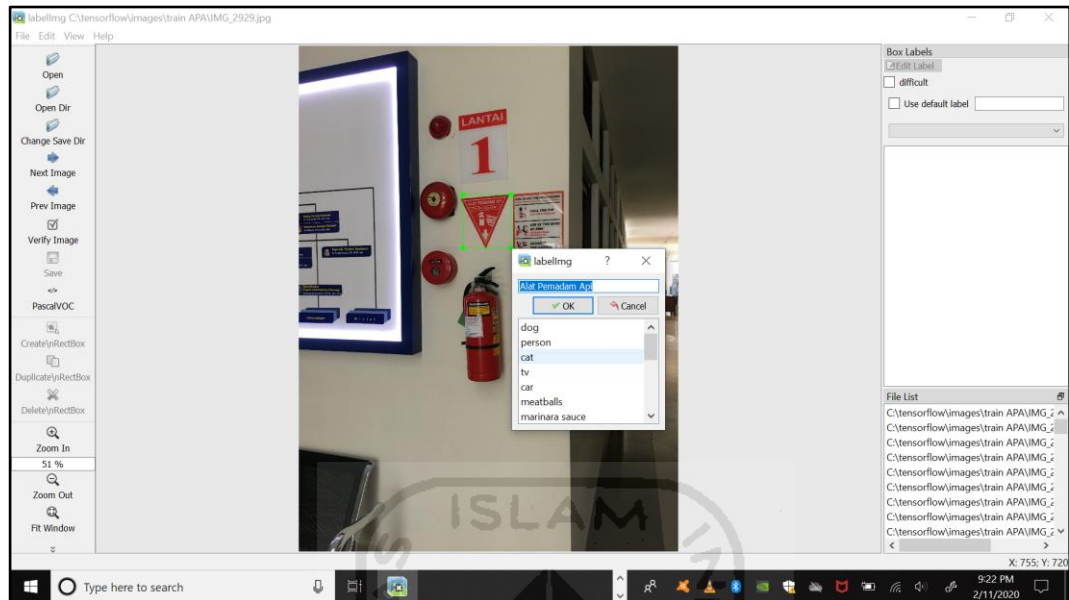
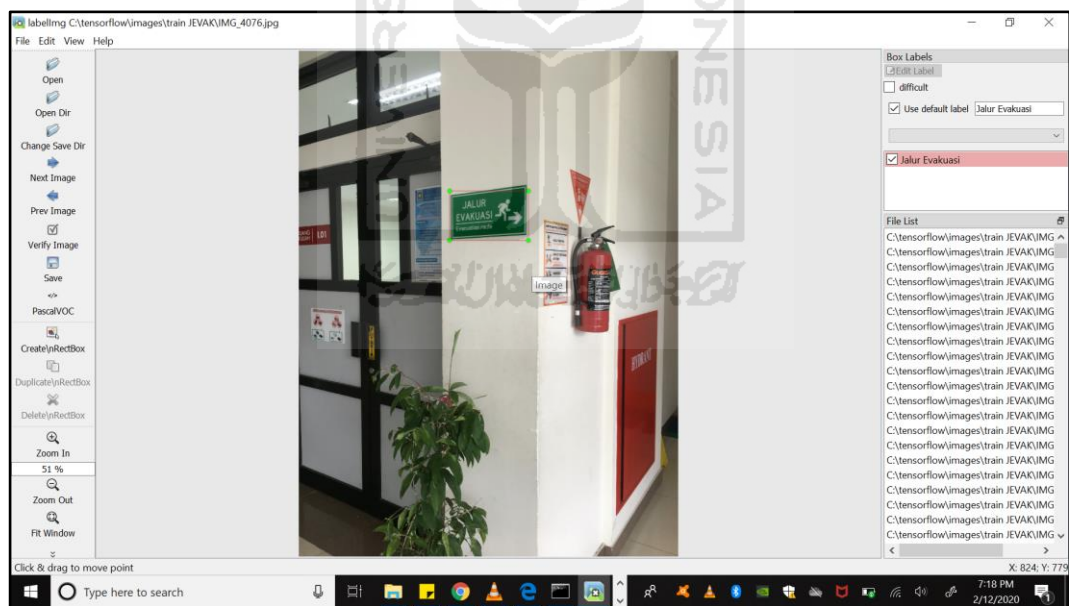
Posisi Kernel Ke -	Perhitungan pada setiap Sel	Output
9	$(7 \times 0) + (2 \times (-1)) + (5 \times 0) + (6 \times (-1)) + (2 \times 5) + (8 \times (-1)) + (3 \times 0) + (3 \times (-1)) + (1 \times 0)$	-9
10	$(2 \times 0) + (5 \times (-1)) + (8 \times 0) + (2 \times (-1)) + (8 \times 5) + (2 \times (-1)) + (3 \times 0) + (1 \times (-1)) + (9 \times 0)$	30
11	$(5 \times 0) + (8 \times (-1)) + (5 \times 0) + (8 \times (-1)) + (2 \times 5) + (7 \times (-1)) + (1 \times 0) + (9 \times (-1)) + (3 \times 0)$	-22
12	$(8 \times 0) + (5 \times (-1)) + (4 \times 0) + (2 \times (-1)) + (7 \times 5) + (4 \times (-1)) + (9 \times 0) + (3 \times (-1)) + (1 \times 0)$	21
13	$(6 \times 0) + (2 \times (-1)) + (8 \times 0) + (3 \times (-1)) + (3 \times 5) + (1 \times (-1)) + (1 \times 0) + (7 \times (-1)) + (8 \times 0)$	2
14	$(2 \times 0) + (8 \times (-1)) + (2 \times 0) + (3 \times (-1)) + (1 \times 5) + (9 \times (-1)) + (7 \times 0) + (8 \times (-1)) + (5 \times 0)$	-23
15	$(8 \times 0) + (2 \times (-1)) + (7 \times 0) + (1 \times (-1)) + (9 \times 5) + (3 \times (-1)) + (8 \times 0) + (5 \times (-1)) + (1 \times 0)$	34
16	$(2 \times 0) + (7 \times (-1)) + (4 \times 0) + (9 \times (-1)) + (3 \times 5) + (1 \times (-1)) + (5 \times 0) + (1 \times (-1)) + (3 \times 0)$	-3

Lampiran 2 Hasil Perhitungan pada *Pooling Layer*

Posisi Kernel Ke -	Perhitungan pada <i>Pooling Layer</i>	Output
1	$f(1,1) = \max (23, 30, 0, 0)$	30
2	$f(1,2) = \max (0, 0, 28, 4)$	28
3	$f(2,1) = \max (0, 30, 2, 0)$	30
4	$f(2,2) = \max (0, 21, 34, 0)$	34



Lampiran 3 Proses Pelabelan gambar



Lampiran 4 Script Konversi Dataset XML ke CSV

Nama File : *xml_to_csv.py*

```
import os
import glob
import pandas as pd
import xml.etree.ElementTree as ET
def xml_to_csv(path):
    xml_list = []
    for xml_file in glob.glob(path + '/*.xml'):
        tree = ET.parse(xml_file)
        root = tree.getroot()
        for member in root.findall('object'):
            value = (root.find('filename').text,
                    int(root.find('size')[0].text),
                    int(root.find('size')[1].text),
                    member[0].text,
                    int(member[4][0].text),
                    int(member[4][1].text),
                    int(member[4][2].text),
                    int(member[4][3].text)
                    )
            xml_list.append(value)
    column_name = ['filename', 'width', 'height', 'class', 'xmin',
'ymax', 'xmax', 'ymax']
    xml_df = pd.DataFrame(xml_list, columns=column_name)
    return xml_df
def main():
    for directory in ['train', 'test']:
        image_path = os.path.join(os.getcwd(),
'annotations/{}'.format(directory))
        xml_df = xml_to_csv(image_path)
        xml_df.to_csv('data/{}_labels.csv'.format(directory),
index=None)
        print('Successfully converted xml to csv.')
main()
```

Lampiran 5 Script Konversi Dataset CSV ke TFRecord

Nama File : *generate_tfrecord.py*

```

from __future__ import division
from __future__ import print_function
from __future__ import absolute_import
import os
import io
import pandas as pd
import tensorflow as tf
import sys
sys.path.append("C:\\\\tensorflow\\\\models\\\\research\\\\")
sys.path.append("C:\\\\tensorflow\\\\models\\\\research\\\\object_detection\\\\utils")
sys.path.append("C:\\\\tensorflow\\\\models\\\\research\\\\slim")
sys.path.append("C:\\\\tensorflow\\\\models\\\\research\\\\slim\\\\nets")
from PIL import Image
from object_detection.utils import dataset_util
from collections import namedtuple, OrderedDict
flags = tf.app.flags
flags.DEFINE_string('type', '', 'Type of CSV input (train/test)')
flags.DEFINE_string('csv_input', '', 'Path to the CSV input')
flags.DEFINE_string('output_path', '', 'Path to output TFRecord')
FLAGS = flags.FLAGS
def class_text_to_int(row_label):
    if row_label == 'Alat Pemadam Api':
        return 1
    if row_label == 'Jalur Evakuasi':
        return 2
    else:
        return 0
def split(df, group):
    data = namedtuple('data', ['filename', 'object'])
    gb = df.groupby(group)
    return [data(filename, gb.get_group(x)) for filename, x in zip(gb.groups.keys(), gb.groups)]
def create_tf_example(group, path):
    with tf.gfile.GFile(os.path.join(path,
'{}'.format(group.filename)), 'rb') as fid:
        encoded_jpg = fid.read()
        encoded_jpg_io = io.BytesIO(encoded_jpg)
        image = Image.open(encoded_jpg_io)
        width, height = image.size
        filename = group.filename.encode('utf8')
        image_format = b'jpg'
        xmins = []
        xmaxs = []
        ymins = []
        ymaxs = []
        classes_text = []
        classes = []
        for index, row in group.object.iterrows():

```

```

        xmins.append(row['xmin'] / width)
        xmaxs.append(row['xmax'] / width)
        ymins.append(row['ymin'] / height)
        ymaxs.append(row['ymax'] / height)
        classes_text.append(row['class'].encode('utf8'))
        classes.append(class_text_to_int(row['class']))
    tf_example =
tf.train.Example(features=tf.train.Features(feature={
    'image/height': dataset_util.int64_feature(height),
    'image/width': dataset_util.int64_feature(width),
    'image/filename': dataset_util.bytes_feature(filename),
    'image/source_id': dataset_util.bytes_feature(filename),
    'image/encoded': dataset_util.bytes_feature(encoded_jpg),
    'image/format': dataset_util.bytes_feature(image_format),
    'image/object/bbox/xmin':
dataset_util.float_list_feature(xmins),
    'image/object/bbox/xmax':
dataset_util.float_list_feature(xmaxs),
    'image/object/bbox/ymin':
dataset_util.float_list_feature(ymins),
    'image/object/bbox/ymax':
dataset_util.float_list_feature(ymaxs),
    'image/object/class/text':
dataset_util.bytes_list_feature(classes_text),
    'image/object/class/label':
dataset_util.int64_list_feature(classes),
}))
    return tf_example
def main():
    writer = tf.python_io.TFRecordWriter(FLAGS.output_path)
    path = os.path.join(os.getcwd(),
'images/{}'.format(FLAGS.type))
    examples = pd.read_csv(FLAGS.csv_input)
    grouped = split(examples, 'filename')
    for group in grouped:
        tf_example = create_tf_example(group, path)
        writer.write(tf_example.SerializeToString())
    writer.close()
    output_path = os.path.join(os.getcwd(), FLAGS.output_path)
    print('Successfully created the TFRecords:
{}'.format(output_path))
if __name__ == '__main__':
    tf.app.run()print('Successfully converted xml to csv.')
main()

```

Lampiran 6 Script Konfigurasi Label Map

Nama File : *rambu_number_map.pbtxt*

```
item {  
  id: 1  
  name: 'Alat Pemadam Api'  
}  
item {  
  id: 2  
  name: 'Jalur Evakuasi'  
}
```



Lampiran 7 Script Konfigurasi Pipeline

Nama File : *rambu_v1.config*

```
model {
  ssd {
    num_classes: 2
    box_coder {
      faster_rcnn_box_coder {
        y_scale: 10.0
        x_scale: 10.0
        height_scale: 5.0
        width_scale: 5.0
      }
    }
    matcher {
      argmax_matcher {
        matched_threshold: 0.5
        unmatched_threshold: 0.5
        ignore_thresholds: false
        negatives_lower_than_unmatched: true
        force_match_for_each_row: true
      }
    }
    similarity_calculator {
      iou_similarity {
      }
    }
    anchor_generator {
      ssd_anchor_generator {
        num_layers: 6
        min_scale: 0.2
        max_scale: 0.95
        aspect_ratios: 1.0
        aspect_ratios: 2.0
        aspect_ratios: 0.5
        aspect_ratios: 3.0
        aspect_ratios: 0.3333
      }
    }
    image_resizer {
      fixed_shape_resizer {
        height: 300
        width: 300
      }
    }
    box_predictor {
      convolutional_box_predictor {
        min_depth: 0
        max_depth: 0
        num_layers_before_predictor: 0
        use_dropout: false
        dropout_keep_probability: 0.8
        kernel_size: 1
        box_code_size: 4
      }
    }
  }
}
```

```

    apply_sigmoid_to_scores: false
    conv_hyperparams {
      activation: RELU_6,
      regularizer {
        l2_regularizer {
          weight: 0.00004
        }
      }
      initializer {
        truncated_normal_initializer {
          stddev: 0.03
          mean: 0.0
        }
      }
      batch_norm {
        train: true,
        scale: true,
        center: true,
        decay: 0.9997,
        epsilon: 0.001,
      }
    }
  }
}

feature_extractor {
  type: 'ssd_mobilenet_v1'
  min_depth: 16
  depth_multiplier: 1.0
  conv_hyperparams {
    activation: RELU_6,
    regularizer {
      l2_regularizer {
        weight: 0.00004
      }
    }
    initializer {
      truncated_normal_initializer {
        stddev: 0.03
        mean: 0.0
      }
    }
  }
  batch_norm {
    train: true,
    scale: true,
    center: true,
    decay: 0.9997,
    epsilon: 0.001,
  }
}

loss {
  classification_loss {
    weighted_sigmoid {

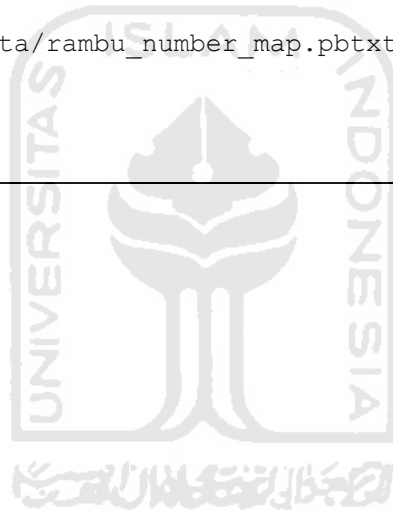
```

```

        anchorwise_output: true
    }
}
localization_loss {
    weighted_smooth_l1 {
        anchorwise_output: true
    }
}
}
hard_example_miner {
    num_hard_examples: 3000
    iou_threshold: 0.99
    loss_type: CLASSIFICATION
    max_negatives_per_positive: 3
    min_negatives_per_image: 0
}
classification_weight: 1.0
localization_weight: 1.0
}
normalize_loss_by_num_matches: true
post_processing {
    batch_non_max_suppression {
        score_threshold: 1e-8
        iou_threshold: 0.6
        max_detections_per_class: 100
        max_total_detections: 100
    }
    score_converter: SIGMOID
}
}
}
train_config: {
    batch_size: 2
    optimizer {
        rms_prop_optimizer: {
            learning_rate: {
                exponential_decay_learning_rate {
                    initial_learning_rate: 0.004
                    decay_steps: 800720
                    decay_factor: 0.95
                }
            }
        }
        momentum_optimizer_value: 0.9
        decay: 0.9
        epsilon: 1.0
    }
}
fine_tune_checkpoint:
"ssd_mobilenet_v1_coco_2017_11_17/model.ckpt"
from_detection_checkpoint: true
num_steps: 150000
data_augmentation_options {
    random_horizontal_flip {
    }
}
}

```

```
data_augmentation_options {  
  ssd_random_crop {  
  }  
}  
}  
train_input_reader: {  
  tf_record_input_reader {  
    input_path: "data/train.record"  
  }  
  label_map_path: "data/rambu_number_map.pbtxt"  
}  
eval_config: {  
  num_examples: 30  
  max_evals: 10  
}  
eval_input_reader: {  
  tf_record_input_reader {  
    input_path: "data/test.record"  
  }  
  label_map_path: "data/rambu_number_map.pbtxt"  
  shuffle: false  
  num_readers: 1  
  num_epochs: 1  
}
```



Lampiran 8 Script Training

Nama File : *train.py*

```
import functools
import json
import os
import tensorflow as tf

import sys
sys.path.append("C:\\\\tensorflow\\models\\research\\")
sys.path.append("C:\\\\tensorflow\\models\\research\\object_detection\\utils")
sys.path.append("C:\\\\tensorflow\\models\\research\\slim")
sys.path.append("C:\\\\tensorflow\\models\\research\\slim\\nets")

from tensorflow.contrib import framework as contrib_framework

from object_detection.builders import dataset_builder
from object_detection.builders import graph_rewriter_builder
from object_detection.builders import model_builder
from object_detection.legacy import trainer
from object_detection.utils import config_util

tf.logging.set_verbosity(tf.logging.INFO)

flags = tf.app.flags
flags.DEFINE_string('master', '', 'Name of the TensorFlow master to use.')
flags.DEFINE_integer('task', 0, 'task id')
flags.DEFINE_integer('num_clones', 1, 'Number of clones to deploy per worker.')
flags.DEFINE_boolean('clone_on_cpu', False,
                    'Force clones to be deployed on CPU. Note that even if '
                    'set to False (allowing ops to run on gpu), some ops may '
                    'still be run on the CPU if they have no GPU kernel.')
flags.DEFINE_integer('worker_replicas', 1, 'Number of worker+trainer '
                    'replicas.')
flags.DEFINE_integer('ps_tasks', 0,
                    'Number of parameter server tasks. If None, does not use '
                    'a parameter server.')
flags.DEFINE_string('train_dir', '',
                    'Directory to save the checkpoints and training summaries.')

flags.DEFINE_string('pipeline_config_path', '',
                    'Path to a pipeline_pb2.TrainEvalPipelineConfig config '
                    'file. If provided, other configs are ignored')
```

```

flags.DEFINE_string('train_config_path', '',
                    'Path to a train_pb2.TrainConfig config
file.')
flags.DEFINE_string('input_config_path', '',
                    'Path to an input_reader_pb2.InputReader
config file.')
flags.DEFINE_string('model_config_path', '',
                    'Path to a model_pb2.DetectionModel config
file.')

FLAGS = flags.FLAGS

@contrib_framework.deprecated(None, 'Use
object_detection/model_main.py.')
def main(_):
    assert FLAGS.train_dir, '`train_dir` is missing.'
    if FLAGS.task == 0: tf.gfile.MakeDirs(FLAGS.train_dir)
    if FLAGS.pipeline_config_path:
        configs = config_util.get_configs_from_pipeline_file(
            FLAGS.pipeline_config_path)
        if FLAGS.task == 0:
            tf.gfile.Copy(FLAGS.pipeline_config_path,
                           os.path.join(FLAGS.train_dir,
                                           'pipeline.config'),
                           overwrite=True)
    else:
        configs = config_util.get_configs_from_multiple_files(
            model_config_path=FLAGS.model_config_path,
            train_config_path=FLAGS.train_config_path,
            train_input_config_path=FLAGS.input_config_path)
        if FLAGS.task == 0:
            for name, config in [ ('model.config',
                                  FLAGS.model_config_path),
                                  ('train.config',
                                  FLAGS.train_config_path),
                                  ('input.config',
                                  FLAGS.input_config_path) ]:
                tf.gfile.Copy(config, os.path.join(FLAGS.train_dir, name),
                               overwrite=True)

        model_config = configs['model']
        train_config = configs['train_config']
        input_config = configs['train_input_config']

        model_fn = functools.partial(
            model_builder.build,
            model_config=model_config,
            is_training=True)

    def get_next(config):
        return dataset_builder.make_initializable_iterator(
            dataset_builder.build(config)).get_next()

```

```

create_input_dict_fn = functools.partial(get_next, input_config)

env = json.loads(os.environ.get('TF_CONFIG', '{}'))
cluster_data = env.get('cluster', None)
cluster = tf.train.ClusterSpec(cluster_data) if cluster_data
else None
task_data = env.get('task', None) or {'type': 'master', 'index':
0}
task_info = type('TaskSpec', (object,), task_data)

# Parameters for a single worker.
ps_tasks = 0
worker_replicas = 1
worker_job_name = 'lonely_worker'
task = 0
is_chief = True
master = ''

if cluster_data and 'worker' in cluster_data:
    # Number of total worker replicas include "worker"s and the
"master".
    worker_replicas = len(cluster_data['worker']) + 1
    if cluster_data and 'ps' in cluster_data:
        ps_tasks = len(cluster_data['ps'])

    if worker_replicas > 1 and ps_tasks < 1:
        raise ValueError('At least 1 ps task is needed for distributed
training.')

    if worker_replicas >= 1 and ps_tasks > 0:
        # Set up distributed training.
        server = tf.train.Server(tf.train.ClusterSpec(cluster),
protocol='grpc',
                                job_name=task_info.type,
                                task_index=task_info.index)

        if task_info.type == 'ps':
            server.join()
            return

        worker_job_name = '%s/task:%d' % (task_info.type,
task_info.index)
        task = task_info.index
        is_chief = (task_info.type == 'master')
        master = server.target

graph_rewriter_fn = None
if 'graph_rewriter_config' in configs:
    graph_rewriter_fn = graph_rewriter_builder.build(
        configs['graph_rewriter_config'], is_training=True)

trainer.train(
    create_input_dict_fn,
    model_fn,

```

```
train_config,  
master,  
task,  
FLAGS.num_clones,  
worker_replicas,  
FLAGS.clone_on_cpu,  
ps_tasks,  
worker_job_name,  
is_chief,  
FLAGS.train_dir,  
graph_hook_fn=graph_rewriter_fn)  
  
if __name__ == '__main__':  
    tf.app.run()
```



Lampiran 9 Script Export Graph Model

Nama File : *export_inference_graph.py*

```
import tensorflow as tf

import sys
sys.path.append("C:\\tensorflow\\models\\research\\")
sys.path.append("C:\\tensorflow\\models\\research\\object_detection\\utils")
sys.path.append("C:\\tensorflow\\models\\research\\slim")
sys.path.append("C:\\tensorflow\\models\\research\\slim\\nets")

from google.protobuf import text_format
from object_detection import exporter
from object_detection.protos import pipeline_pb2

slim = tf.contrib.slim
flags = tf.app.flags

flags.DEFINE_string('input_type', 'image_tensor', 'Type of input
node. Can be '
                    'one of [\'image_tensor\',
\'encoded_image_string_tensor\',
\'tf_example\']')
flags.DEFINE_string('input_shape', None,
                    'If input_type is \'image_tensor\', this can
explicitly set '
                    'the shape of this input tensor to a fixed
size. The '
                    'dimensions are to be provided as a comma-
separated list '
                    'of integers. A value of -1 can be used for
unknown '
                    'dimensions. If not specified, for an
\'image_tensor, the '
                    'default shape will be partially specified as
',
                    '[None, None, None, 3]'.')
flags.DEFINE_string('pipeline_config_path', None,
                    'Path to a
pipeline_pb2.TrainEvalPipelineConfig config '
                    'file.')
flags.DEFINE_string('trained_checkpoint_prefix', None,
                    'Path to trained checkpoint, typically of the
form '
                    'path/to/model.ckpt')
flags.DEFINE_string('output_directory', None, 'Path to write
outputs.')
flags.DEFINE_string('config_override', '',
                    'pipeline_pb2.TrainEvalPipelineConfig '
                    'text proto to override
pipeline_config_path.')
flags.DEFINE_boolean('write_inference_graph', False,
                    'If true, writes inference graph to disk.')
```

```

tf.app.flags.mark_flag_as_required('pipeline_config_path')
tf.app.flags.mark_flag_as_required('trained_checkpoint_prefix')
tf.app.flags.mark_flag_as_required('output_directory')
FLAGS = flags.FLAGS

def main(_):
    pipeline_config = pipeline_pb2.TrainEvalPipelineConfig()
    with tf.gfile.GFile(FLAGS.pipeline_config_path, 'r') as f:
        text_format.Merge(f.read(), pipeline_config)
    text_format.Merge(FLAGS.config_override, pipeline_config)
    if FLAGS.input_shape:
        input_shape = [
            int(dim) if dim != '-1' else None
            for dim in FLAGS.input_shape.split(',')
        ]
    else:
        input_shape = None
    exporter.export_inference_graph(
        FLAGS.input_type, pipeline_config,
        FLAGS.trained_checkpoint_prefix,
        FLAGS.output_directory, input_shape=input_shape,
        write_inference_graph=FLAGS.write_inference_graph)

if __name__ == '__main__':
    tf.app.run()

```

Lampiran 10 Script untuk mendeteksi rambu K3 pada video

Nama File : rambu_video.py

```
import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
import tensorflow as tf
import zipfile
# In[2]:
from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
from PIL import Image
# In[16]:
import cv2
cap =
cv2.VideoCapture(r'C:/tensorflow/models/research/object_detection/
bismillah5.mp4')
# In[5]:
# This is needed since the notebook is stored in the
object_detection folder.
sys.path.append("..")
# In[7]:
from utils import label_map_util
from utils import visualization_utils as vis_util
# In[8]:
# What model to download.
MODEL_NAME = 'rambu_baru'
#MODEL_NAME = 'ssd_mobilenet_v1_coco_11_06_2017'
MODEL_FILE = MODEL_NAME + '.tar.gz'
DOWNLOAD_BASE =
'http://download.tensorflow.org/models/object_detection/'
# Path to frozen detection graph. This is the actual model that is
used for the object detection.
PATH_TO_CKPT = MODEL_NAME + '/frozen_inference_graph.pb'
# List of the strings that is used to add correct label for each
box.
PATH_TO_LABELS = os.path.join('data', 'rambu_number_map.pbtxt')
NUM_CLASSES = 4
# In[7]:
# In[9]:
detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')
# In[10]:
label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
```

```

categories =
label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES, use_display_name=True)
category_index = label_map_util.create_category_index(categories)
# In[11]:
def load_image_into_numpy_array(image):
    (im_width, im_height) = image.size
    return np.array(image.getdata()).reshape(
        (im_height, im_width, 3)).astype(np.uint8)
# In[12]:
# If you want to test the code with your images, just add path to
the images to the TEST_IMAGE_PATHS.
PATH_TO_TEST_IMAGES_DIR = 'test_images'
TEST_IMAGE_PATHS = [ os.path.join(PATH_TO_TEST_IMAGES_DIR,
'image{}.jpg'.format(i)) for i in range(1, 3) ]
# Size, in inches, of the output images.
IMAGE_SIZE = (12, 8)
# In[ ]:
with detection_graph.as_default():
    with tf.Session(graph=detection_graph) as sess:
        while True:
            ret, image_np = cap.read()
            # Expand dimensions since the model expects images to have
            shape: [1, None, None, 3]
            image_np_expanded = np.expand_dims(image_np, axis=0)
            image_tensor =
detection_graph.get_tensor_by_name('image_tensor:0')
            # Each box represents a part of the image where a particular
            object was detected.
            boxes =
detection_graph.get_tensor_by_name('detection_boxes:0')
            # Each score represent how level of confidence for each of
            the objects.
            # Score is shown on the result image, together with the
            class label.
            scores =
detection_graph.get_tensor_by_name('detection_scores:0')
            classes =
detection_graph.get_tensor_by_name('detection_classes:0')
            num_detections =
detection_graph.get_tensor_by_name('num_detections:0')
            # Actual detection.
            (boxes, scores, classes, num_detections) = sess.run(
                [boxes, scores, classes, num_detections],
                feed_dict={image_tensor: image_np_expanded})
            # Visualization of the results of a detection.
            vis_util.visualize_boxes_and_labels_on_image_array(
                image_np,
                np.squeeze(boxes),
                np.squeeze(classes).astype(np.int32),
                np.squeeze(scores),
                category_index,
                use_normalized_coordinates=True,
                line_thickness=8)

```

```
cv2.imshow('object detection', cv2.resize(image_np,  
(1000,800)))  
if cv2.waitKey(25) & 0xFF == ord('q'):  
    cv2.destroyAllWindows()  
    break
```

