

**ESTIMASI KECEPATAN KENDARAAN MELALUI  
VIDEO PENGAWAS LALU LINTAS  
MENGGUNAKAN PARALLEL LINE MODEL**

Skripsi

Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana



**PROGRAM STUDI TEKNIK INFOMATIKA**

**FAKULTAS SAINS DAN TEKNOLOGI**

**UNIVERSITAS ISLAM NEGERI SYARIF HIDAYATULLAH**

**JAKARTA**

**2020**

## LEMBAR PERSETUJUAN

### LEMBAR PERSETUJUAN

#### PREDIKSI KECEPATAN KENDARAAN MELALUI VIDEO PENGAWAS LALU LINTAS MENGGUNAKAN PARALLEL LINE MODEL

Skripsi

Sebagai salah satu syarat untuk memperoleh

Gelar Sarjana Komputer (S.Kom)

Oleh :

ALFI SALIM

11160910000014

Jakarta, 12 Januari 2020

Menyetujui,

Pembimbing I

Pembimbing II

Fenty Eka Muzayyana Agustin, M.Kom  
NIP.197608052009122003

Dr. Imam Marzuki Shofi, M.T.  
NIP.197202052008011010

Mengetahui,

Ketua Prodi Studi Teknik Informatika

Dr. Imam Marzuki Shofi, M.T.  
NIP.197202052008011010

## PENGESAHAN UJIAN

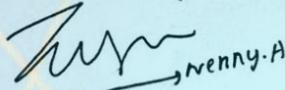
Skripsi berjudul "Prediksi Kecepatan Kendaraan Melalui Video Pengawas Lalu Lintas Menggunakan *Parallel Line Model*" yang ditulis oleh Alfi Salim dengan NIM 11160910000014, telah diuji dan dinyatakan lulus dalam sidang *munaqosyah* Fakultas Sains dan Teknologi, UIN Syarif Hidayatullah Jakarta pada hari Senin, 27 Januari 2020. Skripsi ini telah diterima sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer (S.Kom) pada Program Studi Teknik Informatika.

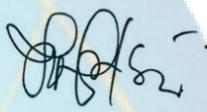
Jakarta, Januari 2020

Tim Penguji,

Penguji I

Penguji II

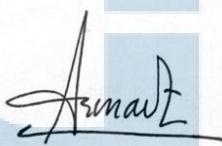
  
Nenny Anggraini, S.Kom, MT

  
Anif Hanifa Setianingrum M.Si

Tim Pembimbing,

Pembimbing I

Pembimbing II

  
Fenty Eka Muzayyana Agustin, M.Kom  
NIP.197608052009122003

  
Dr. Imam Marzuki Shofi, M.T  
NIP.197202052008011010

Mengetahui,



Prof. Dr. Lily Surayya Eka Putri, M.Env.Stud  
NIP.196904042005012005

Ketua Program Studi Teknik  
Informatika

  
Dr. Imam Marzuki Shofi, M.T  
NIP.197202052008011010

## **PERNYATAAN ORISINILITAS**

Dengan ini saya menyatakan bahwa :

1. Skripsi ini merupakan hasil karya asli saya yang diajukan untuk memenuhi salah satu persyaratan memperoleh gelar Strata 1 di UIN Syarif Hidayatullah Jakarta.
2. Semua sumber yang saya gunakan dalam penulisan ini telah saya cantumkan sesuai dengan ketentuan yang berlaku di UIN Syarif Hidayatullah Jakarta.
3. Apabila dikemudian hari terbukti karya ini bukan karya asli saya atau merupakan hasil jiplakan karya orang lain, maka saya bersedia menerima sanksi yang berlaku di UIN Syarif Hidayatullah Jakarta

Jakarta, Januari 2020



## **PERNYATAAN PERSETUJUAN PUBLIKASI SKRIPSI**

Sebagai civitas akademik UIN Syarif Hidayatullah Jakarta, saya yang bertanda tangan dibawah ini :

Nama : Alfi Salim

NIM : 11160910000014

Program Studi : Teknik Informatika

Fakultas : Sains dan Teknologi

Jenis Karya : Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Islam Negeri Syarif Hidayatullah Jakarta Hak Bebas Royaliti Noneksklusif (*Non-exclusive Royalty Free Right*) atas karya ilmiah yang berjudul :

### **PREDIKSI KECEPATAN KENDARAAN MELALUI VIDEO PENGAWAS**

#### **LALU LINTAS MENGGUNAKAN PARALLEL LINE MODEL**

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royaliti Noneksklusif ini Universitas Islam Negeri Syarif Hidayatullah Jakarta berhak menyimpan, mengalih media/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat dan mempublikasikan tugas akhir saya selama teteap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak.

Demikian pernyataan ini saya buat dengan sebenarnya.

Jakarta, Januari 2020



Alfi Salim

11160910000014

**Nama : Alfi Salim**

**Program Studi : Teknik Informatika**

**Judul : ESTIMASI KECEPATAN KENDARAAN MELALUI  
VIDEO PENGAWAS LALU LINTAS MENGGUNAKAN  
*PARALLEL LINE MODEL***

## **ABSTRAK**

Estimasi merupakan suatu pengukuran yang didasarkan pada hasil kuantitatif. Estimasi dapat dilakukan dengan memanfaatkan teknologi *computer vision*. Teknologi *computer vision* mengalami perkembangan yang pesat, termasuk juga dalam membangun *intelligent transportation system (ITS)*. Penelitian ini berfokuskan pada estimasi kecepatan kendaraan melalui video pengawas lalu lintas. *dataset* menggunakan video berdurasi 2 menit yang dipecah menjadi 2.446 *frame*. Terdapat 3 tahapan utama dalam perancangan sistem, Pertama, data diolah untuk menjadi bahan belajar dari algoritma *deep learning* dengan model *Faster R-CNN* untuk memyelesaikan masalah deteksi objek. Selanjutnya, objek yang berhasil terdeteksi ditracking menggunakan algoritma *centroid tracking*. Terakhir, dilakukan perhitungan estimasi kecepatan pada objek tersebut menggunakan *parallel line model*. Input sistem berupa video pengawas lalu lintas di lokasi yang sama pada saat perancangan sistem. Video tersebut akan dipecah tiap framenya untuk diproses ke tahap deteksi, *tracking* dan estimasi kecepatan. Setelah semua proses selesai, tiap *frame* tersebut disatukan kembali menjadi sebuah video baru dengan *bounding box* disekitar objek dan menampilkan estimasi kecepatannya. Penelitian ini berhasil mencapai *score RMSE* 7,116, *score NRMSE* 0,103 dan *score evaluasi metric* 0,897.

**Kata Kunci : Deep Learning, Faster R-CNN, Centroid Tracking,  
Parallel Line Model, Evaluasi Metric.**

**Jumlah Pustaka : 9 Buku + 2 Skripsi + 5 Jurnal + 10 Website**

**Jumlah Halaman : 100 Halaman**

**Nama : Alfi Salim**

**Program Studi : Teknik Informatika**

**Judul : ESTIMASI KECEPATAN KENDARAAN MELALUI  
VIDEO PENGAWAS LALU LINTAS MENGGUNAKAN  
PARALLEL LINE MODEL**

### **ABSTRACT**

*Estimation is a measurement based on quantitative results. Estimation can be made by utilizing computer vision technology. Computer vision technology is evolving, including building intelligent transportation systems (ITS). This research focuses on estimating vehicle speed through traffic control videos. The dataset uses a 2 minute video which is broken up into 2,446 frames. There are 3 main stages in the system design, First, the data is processed to become learning material from the deep learning algorithm with the Faster R-CNN model to solve the object detection problem. Furthermore, the object is successfully traced using a centroid tracking algorithm. Finally, the speed of the object is calculated using the parallel line model. The input system consists of a video traffic controller in the same location at the time of system design. The video will be broken down each frame to be processed into speed detection, tracking and estimation. After all processes are completed, each frame is put back together into a new video with a bounding box containing the object and displaying its estimated speed. This study succeeded in achieving RMSE score of 7.116, NRMSE score of 0.103 and metric evaluation score of 0.897.*

**Key Word : Deep Learning, Faster R-CNN, Centroid Tracking,  
Parallel Line Model, Metric Evaluation.**

**Resource : 9 Books + 2 Thesis + 5 Journals + 10 Websites**

**Number Of Page : 100 Pages**

## KATA PENGANTAR

Puji syukur kehadirat Allah SWT yang telah memberikan penulis berbagai macam nikmat, rahmat, taufik dan hidayah-Nya sehingga penulis dapat menyelesaikan skripsi ini dengan sebaik-baiknya. Shalawat serta salam penulis ucapan kepada Nabi Muhammad SAW beserta keluarganya, sahabatnya keturunannya serta para pengikutnya hingga akhir zaman.

Skripsi berjudul “Estimasi Kecepatan Kendaraan Melalui Video Pengawas Lalu Lintas Menggunakan *Parallel Line Model*” penulis susun sebagai syarat untuk mendapatkan gelar Sarjana Komputer (S.Kom) pada Program Studi Teknik Informatika UIN Syarif Hidayatullah Jakarta.

Penulis mengucapkan rasa terimakasih karena dalam penyusunan laporan ini, penulis mendapatkan bantuan dari berbagai pihak, diantaranya :

1. Ibu Prof. Dr. Lily Surayya Eka Putri, M.Env.Stud selaku Dekan Fakultas Sains dan Teknologi UIN Syarif Hidayatullah Jakarta.
2. Bapak Dr. Imam Marzuki Shofi, M.T selaku Ketua Program Studi Teknik Informatika UIN Syarif Hidayatullah Jakarta sekaligus sebagai Dosen Pembimbing II yang telah meluangkan waktunya untuk memberikan banyak saran, arahan, motivasi kepada penulis, sehingga penulis dapat menyelesaikan skripsi ini dengan baik.
3. Ibu Fenty Eka Muzayyana Agustin, M.Kom sebagai Dosen Pembimbing I yang juga telah meluangkan waktunya untuk memberikan banyak saran, arahan, motivasi kepada penulis, sehingga penulis dapat menyelesaikan skripsi ini dengan baik.
4. Seluruh Dosen dan Staff Karyawan Fakultas Sains dan Teknologi UIN Syarif Hidayatullah Jakarta yang telah memberikan banyak ilmu dan dukungan selama masa perkuliahan.
5. Kelurga penulis, khususnya kedua orang tua penulis yang telah memberikan dukungan baik dalam bentuk lahir maupun batin, sehingga setiap langkah

penulis dalam melaksanakan kegiatan perkuliahan selalu diiringi oleh doanya ilmu yang didapat menjadi manfaat dimasa depan.

6. Bapak M. Octaviano Pratama, S.Kom., M.Kom selaku Head of Artificial Intelligence, yang telah meluangkan waktunya untuk memberikan saran, arahan dan masukkan yang bermanfaat bagi penulis dalam menyelesaikan skripsi ini.
7. Seluruh tim dari Bisa.AI khususnya untuk Adi, Rusnandi Fikri, Rusnanda Farhan yang telah meluangkan waktunya juga untuk memberikan arahan dan masukkan kepada penulis.
8. Teman-teman Teknik Informatika, terutama kepada para sobat-sobat yang telah memberikan warna dan keceriaan selama masa perkuliahan.
9. Teman-teman KKN SERSAN 2019 yang telah memberikan dukungan serta dorongan kepada penulis dalam menyelesaikan skripsi ini.
10. Kepada Alfiyyah Ramadhanty yang telah memberikan doa, dukungan dan motivasi kepada penulis dalam menyelesaikan skripsi ini.
11. Serta seluruh pihak yang telah membantu penulis yang tidak dapat penulis sebutkan namanya satu persatu.

Penulis menyadari bahwa dalam skripsi ini masih terdapat banyak kekurangan. oleh kerena itu, penulis sangat mengharapkan saran dan kritik dari pembaca. Sehingga penulis dapat memperbaiki dan menyempurnakan laporan ini. penulis berharap laporan ini dapat memberikan manfaat bagi yang membacanya.

Jakarta, 02 Januari 2020



Alfi Salim  
11160910000014

## DAFTAR ISI

<b>LEMBAR PERSETUJUAN .....</b>	i
<b>PENGESAHAN UJIAN.....</b>	ii
<b>PERNYATAAN ORISINILITAS .....</b>	iii
<b>PERNYATAAN PERSETUJUAN PUBLIKASI SKRIPSI.....</b>	iv
<b>ABSTRAK .....</b>	v
<b>ABSTRACT .....</b>	vi
<b>KATA PENGANTAR .....</b>	vii
<b>DAFTAR ISI.....</b>	ix
<b>DAFTAR GAMBAR.....</b>	xii
<b>DAFTAR TABEL .....</b>	xiv
<b>BAB I PENDAHULUAN.....</b>	1
1.1    Latar Belakang .....	1
1.2    Tujuan Penelitian.....	4
1.3    Manfaat Penelitian.....	5
1.3.1    Bagi Mahasiswa .....	5
1.3.2    Bagi Universitas .....	5
1.3.3    Bagi User.....	5
1.4    Pertanyaan Penelitian .....	5
1.5    Rumusan Masalah .....	5
1.6    Batasan Masalah.....	6
1.6.1    Proses .....	6
1.6.2    Metode .....	6
1.6.3    Tools .....	7
1.7    Metodologi Penelitian .....	7
1.7.1    Metode Pengumpulan Data.....	7
1.7.2    Metode Pengembangan Sistem .....	7
1.8    Sistematika Penulisan.....	8
<b>BAB II LANDASAN TEORI .....</b>	9
2.1    Video .....	9

2.2	Deteksi Objek .....	9
2.3	Pelacakan Objek .....	12
2.4	Kecerdasan Buatan .....	14
2.5	<i>Datasets</i> .....	14
2.6	<i>Machine Learning</i> .....	15
2.7	<i>Deep Learning</i> .....	18
2.8	<i>Artificial Neural Network</i> .....	19
2.7.1	<i>Multi-layer Network</i> .....	22
2.7.2	<i>Backpropagation</i> .....	23
2.7.3	<i>Activation Function</i> .....	25
2.9	<i>Dropout Regularization</i> .....	26
2.10	<i>Convolutional Neural Network</i> .....	27
2.9.1	<i>Convolution Layer</i> .....	27
2.9.2	<i>Stride</i> .....	28
2.9.3	<i>Padding</i> .....	29
2.9.4	<i>Pooling Layer</i> .....	29
2.9.5	<i>Activation Function CNN</i> .....	30
2.9.6	<i>Arsitektur Convolutional Neural Network</i> .....	30
2.11	<i>Faster R-CNN</i> .....	33
2.12	<i>Tensorflow</i> .....	33
2.13	<i>Centroid Tracking</i> .....	34
2.14	<i>Parallel Line Model</i> .....	35
2.15	<i>Normalized Root Mean Square Error</i> .....	36
2.16	Penelitian Terdahulu.....	37
	<b>BAB III METODE PENELITIAN .....</b>	<b>41</b>
3.1	Metode Pengumpulan Data .....	41
3.1.1	Studi Pustaka.....	41
3.1.2	Observasi.....	41
3.1.3	Wawancara.....	42
3.2	Metode Pengembangan Sistem .....	42
3.2.1	Analisis Kebutuhan.....	42
3.2.2	Desain <i>Prototyping</i> .....	43

3.2.3 <i>Development</i> .....	43
3.2.4 <i>Testing</i> .....	44
3.2.5 Evaluasi Sistem.....	44
3.3 Kerangka Berpikir .....	44
<b>BAB IV IMPLEMENTASI .....</b>	<b>46</b>
4.1 Analisis Kebutuhan .....	46
4.2 Desain <i>Prototyping</i> .....	49
4.3 <i>Development</i> .....	52
4.3.1 <i>Annotasi</i> .....	52
4.3.2 <i>Split Dataset</i> .....	55
4.3.3 <i>Training</i> .....	57
4.3.4 <i>Tracking</i> .....	67
4.3.5 <i>Speed Estimation</i> .....	70
4.4 <i>Testing</i> .....	73
4.5 Evaluasi Sistem .....	73
<b>BAB V HASIL DAN PEMBAHASAN .....</b>	<b>74</b>
5.1 <i>Dataset</i> .....	74
5.2 <i>Training</i> .....	75
5.3 <i>Testing</i> .....	80
5.3.1 <i>Object Detection</i> .....	80
5.3.2 <i>Object Tracking</i> .....	82
5.3.3 <i>Speed Estimation</i> .....	83
5.4 Hasil.....	84
<b>BAB VI PENUTUP .....</b>	<b>88</b>
6.1 Kesimpulan.....	88
6.2 Saran .....	88
<b>DAFTAR PUSTAKA .....</b>	<b>90</b>
<b>DAFTAR LAMPIRAN .....</b>	<b>89</b>

## DAFTAR GAMBAR

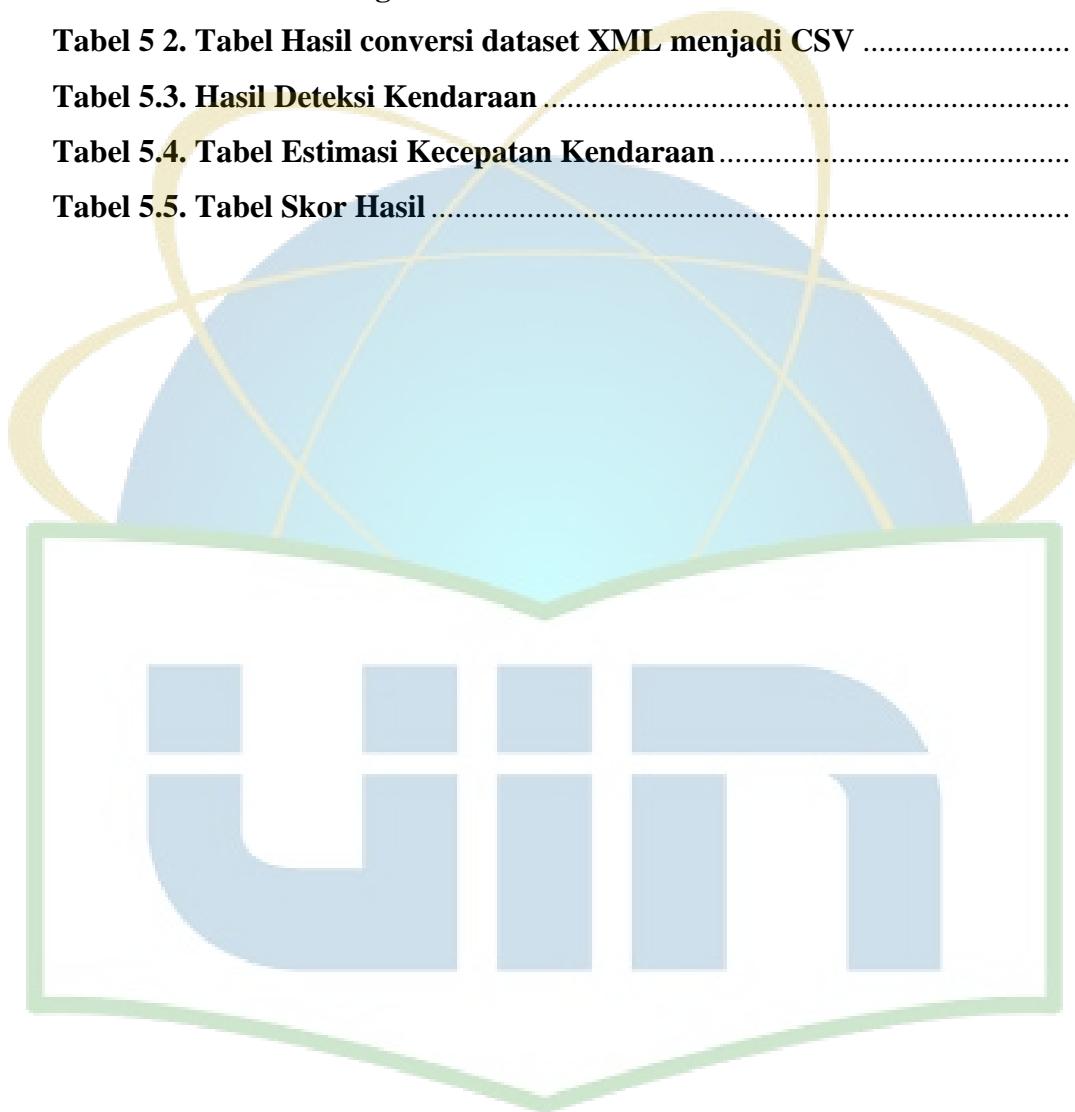
<b>Gambar 2.1. Contoh Deteksi Objek Pada Gambar .....</b>	12
<b>Gambar 2.2. Layer Machine Learning .....</b>	17
<b>Gambar 2.3. Konsep Kejra Machine Learning.....</b>	17
<b>Gambar 2.4. Layer Deep Learning.....</b>	18
<b>Gambar 2.5. Konsep Kejra Deep Learning .....</b>	19
<b>Gambar 2.6. Preception Sederhana Dengan Layer Input dan Output .....</b>	20
<b>Gambar 2.7. Artificial Neural Network Sederhana.....</b>	22
<b>Gambar 2.8. Artificial Neural Network Multi-layer .....</b>	23
<b>Gambar 2.9. Backpropagation .....</b>	24
<b>Gambar 2.10. Dropout .....</b>	26
<b>Gambar 2.11. Max Pooling.....</b>	29
<b>Gambar 2.12. Citra RGB .....</b>	31
<b>Gambar 2.13. Feature Map .....</b>	32
<b>Gambar 2.14. Ilustrasi model Parallel Line .....</b>	36
<b>Gambar 3 1. Kerangka Berpikir .....</b>	45
<b>Gambar 4.1. Contoh Gambar Dataset .....</b>	47
<b>Gambar 4.2. Panjang Landmark dan Jarak Antara Landmark .....</b>	47
<b>Gambar 4.3. Hasil Split Dataset Video .....</b>	48
<b>Gambar 4.4. Perbandingan Model Objek Deteksi.....</b>	49
<b>Gambar 4.5. Perbandingan Model Objek Deteksi.....</b>	50
<b>Gambar 4.6. Proseses Annotasi .....</b>	53
<b>Gambar 4.8. Hasil Proses Annotasi .....</b>	54
<b>Gambar 4.9. Data train.....</b>	56
<b>Gambar 4.10. Data test .....</b>	56
<b>Gambar 4.11. Ilustrasi Langkah Perhitungan Nilai Centroid .....</b>	68
<b>Gambar 4.12. Gambar Menghitung Jarak Centroid Terdekat Antar Frame .....</b>	68
<b>Gambar 4.13. Menentukan Centroid Terdekat .....</b>	69
<b>Gambar 4.14. Menentukan ID Pada Objek Yang Terdeteksi .....</b>	69

<b>Gambar 4.15. Lintasan Estimasi Kecepatan Kendaraan .....</b>	71
<b>Gambar 5.1. Proses Training .....</b>	76
<b>Gambar 5.2. Grafik Total Loss .....</b>	77
<b>Gambar 5.3. Grafik Clone Loss .....</b>	77
<b>Gambar 5.4. Grafik Box Classifier Loss/Classification .....</b>	77
<b>Gambar 5.5. Grafik Box Classifier Loss/Localization .....</b>	78
<b>Gambar 5.6. Grafik RPN Loss/Localization .....</b>	78
<b>Gambar 5.7. Grafik RPN Loss/Objectness .....</b>	78
<b>Gambar 5.8. Gambar Hasil Deteksi Kendaraan Pada Video .....</b>	81
<b>Gambar 5.9. Hasil Tracking Objek .....</b>	83
<b>Gambar 5.10. Hasil Estimasi Kecepatan .....</b>	84



## DAFTAR TABEL

<b>Tabel 2 1. Tabel Literatur Sejenis .....</b>	37
<b>Tabel 4.1. Annotasi Terhadap Objek Yang Sama .....</b>	54
<b>Tabel 5.1. Tabel Pembagian Dataset .....</b>	74
<b>Tabel 5 2. Tabel Hasil conversi dataset XML menjadi CSV .....</b>	75
<b>Tabel 5.3. Hasil Deteksi Kendaraan .....</b>	85
<b>Tabel 5.4. Tabel Estimasi Kecepatan Kendaraan .....</b>	86
<b>Tabel 5.5. Tabel Skor Hasil .....</b>	87



## **DAFTAR LAMPIRAN**

<b>Lampiran 1 Surat Pernyataan Data Validasi.....</b>	<b>93</b>
<b>Lampiran 2 Data Validasi .....</b>	<b>94</b>
<b>Lampiran 3 Surat Pernyataan Wawancara .....</b>	<b>97</b>
<b>Lampiran 4 Daftar Pertanyaan Wawancara.....</b>	<b>98</b>



## **BAB I**

### **PENDAHULUAN**

#### **1.1 Latar Belakang**

Pengertian estimasi adalah keseluruhan proses yang membutuhkan serta menggunakan estimator untuk menghasilkan sebuah estimate dari suatu parameter (Harinaldi : 2005). Sedangkan arti estimasi menurut Tockey, 2004 adalah suatu pengukuran yang didasarkan pada hasil kuantitatif atau dengan kata lain, tingkat akurasinya bisa diukur dengan angka. Sedangkan menurut KBBI estimasi adalah perkiraan, penilaian, atau pendapat. Ini menunjukkan bahwa istilah estimasi bisa kita gunakan secara umum untuk menyatakan perkiraan, penilaian, atau pendapat kita tentang sesuatu (Dani 2019). Contoh dari estimasi adalah mengestimasi waktu dan biaya dalam menyelesaikan sebuah proyek pembuatan jalan. Estimasi juga dapat dilakukan dengan memanfaatkan teknologi *artificial intelligence* dengan memanfaatkan data yang banyak. Contoh dari aplikasi *artificial intelligence* dalam hal estimasi adalah aplikasi untuk mengestimasi biaya dalam sebuah proyek, estimasi nilai saham dan sebaginya.

Penggunaan teknologi *artificial intelligence* terus mengalami peningkatan, khususnya dalam bidang *computer vision*. Tujuan *computer vision* adalah untuk membuat keputusan bermanfaat tentang objek fisika nyata berdasarkan apa yang didapat didalam citra tersebut. Cara kerja *computer vision* adalah mencoba untuk meniru cara kerja penglihatan manusia yang menangkap berbagai macam informasi terhadap apa yang dilihatnya. Intinya, *computer vision* bertujuan untuk membuat sistem bisa sepandai manusia. Banyak informasi yang dapat diperoleh dengan menggunakan teknik *computer vision* karena teknik *computer vision* membuat sebuah sistem seolah-olah dapat melihat objek disekitarnya untuk selanjutnya dianalisis sehingga mendapatkan informasi yang bermanfaat untuk diolah menjadi perintah tertentu.

Teknologi *computer vision* telah banyak dimanfaatkan dalam berbagai macam bidang seperti dalam bidang keamanan, dalam bidang social, dalam bidang kesehatan, dalam bidang transportasi dan sebagainya. Teknologi *smart city* dalam bidang transportasi menjadi salah satu domain yang paling menarik dari ilmu *computer vision*. Melalui sebuah video pengawas lalu lintas, dapat diperoleh sejumlah besar data yang dapat diolah menjadi suatu informasi yang dapat digunakan dalam manajemen lalu lintas, meningkatkan keselamanatan di lokasi lalu lintas, mendeteksi, memprediksi hingga mencegah kecelakaan lalu lintas serta berbagai macam aplikasi lainnya.

Dalam *intelligent transportation system (ITS)*, data lalu lintas menjadi sebuah kunci untuk melakukan penelitian dan merancang sebuah sistem. Kamera pengawas lalu lintas telah digunakan secara luas namun belum dimanfaatkan secara maksimal. Mengubah kamera pengawas lalu lintas dari pengawasan manual menjadi pengawasan otomatis merupakan hal yang menarik untuk dilakukan. Jadi, petugas tidak hanya menggunakan kamera sebagai pengawas lalu lintas namun juga bisa menggunakan kamera sebagai media untuk mengumpulkan data kecepatan kendaraan dan pemantauan arus lalu lintas. Banyak penelitian yang telah dikembangkan untuk analisis otomatis data video lalu lintas seperti deteksi dan pelacakan kendaraan otomatis, analisis kecepatan dan lain sebagainya. Semua itu telah dikembangkan dan berusaha ditingkatkan akurasinya secara terus menerus. Namun tantangan sesungguhnya terdapat pada pengembangan metode yang cepat, efisien dan akurat secara komputasi untuk digunakan dalam dunia nyata.

Berbagai metode telah dikembangkan sebelumnya menggunakan *computer vision* dan *machine learning*. (Hua, Kapoor, and Anastasiu n.d.) 2018, melakukan sebuah penelitian yang berjudul “*Vehicle Tracking and Speed Estimation from Traffic Videos*”, penelitian menggunakan metode pendekatan *detect-then-track* dengan model deteksi menggunakan model *3D Deformable* dengan *minimum confidence scores*  $\alpha$  antara 0,01 dan 0,05. Untuk *tracking* objek menggunakan metode *Lucas-Kanade* dan menggunakan metode

pendekatan berbasis data dengan trafik rekaman statis untuk mendeteksi kecepatan. (Kumar et al. n.d.) 2018, menggunakan metode *deep learning* model *Mask R-CNN* dalam mendeteksi objek. Algoritma *SORT (Simple Online and Real-Time Tracking)* dan *DEEPSORT* untuk melacak objek. Algoritma *SORT* menggunakan *filter Kalman*. Untuk menghitung kecepatan dilakukan dua pendekatan yaitu *affine rectification to restore affine-invariant properties of the scene*, dan *scale recovery which estimates vehicle speed in the real world*.

Menggunakan metode *deep learning* dengan model *Faster R-CNN ResNet 101* untuk mendeteksi kendaraan dalam video dan menggunakan algoritma *KLT* sebagai algoritma pelacakan kendaraan. Untuk memperkirakan kecepatan memanfaatkan perpindahan objek antar *frame* yang nantinya akan dikonversi ke jarak dan kecepatan yang sebenarnya di dunia nyata (Giannakeris and Briassouli n.d.) 2018. Masih menggunakan metode *deep learning* dengan model *Faster R-CNN* model *ResNet 101* untuk mendeteksi kendaraan penelitian yang dilakukan Huang, menggunakan algoritma *Histogram-Based Tracking* dalam pelacakannya (Huang n.d.) 2018. Menghitung estimasi kecepatan menggunakan *Image Wrapping* melalui pendekatan merubah domain piksel ke dunia nyata. Menggunakan algoritma *Heuristic* untuk melakukan pelacakan kendaraan dan memperbaiki estimasi kecepatan dengan *Average Filter*. Penelitian ini menghasilkan kecepatan rata-rata 6,9762 mph di jalan raya dan 8,9144 di jalan tol dan perkotaan (Tran et al. n.d.) 2018.

Pada penelitian ini akan difokuskan pada kasus estimasi (perkiraan) kecepatan kendaraan di lalu lintas dalam sebuah video. dikutip dari website Kompas.com, Peraturan pemerintah nomor 79 tahun 2013 pasal 23 ayat 4, bagian kedua: (Kompas.com 2019)

- a Paling rendah 60 kpj dalam kondisi arus bebas dan paling tinggi 100 kpj untuk jalan bebas hambatan.
- b Paling lambat 80 kpj untuk jalan antarkota.
- c Paling tinggi 50 kpj untuk wawasan perkotaan.
- d Paling tinggi 30 kpj untuk kawasan pemukiman.

Dari hasil wawancara yang penulis lakukan pada salah satu anggota SATLANTAS, Aiptu Tony, sistem yang dapat mendeteksi pelanggaran dalam hal kecepatan sangat dibutuhkan. Hal tersebut dikarenakan pelanggaran dalam hal melanggar batas kecepatan sulit untuk dideteksi tanpa alat khusus atau hanya dengan melalui penglihatan petugas. Alasan lain mengapa sistem ini perlu dibuat adalah karena masyarakat Indonesia hanya menaati peraturan lalu lintas ketika terdapat petugas yang bertugas, sedangkan jika menggunakan sistem, perilaku pengendara dapat terpantau selama 24 jam.

Sistem akan menerima *input* berupa video keadaan lalu lintas yang nantinya akan menghasilkan *output* berupa estimasi kecepatan masing-masing kendaraan yang melintas dalam video tersebut. Dari *output* tersebut, diharapkan nantinya sistem pada penelitian ini akan menjadi landasan atau acuan untuk terciptanya suatu aplikasi yang dapat membantu memanajemen lalu lintas secara otomatis. Penelitian ini menggunakan metode *deep learning* dengan model *Faster R-CNN* untuk menyelesaikan masalah deteksi kendaraan pada video. Implementasi algoritma *Centroid Tracking* untuk melakukan pelacakan kendaraan. Estimasi kecepatan kendaraan dilakukan melalui pendekatan perpindahan piksel dari masing-masing objek pada tiap frame yang nantinya akan di konversi ke dunia nyata menggunakan model *parallel line* yang telah diperbaiki.

Berdasarkan latar belakang diatas, maka penulis akan melakukan penelitian dengan judul “Estimasi Kecepatan Kendaraan Melalui Video Pengawas Lalu Lintas Menggunakan *Parallel Line Model*”. Hasil pada penelitian ini adalah kecepatan kendaraan yang melintas dalam video dapat diperkirakan menggunakan metode, model dan pendekatan sesuai yang ditentukan pada penelitian ini.

## 1.2 Tujuan Penelitian

Penelitian ini bertujuan untuk mengestimasi kecepatan kendaraan yang melintas dari sebuah video pengawas lalu lintas.

### 1.3 Manfaat Penelitian

#### 1.3.1 Bagi Mahasiswa

Menambah ilmu dan pengetahuan mahasiswa mengenai kegunaan teori dan praktiknya mengenai apa yang dibahas pada penelitian ini.

#### 1.3.2 Bagi Universitas

Menambah koleksi karya ilmiah Universitas mengenai computer vision mengenai estimasi kecepatan kendaraan melalui video pengawas lalu lintas.

#### 1.3.3 Bagi User

- a. Membantu pihak-pihak terkait seperti kepolisian lalu lintas, dinas perhubungan dan sebagainya guna melakukan pengawasan lalu lintas.
- b. Mengurangi pelanggaran lalu lintas khususnya dalam hal batas kecepatan kendaraan karena adanya kamera pengawas yang dapat mendeteksi kecepatan kendaraan.
- c. Sistem ini nantinya dapat digunakan untuk tilang online terhadap pengendara yang melanggar batas kecepatan.

### 1.4 Pertanyaan Penelitian

Bagaimana sistem dapat mengestimasi kecepatan kendaraan melalui sebuah video pengawas lalu lintas?

### 1.5 Rumusan Masalah

- a. Bagaimana pendekatan kendaraan dilakukan menggunakan metode *deep learning* dengan model *Faster R-CNN*?
- b. Bagaimana pelacakan kendaraan dilakukan menggunakan algoritma *Centroid Tracking*?

- c. Bagaimana estimasi kecepatan kendaraan didapat dengan mengkonversi kecepatan dari piksel ke dunia nyata menggunakan model *parallel line* yang telah diperbaiki?

## 1.6 Batasan Masalah

Untuk membatasi cakupan permasalahan yang akan dibahas dalam penelitian ini, peneliti membuat batasan :

### 1.6.1 Proses

- a. Dataset berupa video dari sebuah kamera pengawas lalu lintas yang pengaturannya telah ditentukan sebelumnya yaitu di ITC permata Hijau, Jl. Arteri Permata Hijau, Grogol Utara Kebayoran Lama Jakarta Selatan.
- b. *Annotasi* dilakukan hanya membagi dataset menjadi dua kelas, yaitu kelas mobil dan kelas motor.
- c. Data dilatih sebanyak dua ratus ribu pelatihan.
- d. Estimasi kecepatan kendaraan hanya menggunakan lintasan yang telah ditentukan penulis.

### 1.6.2 Metode

- a. Metode *deep learning* dengan model *Faster R-CNN* digunakan untuk mendeteksi kendaraan dalam video
- b. Algoritma *Centroid Tracking* diimplementasikan dalam menyelesaikan masalah pelacakan kendaraan yang telah terdeteksi
- c. Mengestimasi kecepatan menggunakan pendekatan perpindahan kendaraan pada tiap frame yang selanjutnya akan dikonversi ke kecepatan dalam dunia nyata dalam satuan km/h menggunakan model *parallel line* yang telah diperbaiki
- d. Metode pengembangan sistem menggunakan metode *prototyping* dengan penulis merangkap juga sebagai kliennya. Jadi pada tahap evaluasi, penulis mengevaluasi sistem sesuai

dengan apa yang penulis harapkan. Penulis hanya menggunakan tahap-tahap yang dibutuhkan dalam penelitian ini dikarenakan klien adalah penulis sendiri.

#### 1.6.3 Tools

- a. Menggunakan bahasa pemrograman *python* dengan *sublime text* sebagai *code editor*
- b. Menggunakan *software Anaconda* sebagai *packaging* dalam membantu membangun sistem
- c. Membangun sistem deteksi objek pada video menggunakan *framework Tensorflow* yang telah disediakan Google
- d. Menggunakan *software labeling* dalam membantu membuat *annotasi* pada tiap data
- e. Menggunakan kamera *Samsung 32 MP* dalam pengambilan video yang diasumsikan sebagai kamera pengawas lalu lintas
- f. Menggunakan *Google Colab* untuk *training data*

### 1.7 Metodologi Penelitian

#### 1.7.1 Metode Pengumpulan Data

Penelitian ini diawali dengan melakukan pembelajaran terlebih dahulu dari beberapa artikel, paper, jurnal, makalah maupun situs internet mengenai pembahasan deteksi objek, pelacakan objek, konsep perhitungan kecepatan kendaraan, *deep learning*, model *Faster R-CNN*, *centroid tracking* dan model *parallel line*. Selanjutnya penulis melakukan observasi untuk mendapatkan *dataset* dan validasi kecepatan dari seorang yang terbiasa dalam bidang lalu lintas.

#### 1.7.2 Metode Pengembangan Sistem

Pengembangan sistem pada metode ini menggunakan model *prototyping*. Model ini didasari pada konsep model bekerja (*working model*). Model terdiri dari 7 tahapan, namun pada penelitian ini, penulis

hanya menggunakan 5 tahapan dari model *prototyping*, yaitu analisis kebutuhan, desain *prototyping*, *development*, *testing*, evaluasi sistem.

## 1.8 Sistematika Penulisan

### BAB 1 Pendahuluan

Bab ini membahas tentang Latar Belakang, Perumusan Masalah, Batasan Masalah, Tujuan penelitian, Manfaat Penelitian yang dilakukan serta Sistematika penulisan.

### BAB 2 Landasan Teori

Pada bab ini membahas tentang semua teori yang menjadi acuan penulis dalam menyelesaikan masalah yang dihadapi seperti teori tentang video, *deep learning*, *Faster R-CNN*, *Centroid Tracking* dan juga membahas tentang penelitian-penelitian sebelumnya yang menjadi acuan penulis.

### BAB 3 Metode Penelitian

Membahas mengenai langkah-langkah bagaimana sistem dapat mengestimasi kecepatan melalui sebuah video pengawas lalu lintas, termasuk metode pengumpulan data dan metode perancangan sistem.

### BAB 4 Implementasi Eksperimen

Bab ini membahas tentang implementasi dan pengujian sistem

### BAB 5 Hasil dan Pembahasan

Pada bab ini dibahas mengenai hasil yang didapat dari berbagai langkah dan metode yang telah dipilih

### BAB 6 Penutup

Bab ini berisi kesimpulan dari apa yang telah dibahas pada seluruh bab dan juga saran-saran dari penulis kepada pembaca mengenai apa yang harus diperbaiki

pada penelitian ini, sehingga kekurangan-kekurangan pada penelitian ini dapat diatasi.



## BAB II

### LANDASAN TEORI

#### 2.1 Video

Video adalah bentuk dinamis dari gambar statis. Dengan kata lain, video terdiri dari serangkaian gambar statis dalam urutan tertentu, dan setiap gambar disebut '*frame*'. *Frame* tersebut diproyeksikan secara terus-menerus ke layar dengan kecepatan konstan, sehingga menghasilkan efek dinamik yang mempengaruhi tingkat kehadiran visual. Sementara itu, video juga dapat dikategorikan menjadi analog dan digital. (Gong et al. 2019). Karena video merupakan serangkaian gambar statis, jadi video bisa diproses layaknya kita memproses gambar. Gambar tidak lain adalah kumpulan piksel dalam ruang warna yang berbeda. Gambar lengkap sebagai set yang terdiri dari sampel kecil. Sampel-sampel ini disebut piksel. Mereka adalah elemen terkecil dalam gambar digital apa pun (Himanshu Singh 2019).

Gambar bernilai lebih dari seribu kata tergantung kita melihat dari sisi tertentu. Jadi, mata dan otak kita mampu mengekstraksi informasi terperinci jauh melampaui apa yang dapat dijelaskan dalam teks. Kemampuan inilah yang ingin kita tiru dalam "*computer vision*" yang mana kamera menggantikan mata dan perangkat lunak pengolah (video dan gambar) menggantikan otak manusia (Moeslund 2019).

#### 2.2 Deteksi Objek

Pengenalan objek, salah satu tugas penting dari pengenalan gambar, terutama ditujukan untuk pengenalan gambar yang terlihat. Itu dapat secara akurat mendefinisikan dan menggambarkan objek dengan atribut dan fitur penampilan geometris, tekstur dan bahan gambar. Dalam arti luas, proses pengenalan dapat membedakan objek dari latar belakang dan objek mencurigakan lainnya, seperti mobil dan jalan, yaitu deteksi objek (Gong et al. 2019).

Deteksi objek adalah istilah yang digunakan ketika Anda memiliki beberapa *instance* objek dalam gambar. Dalam deteksi objek, Anda ingin menentukan semua contoh beberapa objek (misalnya, orang, mobil, tanda, dll.) Dan menggambar kotak pembatas di sekitarnya (Michelluci and Moolayil 2019).

Deteksi objek adalah proses menemukan *instance* objek dari kelas tertentu, seperti wajah, mobil, dan pohon, dalam gambar atau video. Tidak seperti klasifikasi, deteksi objek dapat mendeteksi banyak objek, serta lokasinya di gambar. Detektor objek akan mengembalikan daftar objek yang terdeteksi dengan informasi berikut untuk setiap objek: (Vasilev 2019)

1. Kelas objek (orang, mobil, pohon, dan sebagainya).
2. Probabilitas (atau skor kepercayaan) dalam rentang [0, 1], yang menyampaikan seberapa yakin detektor terhadap objek yang ada di lokasi tersebut. Ini mirip dengan *output* dari classifier biasa.
3. Koordinat wilayah segi empat dari gambar tempat objek berada.

Kotak ini disebut kotak pembatas.

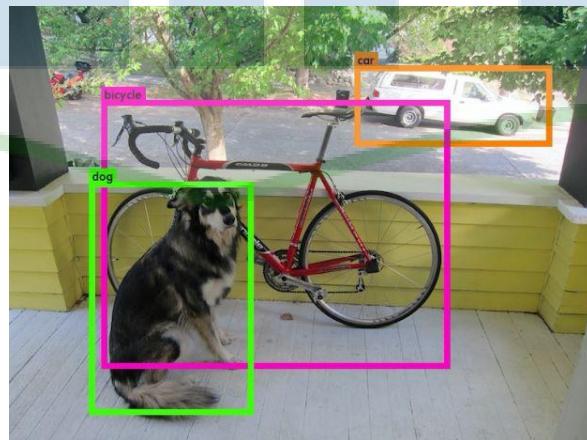
Terdapat beberapa model yang dapat digunakan dalam mendeteksi objek pada gambar ataupun video, diantaranya adalah *Faster R-CNN*, *YOLO*, *MobileNet*, *Mask R-CNN* dan sebagainya. Kebanyakan model objek deteksi memanfaatkan wilayah pada gambar dalam menentukan objek. Sehingga model tidak melihat gambar secara lengkap. *Faster R-CNN* adalah model objek bagus dengan komputasi yang tidak terlalu berat untuk menyelesaikan masalah deteksi objek. Model *Mask R-CNN* merupakan perkembangan dari *Faster R-CNN* namun lebih fokus untuk masalah *object segmentation*. Pada model *Mask R-CNN* objek dalam gambar tidak hanya diberikan *bounding box*, namun objek juga dilapisi semacam topeng yang membentuk persis objek tersebut.

Deteksi objek pada gambar adalah suatu proses yang digunakan untuk menentukan keberadaan objek tertentu dalam suatu gambar. Konsep deteksi objek pada gambar adalah dengan melakukan pembacaan fitur ekstraksi pada gambar yang telah *diinput* yang dapat dilakukan dengan berbagai macam

metode. fitur ekstraksi tersebut akan dibandingkan dengan fitur dari objek referensi. Hasil dari perbandingan dapat menentukan apakah suatu objek yang dideteksi merupakan objek yang dimaksud atau bukan. *Object detection* menentukan keberadaan suatu objek dan ruang lingkupnya serta lokasi pada sebuah 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 mendekripsi adanya objek sedangkan *hard detection* mendekripsi adanya objek serta lokasi objek (Dewi 2018).

Proses dalam deteksi objek pada video sama dengan proses dekteksi objek dalam gambar. Video terdiri dari beberapa gambar atau frame Dalam video, pertama video dipecah menjadi beberapa frame, pada setiap frame dilakukan proses pendeksian objek seperti yang dilakukan pada proses deteksi objek pada gambar. Setelah itu, frame yang telah diproses disatukan kembali menjadi video utuh. Untuk memproses deteksi objek pada video, *resource* atau komputasi yang dibutuhkan cukup berat bahkan mustahil dilakukan dengan *resource* atau komputasi yang standar. Pemrosesan deteksi objek pada video dilakukan menggunakan GPU (*Graphic Processing Unit*).

Contoh deteksi objek pada gambar, yang mana objek yang dimaksud berupa anjing, sepeda dan mobil :



### Gambar 2.1 Contoh Deteksi Objek Pada Gambar

Sumber : (Redmon 2018)

#### 2.3 Pelacakan Objek

Pelacakan objek (*Object Tracking*) mengacu pada menemukan objek bergerak yang menarik (misalnya, kendaraan, pejalan kaki, hewan, dll.) Dalam urutan video kontinu. Ini adalah cabang penting dari visi komputer, dan memiliki berbagai aplikasi dalam bimbingan militer, navigasi visual, robotika, transportasi cerdas, keselamatan publik, dan bidang lainnya (Gong et al. 2019).

Kunci dari pelacakan objek adalah mengekstraksi fitur yang kuat dari objek yang bergerak dan mengidentifikasinya secara akurat. Terkadang, kita juga perlu mempertimbangkan biaya waktu dari algoritma pelacakan dalam menerapkan sistem waktu nyata. Metode pelacakan objek bergerak dapat dibagi menjadi dua jenis metode berbasis analisis bergerak dan metode berbasis pencocokan gambar. Metode yang didasarkan pada analisis bergerak umumnya diimplementasikan oleh diferensiasi antar-frame dan segmentasi aliran optik. Perbedaan antar-bingkai mengurangi frame yang berdekatan terlebih dahulu, dan kemudian menggunakan nilai ambang tertentu untuk mengekstrak objek bergerak. Metode segmentasi aliran optik mendeteksi objek yang bergerak dengan kecepatan berbeda antara objek dan latar belakang. Sistem pelacakan objek bergerak umumnya berisi langkah-langkah berikut: (Gong et al. 2019)

- a. Mengekstrak deskriptor efektif objek. Sistem pelacakan objek tergantung pada keefektifan deskriptor yang digunakan untuk menangkap karakteristik objek. Fitur yang paling banyak digunakan termasuk tepi gambar, kontur, bentuk, tekstur, momen, koefisien transformasi, dll.
- b. Metrik kesamaan. Dalam pelacakan objek bergerak, kesamaan target bergerak antara frame yang berdekatan biasanya diukur dengan beberapa metrik kesamaan seperti jarak Eropa, jarak Mahalanobis, jarak papan catur, jarak tertimbang, koefisien kesamaan, dan koefisien korelasi.

- c. Pencocokan wilayah objek. Dalam pelacakan objek bergerak, memperkirakan wilayah objek bergerak dapat sangat mengurangi pencarian lengkap dan mempercepat sistem pelacakan. Algoritma pencarian wilayah objek yang umum digunakan berisi *filter Kalman*, *filter partikel*, pergeseran rata-rata dan sebagainya.

Menurut (Howse 2015), Ada empat langkah utama yang terlibat dalam jenis pelacakan :

- a. Temukan fitur dalam gambar dan adegan referensi. Fitur adalah titik yang cenderung mempertahankan penampilan yang sama jika dilihat dari jarak atau sudut yang berbeda. Misalnya, sudut sering memiliki karakteristik ini.
- b. Temukan deskriptor untuk setiap *set* fitur (fitur referensi dan adegan). Deskriptor adalah vektor data tentang fitur. Beberapa fitur tidak cocok untuk menghasilkan deskriptor, sehingga gambar memiliki deskripsi yang lebih sedikit daripada fitur.
- c. Temukan kecocokan antara dua *set* deskriptor (deskriptor referensi dan adegan). Jika kita membayangkan deskriptor sebagai titik dalam ruang multidimensi, kecocokan didefinisikan dalam ukuran jarak antara titik. Deskriptor yang cukup dekat satu sama lain dianggap cocok. Ketika sepasang deskriptor adalah pasangan, kami dapat juga mengatakan bahwa pasangan fitur yang mendasarinya adalah pasangan.
- d. Temukan *homografi* antara gambar referensi dan gambar yang cocok dalam adegan. *Homografi* adalah transformasi 3D yang akan diperlukan untuk menyejajarkan dua gambar 2D yang diproyeksikan (atau sedekat mungkin untuk meluruskannya). Ini dihitung berdasarkan *point* fitur pencocokan dua gambar. Dengan menerapkan *homografi* ke sebuah persegi panjang, kita bisa mendapatkan garis besar objek yang dilacak.

Terdapat beberapa algoritma dalam pelacakan objek seperti *centroid tracking* yang melacak objek berdasarkan jarak terdekat pada tiap *frame*. Jarak terdekat diambil dari perhitungan *centroid* pada tiap objek. Algoritma

*background subtraction*, algoritma *Kalman Filter* dan sebagainya. Dalam algoritma *background subtraction* setiap *frame* disubstraksi, hasil substraksi dianalisis untuk menemukan pergerakan objek.

## 2.4 Kecerdasan Buatan

Kecerdasan buatan atau *Artificial Intelligence* merupakan ilmu dan teknik pembuatan mesin cerdas, khususnya program komputer cerdas. Hal ini terkait dengan tugas yang sama dengan menggunakan komputer untuk memahami kecerdasan manusia, tetapi *Artificial Intelligence* tidak harus membatasi dirinya terhadap metode yang diamati secara biologis (Dewi 2018).

*Artificial Intelligence* (AI) dapat didefinisikan (meskipun definisi ini mungkin tidak unik) sebagai sistem yang dapat berinteraksi dengan lingkungannya. Juga, mesin AI diberkahi dengan sensor yang memungkinkan mereka untuk mengetahui lingkungan tempat mereka berada, dan alat yang dengannya mereka dapat berhubungan kembali dengan lingkungan (Vasilev 2019). Niat menciptakan mesin yang bisa meniru aspek kecerdasan manusia seperti penalaran, pembelajaran, dan pemecahan masalah melahirkan perkembangan teknologi AI. AI benar-benar menyaingi sifat manusia. Dengan kata lain, AI membuat mesin berpikir dan berperilaku seperti manusia. (Eckroth 2018).

## 2.5 Datasets

Pada dasarnya, sebuah *Dataset* hanyalah serangkaian elemen, di mana setiap elemen berisi satu atau lebih *tensor*. Biasanya, setiap elemen akan menjadi satu contoh pelatihan atau kumpulan dari mereka. Ide dasarnya adalah bahwa pertama-tama Anda membuat *Dataset* dengan beberapa data, dan kemudian Anda memanggil metode itu (Michelluci and Moolayil 2019). Dalam membangun sebuah model, dataset akan dibagi menjadi 2 kelompok, yaitu data *train* dan data *test*. Dalam teorinya, data *train* harus lebih banyak daripada data

*test*. Merujuk dari (Dewi 2018), pembagian data *train* dan data *test* sebesar 80% dan 20%.

## 2.6 Machine Learning

*Machine Learning* merupakan cabang ilmu dari *Artificial Intelligence* yang memungkinkan komputer memiliki kemampuan untuk belajar tanpa perlu di program lagi. Secara sederhana *machine learning* membangun sebuah algoritma yang memungkinkan program komputer untuk belajar dan melakukan tugasnya sendiri tanpa adanya instruksi dari penggunanya. Algoritma semacam ini bekerja dengan cara membangun sebuah model dari *input* atau masukan untuk dapat menghasilkan suatu prediksi atau pengambilan keputusan berdasarkan data yang ada (Dewi 2018).

*Machine learning* adalah alat yang digunakan untuk pemrosesan data skala besar. Ini sangat cocok untuk dataset kompleks yang memiliki sejumlah besar variabel dan fitur. *Machine learning* melakukan yang terbaik ketika digunakan pada *dataset* besar, sehingga meningkatkan daya analitik dan prediktifnya. *Machine learning* dapat mengakses dataset besar di mana mereka dapat menemukan pola dan keteraturan yang tersembunyi dalam data. *Machine learning* adalah otak yang memungkinkan mesin untuk menganalisis data yang dicerna melalui sensornya untuk merumuskan jawaban yang tepat. (Vasilev 2019).

*Machine learning* terdiri dari perancangan algoritma prediksi yang efisien dan akurat. Seperti di bidang ilmu komputer lainnya, beberapa ukuran penting kualitas algoritma ini adalah kompleksitas waktu dan ruangnya. Namun, dalam *Machine learning*, kita akan membutuhkan tambahan gagasan kompleksitas sampel untuk mengevaluasi ukuran sampel yang diperlukan untuk algoritma untuk mempelajari serangkaian konsep. Beberapa aplikasi *machine learning* diantaranya : (Mohri, Rostamizadeh, dkk 2018).

### a. Document Classification

- b. *Natural Language Processing*
- c. *Speech Processing Application*
- d. *Computer Vision Application*

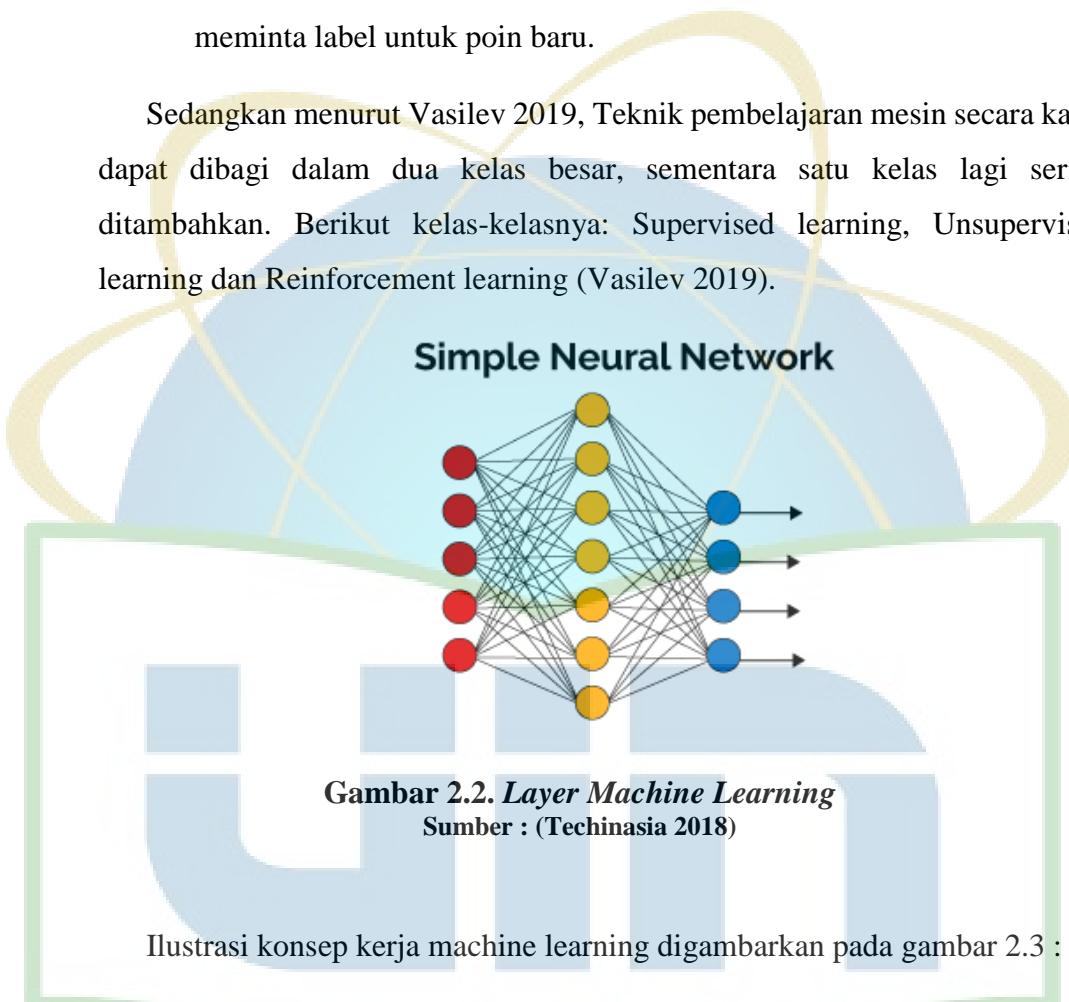
Menurut (Mohri, Rostamizadeh, dkk 2018), pembagian *machine learning* sebagai berikut :

- a. *Supervised Learning* : Pelajar menerima satu set contoh berlabel sebagai data pelatihan dan membuat prediksi untuk semua poin yang tidak terlihat. Ini adalah skenario paling umum yang terkait dengan masalah klasifikasi, regresi, dan peringkat.
- b. *Unsupervised Learning* : Pelajar secara eksklusif menerima data pelatihan yang tidak berlabel, dan membuat prediksi untuk semua poin yang tidak terlihat. Karena secara umum tidak ada contoh berlabel yang tersedia di pengaturan itu, itu bisa sulit untuk mengevaluasi kinerja pelajar secara kuantitatif.
- c. *Semi-Supervised Learning* : Pelajar menerima sampel pelatihan yang terdiri dari data yang diberi label dan tidak berlabel, dan membuat prediksi untuk semua poin yang tidak terlihat.
- d. *Tracedutive Inference* : pelajar menerima sampel pelatihan berlabel bersama dengan serangkaian poin tes yang tidak berlabel. Namun, tujuan dari *tracedutive inference* adalah untuk memprediksi label hanya untuk titik uji khusus ini.
- e. *On-line learning* : melibatkan beberapa putaran di mana fase pelatihan dan pengujian dicampur. Pada setiap putaran, pelajar menerima poin pelatihan yang tidak berlabel, membuat prediksi, menerima label yang benar, dan menimbulkan kerugian. Tujuan dalam *on-line learning* adalah untuk meminimalkan kerugian kumulatif pada semua putaran atau untuk meminimalkan penyesalan, yaitu perbedaan dari kerugian kumulatif yang terjadi dan ahli terbaik di belakang.
- f. *Reinforcement Learning* : Fase pelatihan dan pengujian juga saling terkait dalam pembelajaran penguatan. Untuk mengumpulkan

informasi, pelajar secara aktif berinteraksi dengan lingkungan dan dalam beberapa kasus memengaruhi lingkungan, dan menerima hadiah langsung untuk setiap tindakan.

- g. *Active Learning* : Pelajar secara adaptif atau interaktif mengumpulkan contoh-contoh pelatihan, biasanya dengan menanyakan oracle untuk meminta label untuk poin baru.

Sedangkan menurut Vasilev 2019, Teknik pembelajaran mesin secara kasar dapat dibagi dalam dua kelas besar, sementara satu kelas lagi sering ditambahkan. Berikut kelas-kelasnya: Supervised learning, Unsupervised learning dan Reinforcement learning (Vasilev 2019).



**Gambar 2.3. Konsep Kejra Machine Learning**  
Sumber : <https://www.guru99.com/machine-learning-vs-deep-learning.html>

## 2.7 Deep Learning

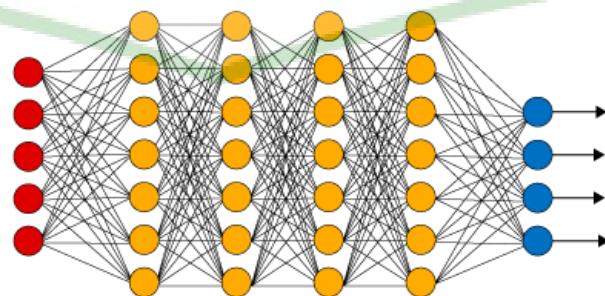
*Deep learning* adalah sebuah pendekatan dalam penyelesaian masalah pada sistem pembelajaran komputer yang menggunakan konsep hierarki. Konsep hierarki membuat komputer mampu mempelajari konsep yang kompleks dengan menggabungkan dari konsep-konsep yang lebih sederhana. Jika digambarkan sebuah graf bagaimana konsep tersebut dibangun di atas konsep yang lain, graf ini akan dalam dengan banyak *layer*, hal tersebut menjadi alasan disebut sebagai *deep learning* (pembelajaran mendalam) (Dewi 2018).

*Deep learning* mengacu pada jaringan saraf dengan banyak lapisan. Ini semacam kata kunci, tetapi teknologi di baliknya nyata dan cukup canggih. Keuntungan utama dari *Deep learning* adalah bahwa menambahkan lebih banyak data dan lebih banyak daya komputasi sering menghasilkan hasil yang lebih akurat, tanpa upaya signifikan yang diperlukan untuk rekayasa (Eckroth 2018).

Ide utama dalam *deep learning* terbagi menjadi dua (Gong et al. 2019) :

- Jaringan saraf dengan multi-*layer* artikel memiliki kemampuan belajar fitur yang sangat baik, dan belajar lebih baik dengan mempertimbangkan karakteristik penting dari data, yang kondusif untuk visualisasi atau klasifikasi
- Kesulitan pelatihan jaringan saraf dalam dapat secara efektif diatasi dengan *Unsupervised learning* dari *layer-wise*.

**Deep Learning Neural Network**



**Gambar 2.4. Layer Deep Learning**

Sumber : (Techinasia 2018)

Ilustrasi konsep kerja *deep learning* digambarkan pada gambar 2.5 :



**Gambar 2.5. Konsep Kejra Deep Learning**

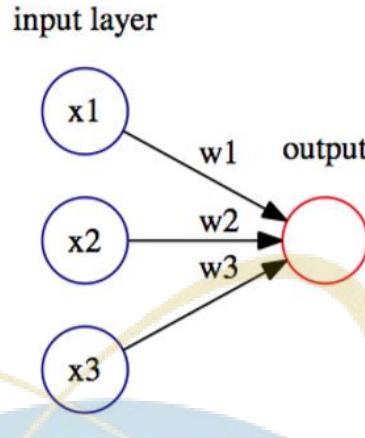
Sumber : <https://www.guru99.com/machine-learning-vs-deep-learning.html>

## 2.8 Artificial Neural Network

Contoh pertama dari *neural network* disebut *perceptron*, dan ini ditemukan oleh *Frank Rosenblatt* pada tahun 1957. *Perceptron* adalah algoritma klasifikasi yang sangat mirip dengan *regresi logistik*. Seperti *regresi logistik*, ia memiliki bobot,  $w$ , dan *outputnya* adalah fungsi,  $f(\vec{x} \cdot \vec{w})$ , dari produk titik,  $\vec{x} \cdot \vec{w}$  (atau dari  $f \sum_i w_i \cdot x_i$ ) bobot dan *input* (Vasilev 2019).

Menurut (Vasilev 2019) karakteristik dari *neural network* adalah sebagai berikut :

- Pemrosesan informasi terjadi dalam bentuknya yang paling sederhana, lebih dari elemen sederhana yang disebut neuron.
- Neuron terhubung dan mereka bertukar sinyal di antara mereka melalui tautan koneksi.
- Tautan koneksi antara neuron dapat menjadi lebih kuat atau lebih lemah, dan ini menentukan bagaimana informasi diproses.
- Setiap neuron memiliki keadaan internal yang ditentukan oleh semua koneksi yang masuk dari neuron lain.
- Setiap neuron memiliki fungsi aktivasi berbeda yang dihitung pada keadaannya, dan menentukan sinyal *outputnya*.



**Gambar 2.6. Perceptron Sederhana Dengan Layer Input dan Output**

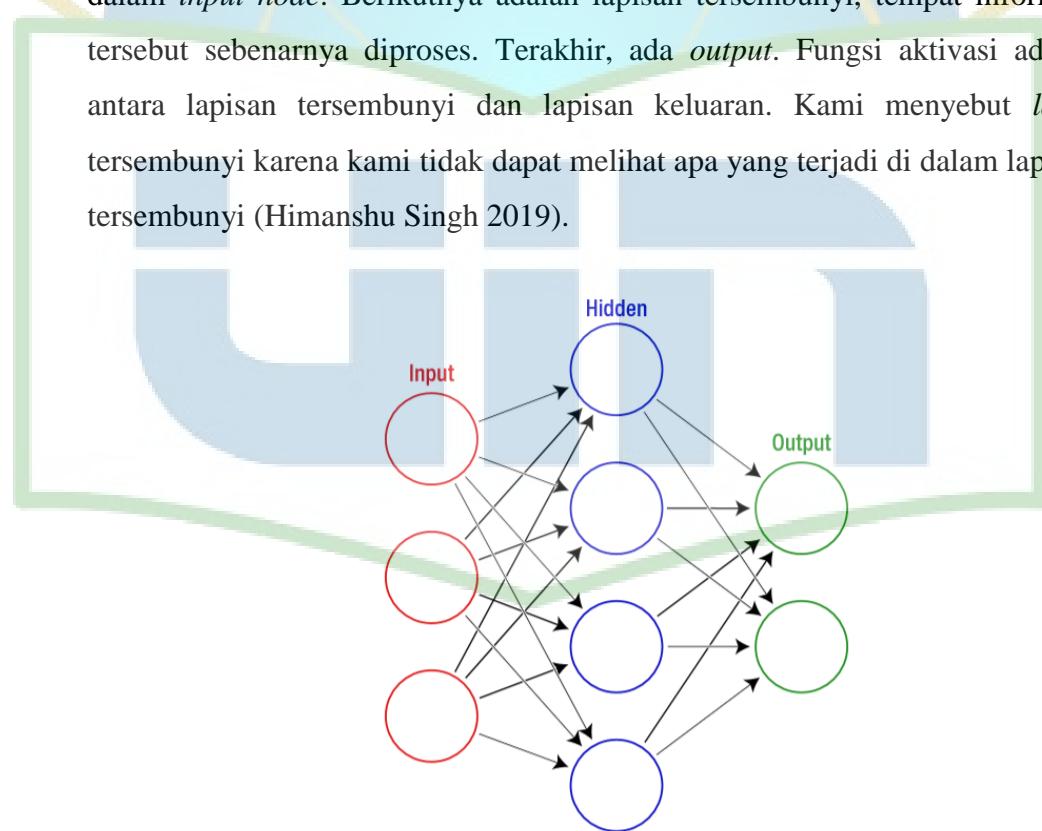
Sumber : (Vasilev 2019)

Perceptron memiliki kelemahan karena hanya terbatas pada kelas yang dipisahkan secara *linear*. Namun masalah ini dapat diatasi dengan pendekatan *neural network* yang berbeda. Sebuah perceptron multilayer klasik memiliki beberapa perceptron yang saling berhubungan, seperti unit yang disusun dalam lapisan sekuensial yang berbeda (lapisan *input*, satu atau lebih lapisan tersembunyi, dan lapisan keluaran). Setiap unit lapisan terhubung ke semua unit lapisan berikutnya. Pertama, informasi disajikan ke lapisan *input*, lalu kami menggunakan untuk menghitung *output* (atau aktivasi),  $y_i$ , untuk setiap unit lapisan tersembunyi pertama. Kami merambat ke depan, dengan *output* sebagai *input* untuk lapisan berikutnya dalam jaringan (maka *feedforward*), dan seterusnya sampai kami mencapai *output* (Vasilev 2019).

*Artificial neural network (ANN)* atau jaringan saraf tiruan adalah sistem komputerisasi sebagai pemroses informasi yang memiliki karakter mirip dengan jaringan syaraf biologi pada saat menangkap informasi dari ‘dunia luar’. Maksud sebenarnya dari JST adalah berusaha membuat sebuah model sistem komputasi informasi yang dapat menirukan rangkaian cara kerja jaringan syaraf biologis (Hakim, 2017). Jaringan syaraf tiruan tidak diprogram untuk menghasilkan suatu keluaran tertentu. Keluaran yang dihasilkan oleh jaringan syaraf tiruan didapat berdasarkan pengalamannya selama mengikuti proses pembelajaran. Jadi tingkat akurasi keluaran yang dihasilkan jaringan syaraf

tiruan tergantung seberapa baik sistem dalam mengikuti proses pembelajaran. JST Merupakan gabungan dari pendekatan statistik dan pendekatan sintaks. Dengan gabungan dari dua metode maka JST merupakan pengenalan pola yang lebih akurat (Rohimah 2017).

*ANN* memproses informasi menggunakan neuron buatan. Informasi ditransfer dari satu neuron buatan ke neuron buatan lainnya, yang akhirnya mengarah ke fungsi aktivasi, yang bertindak seperti otak dan membuat keputusan (Himanshu Singh 2019). Ini berarti bahwa setiap neuron terhubung ke setiap neuron di lapisan berikutnya. Lapisan pertama menerima *input* dan lapisan terakhir memberikan keluaran. Struktur jaringan, yang berarti jumlah neuron dan koneksinya, diputuskan sebelumnya dan tidak dapat berubah, setidaknya tidak selama pelatihan. Juga, setiap *input* harus memiliki jumlah nilai yang sama (Eckroth 2018). Jadi, informasi yang akan diproses disimpan dalam *input node*. Berikutnya adalah lapisan tersembunyi, tempat informasi tersebut sebenarnya diproses. Terakhir, ada *output*. Fungsi aktivasi adalah antara lapisan tersembunyi dan lapisan keluaran. Kami menyebut *layer* tersembunyi karena kami tidak dapat melihat apa yang terjadi di dalam lapisan tersembunyi (Himanshu Singh 2019).



**Gambar 2.7. Artificial Neural Network Sederhana**  
 Sumber : (Himanshu Singh 2019)

Hasil perhitungan pada tiap *neuron* dapat didefinisikan melalui rumus :

$$y = \sum_i x_i \cdot w_i \quad (2.1)$$

Keterangan :

y : hasil perhitungan antara *node input* dengan bobot

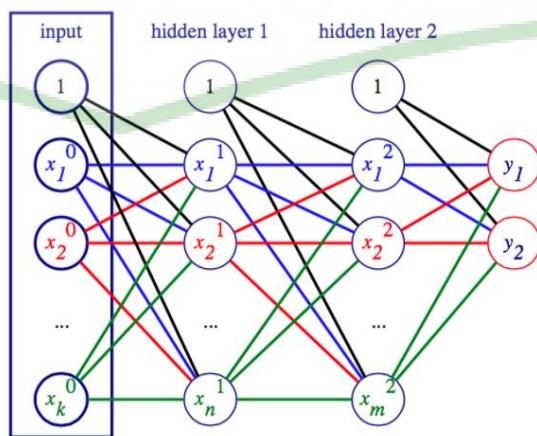
x : nilai pada setiap *node*

w : nilai bobot

Pikirkan *hidden layer* sebagai representasi abstrak dari data *input*. Ini adalah cara jaringan saraf memahami fitur-fitur data dengan logika internalnya sendiri. Namun, jaringan saraf adalah model yang tidak dapat ditafsirkan (Vasilev 2019).

### 2.7.1 Multi-layer Network

*Neural network* satu *layer* hanya dapat mengklasifikasikan kelas yang dapat dipisahkan secara *linear*. Tetapi tidak ada yang mencegah kita untuk memperkenalkan lebih banyak *layer* antara *input* dan *output*. Lapisan tambahan ini disebut lapisan tersembunyi. Unit dari satu lapisan terhubung ke semua unit dari lapisan sebelumnya dan berikutnya (karenanya terhubung penuh). Setiap koneksi memiliki bobotnya sendiri (Vasilev 2019).



**Gambar 2.8. Artificial Neural Network Multi-layer**  
Sumber : (Vasilev 2019)

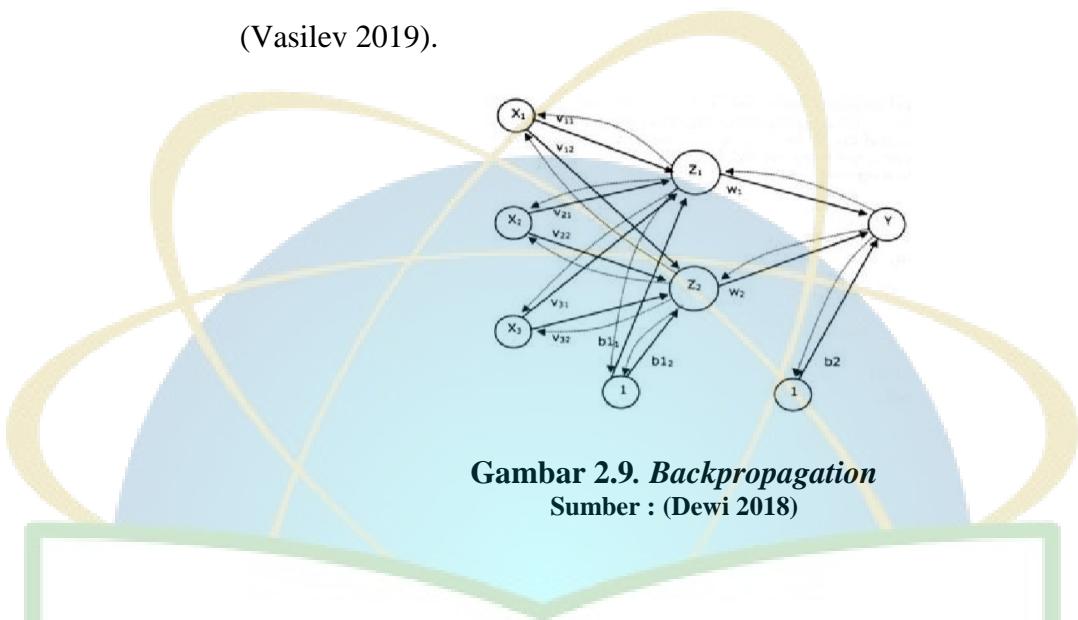
Sumber *node* di *input layer* dari jaringan menyediakan masing-masing elemen dari pola aktivasi (vector *input*), yang merupakan sinyal *input* yang diaplikasikan ke neuron-neuron pada *layer* kedua (*hidden layer* pertama). Sinyal *output* dari *layer* kedua digunakan sebagai *input*-*input* ke *layer* ketiga, dan seterusnya sampai ke sisa dari jaringan (Dewi 2018). *Neural netwoek multi-layer* mengandung banyak *layer* komputasi, *layer* menengah tambahan (antara *input* dan *output*) disebut sebagai *hidden layer* karena perhitungan yang dilakukan tidak terlihat oleh pengguna (Aggarwal 2018).

Arsitektur spesifik dari *neural network multi-layer* disebut sebagai *feed-forward network*, karena lapisan berturut-turut saling memberi makan satu sama lain dalam arah maju dari *input* ke *output*. Arsitektur default dari *feed-forward network* mengasumsikan bahwa semua *node* dalam satu *layer* terhubung ke orang-orang dari *layer* berikutnya. Oleh karena itu, arsitektur *neural network* hampir sepenuhnya didefinisikan (Aggarwal 2018).

#### 2.7.2 Backpropagation

*Deep neural network* yang berisi puluhan *layer* yang dapat dijelaskan dalam beberapa ratus baris kode. Semua pembelajaran bobot dilakukan secara otomatis oleh algoritma *backpropagation* yang menggunakan pemrograman dinamis untuk mengetahui langkah-langkah pembaruan parameter yang rumit pada grafik komputasi yang mendasarinya. Sebagian besar kemudahan dalam pelatihan *neural network* ini disebabkan oleh algoritma *backpropagation*, yang melindungi analis dari secara eksplisit mengerjakan langkah-langkah pembaruan parameter dari apa yang sebenarnya merupakan masalah optimasi yang sangat rumit. modularitas dalam desain *neural network* diterjemahkan menjadi

modularitas dalam mempelajari parameteranya; nama spesifik untuk tipe terakhir dari modularitas adalah "*backpropagation*" (Aggarwal 2018). *Backpropagation* adalah salah satu algoritma yang paling sulit untuk dipahami, tetapi yang Anda butuhkan hanyalah pengetahuan tentang kalkulus diferensial dasar dan aturan rantai (Vasilev 2019).



Algoritma *backpropagation* memanfaatkan aturan rantai kalkulus yang berbeda, yang menghitung gradien kesalahan dalam hal penjumlahan produk-produk gradien lokal melalui berbagai jalur dari sebuah *node* ke *output*. Algoritma *backpropagation* adalah aplikasi langsung dari pemrograman dinamis. Ini berisi dua fase utama, masing-masing disebut *forward phase* dan *backward phase* : (Aggarwal 2018)

- a. *Forward Phase* : Dalam fase ini, *input* untuk *instance* pelatihan dimasukkan ke dalam *neural network*. Ini menghasilkan *cascade* maju komputasi di seluruh *layer*, menggunakan set bobot saat ini. *Output* yang diprediksi akhir dapat dibandingkan dengan contoh pelatihan dan turunan dari fungsi kerugian sehubungan dengan *output* dihitung. Turunan dari kerugian ini sekarang perlu dihitung

sehubungan dengan bobot di semua *layer* dalam *backward phase*

- b. Tujuan utama dari *backward phase* adalah untuk mempelajari gradien fungsi kerugian sehubungan dengan bobot yang berbeda dengan menggunakan aturan rantai kalkulus yang berbeda. Gradien ini digunakan untuk memperbarui bobot. Karena gradien ini dipelajari dalam arah mundur, mulai dari simpul keluaran, proses pembelajaran ini disebut sebagai *backward phase*. Pertimbangkan urutan unit tersembunyi.

### 2.7.3 Activation Function

Jika neuron tidak memiliki fungsi aktivasi, *outputnya* akan menjadi jumlah terbobot dari *input*, yang merupakan fungsi *linier*. Kemudian seluruh jaringan saraf, yaitu, komposisi *neuron*, menjadi komposisi fungsi *linier*, yang juga merupakan fungsi *linier*. Ini berarti bahwa bahkan jika kita menambahkan *hidden layer*, jaringan masih akan setara dengan model regresi linier sederhana, dengan segala keterbatasannya. Untuk mengubah jaringan menjadi fungsi *non-linear*, kami akan menggunakan fungsi aktivasi *non-linear* untuk neuron (Vasilev 2019).

Beberapa fungsi aktivasi yang terdapat dalam neural network diantaranya :

- a. *Sigmoid biner Activation* : Turunan dari aktivasi sigmoid sangat sederhana, ketika diekspresikan dalam bentuk *output* dari sigmoid, daripada *input*. Fungsi aktivasi ini menjadikan nilai antara 0 sampai 1 :

$$\varphi = \frac{1}{1+e^{-x}} \quad (2.2)$$

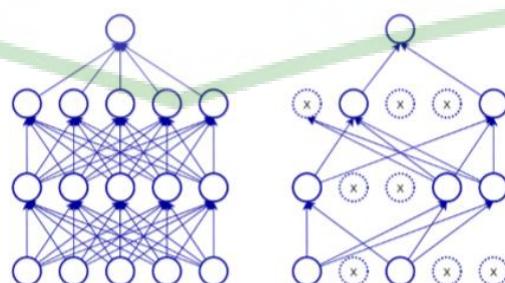
- b. *Sigmoid Bipolar Activation* : Seperti dalam kasus aktivasi *sigmoid biner*, fungsi aktivasi ini menjadikan nilai antara 1 sampai -1:

$$\varphi = \frac{1-e^{-x}}{1+e^{-x}} \quad (2.3)$$

### 2.9 Dropout Regularization

*Regularization* adalah modifikasi apa pun yang dibuat untuk algoritma pembelajaran yang dimaksudkan untuk mengurangi kesalahan generalisasi tetapi bukan kesalahan pelatihannya (Vasilev 2019). *Regularization* merupakan teknik yang digunakan untuk mengurangi *overfitting*, yakni kondisi dimana sistem jaringan syaraf tiruan mampu belajar dengan baik dengan data pelatihan, namun tidak bisa menggeneralisasi pada data tes. Terdapat beberapa teknik dalam regularisasi, misalnya L2 *regularization* dan *dropout* (Dewi 2018).

*Dropout* adalah teknik regularisasi, yang dapat diterapkan pada *output* dari beberapa lapisan jaringan. *Dropout* secara acak dan berkala menghapus beberapa *neuron* (bersama dengan koneksi *input* dan *output*) dari jaringan. Selama pelatihan mini-batch, setiap neuron memiliki probabilitas  $p$  untuk secara stokastik turun. Ini untuk memastikan bahwa tidak ada neuron yang akhirnya terlalu mengandalkan neuron lain dan "belajar" sesuatu yang berguna untuk jaringan. *Dropout* dapat diterapkan setelah lapisan *convolutional*, *pooling*, atau terhubung sepenuhnya (Vasilev 2019).



**Gambar 2.10. Dropout**  
Sumber : (Vasilev 2019)

## 2.10 Convolutional Neural Network

Karakteristik pendefinisian penting dari *convolutional neural network* adalah operasi, yang disebut sebagai *convolution*. Operasi *convolution* adalah operasi titik-produk antara set bobot struktur-grid dan *input* struktur-grid serupa yang diambil dari lokasi spasial berbeda dalam volume *input*. Jenis operasi ini berguna untuk data dengan tingkat spasial tinggi atau lokalitas lain, seperti data gambar (Aggarwal 2018). *Convolution* adalah proses sederhana di mana kita menerapkan matriks (juga disebut kernel atau filter) ke gambar sehingga kita dapat memperkecil ukurannya, atau menambahkan beberapa lapisan pengisi agar ukurannya tetap sama. *Convolution* juga digunakan untuk mengekstraksi fitur spesifik dari suatu gambar, seperti bentuk, tepi, dan sebagainya (Himanshu Singh 2019).

Semakin banyak fitur yang diekstrak, semakin banyak ruang fitur yang diwakili jaringan, dan hasil pengenalan akhir akan lebih akurat. Tetapi jika jumlah kernel konvolusi terlalu besar, kompleksitas jaringan dan jumlah parameter akan meningkat, yang mengarah pada peningkatan kompleksitas perhitungan dan fenomena *overfitting*. Jadi jumlah kernel konvolusi harus ditentukan sesuai dengan ukuran dataset gambar tertentu (Gong et al. 2019).

Secara teknis, *convolutional network* adalah arsitektur yang bisa di *training* dan terdiri dari beberapa tahap. *Input* dan *output* dari masing-masing tahap adalah beberapa *array* yang disebut *feature map* atau peta fitur. *Output* dari masing-masing tahap adalah *feature map* hasil pengolahan dari semua lokasi pada *input*. Masing-masing tahap terdiri dari tiga *layer* yaitu *convolution layer*, *activation layer* dan *pooling layer* (Dewi 2018).

### 2.9.1 Convolution Layer

Cara kerja *convolutional neural network* (*CNN*) dimulai dengan *convolution layer*, tempat kami menerapkan filter (juga dikenal sebagai kernel) ke gambar *input*. Kernel ini melangkah di atas gambar, blok demi blok, di mana setiap blok adalah kumpulan sel piksel. Selama proses ini, kami melakukan perkalian matriks, yang

menghasilkan gambar dengan resolusi lebih rendah. Dalam lapisan *subsampling* (juga disebut lapisan *downsampling*) kami menemukan nilai piksel rata-rata (*average pooling*) atau nilai piksel maksimum (*max pooling*), dan mendapatkan gambar dengan resolusi lebih rendah (Himanshu Singh 2019).

Dalam *convolutional layer*, parameter yang dipelajari adalah filter itu sendiri. Misalnya, jika Anda memiliki 32 filter, masing-masing dimensi 5x5, Anda akan mendapatkan  $32 \times 5 \times 5 = 832$  parameter yang dapat dipelajari, karena untuk setiap filter ada juga istilah bias yang perlu Anda tambahkan (Michelluci and Moolayil 2019). Tujuan dilakukannya konvolusi pada data citra adalah untuk mengekstraksi fitur dari citra *input* (Dewi 2018). Secara umum operasi konvolusi dapat ditulis dengan rumus berikut :

$$(l \cdot k)_{(i,j)} = \sum_m \sum_n (l)_{(m,n)} (k)_{(i+m,j+n)} \quad (2.4)$$

Persamaan 2.10 merupakan perhitungan saat proses konvolusi dengan l merupakan *inputan* sedangkan k merupakan kernel (*filter*). i dan j merupakan representasi piksel dari sebuah citra. Proses pada konvolusi dapat dikatakan sebagai perkalian matriks antara piksel dalam citra *input* dengan piksel kernel (*filter*) yang mana *outputnya* dapat dihitung dengan proses ‘dot’.

### 2.9.2 Stride

*Filter* geser terjadi satu piksel pada satu waktu, tetapi tidak selalu demikian. Kami dapat menggeser *filter* beberapa posisi. Parameter *convolutional layer* ini disebut *stride*. Biasanya, langkahnya sama di semua dimensi *input* (Vasilev 2019). Semakin kecil *stride* yang digunakan, maka semakin detail informasi yang didapatkan dari sebuah *input*, namun membutuhkan komputasi lebih jika dibandingkan dengan *stride* yang besar (Dewi 2018).

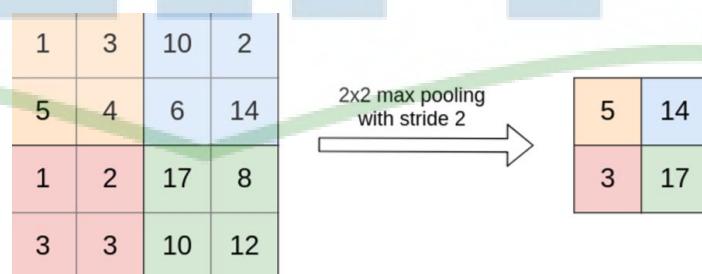
### 2.9.3 Padding

Untuk menghasilkan *output* dengan dimensi yang sama dengan *input*, dapat menggunakan *padding*. Dalam praktiknya, sering diinginkan untuk mengontrol ukuran *output*. Kita dapat menyelesaikan ini dengan mengisi tepi irisan input dengan baris dan kolom nol sebelum operasi konvolusi (Vasilev 2019).

### 2.9.4 Pooling Layer

*Pooling layer* membagi irisan input ke dalam kisi, di mana setiap sel kisi mewakili bidang *receptif* dari sejumlah *neuron* (seperti halnya *convolution layer*). Kemudian, operasi pengumpulan diterapkan pada setiap sel dari *grid*. Ada berbagai jenis *pooling layer*. *Pooling layer* tidak mengubah kedalaman volume, karena operasi *pooling* dilakukan secara independen pada setiap slice (Vasilev 2019). *Pooling layer* tidak memiliki parameter yang dapat dipelajari, dan seperti yang disebutkan, inilah alasan biasanya dikaitkan dengan *convolution layer*. Di lapisan ini (operasi), tidak ada bobot yang bisa dipelajari (Michelluci and Moolayil 2019).

Terdapat dua macam *pooling* yang sering dipakai yaitu *average pooling* dan *max pooling*. Dalam *average pooling*, nilai yang diambil adalah nilai rata-rata, sementara pada *max pooling*, nilai yang diambil adalah nilai maksimal (Dewi 2018).



**Gambar 2.11. Max Pooling**  
Sumber : (Vasilev 2019)

Pada gambar 2.11 menunjukkan proses *max pooling* untuk citra berukuran 4x4 dengan *pooling mask* berukuran 2x2. Tujuan dari

*pooling* adalah untuk mendapatkan matriks dengan dimensi yang lebih kecil dibandingkan dengan matriks awal sehingga mempercepat komputasi karena parameter yang diproses semakin dikit dan mengawasi *overfitting*. Proses konvolusi dan *pooling* dilakukan beberapa kali sehingga didapatkan *feature map* (peta fitur) dengan ukuran yang dikehendaki. Peta fitur tersebut akan menjadi *input* bagi *fully connected neural network* (Dewi 2018).

#### 2.9.5 Activation Function CNN

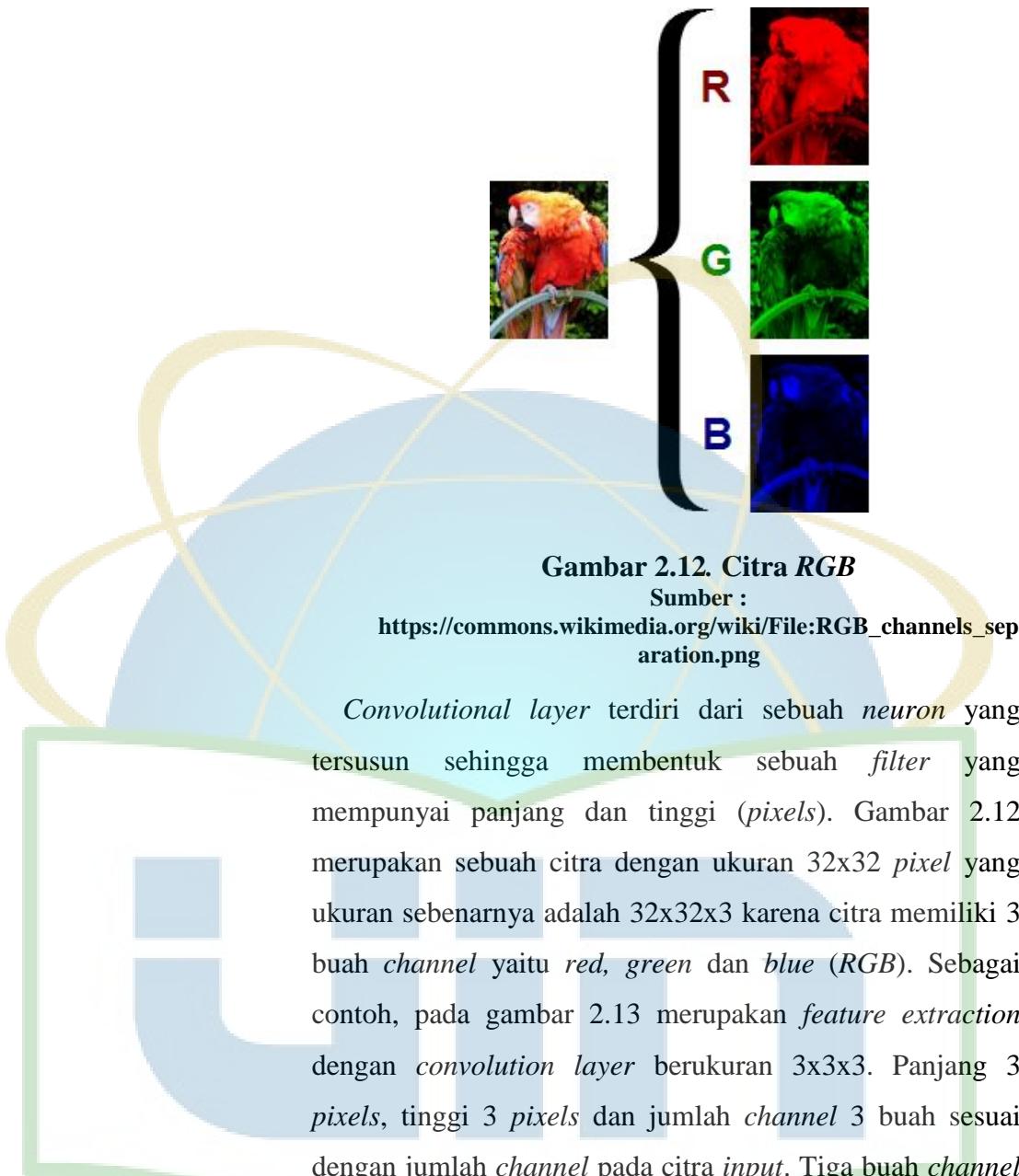
Sama seperti fungsi aktifasi pada *artificial neural network*, fungsi aktivasi pada *convolutional neural network*, yaitu untuk mengubah fungsi *linear* menjadi *non-linear*. Disetiap data *input* pada *layer* dilakukan operasi *linear* dengan nilai bobot yang ada, kemudian hasil komputasi akan ditransformasi menggunakan operasi *non linear* yang disebut sebagai fungsi aktivasi (Sofia 2018). Pada arsitektur CNN, fungsi aktivasi terletak pada perhitungan akhir keluaran feature map atau sesudah proses perhitungan konvolusi atau pooling untuk menghasilkan suatu pola fitur. Beberapa macam fungsi aktivasi yang sering digunakan dalam penelitian antara lain fungsi sigmoid, tanh, Rectified Linear Unit (ReLU), Leaky ReLU (LReLU) dan Parametric ReLU (Dewi 2018).

#### 2.9.6 Arsitektur Convolutional Neural Network

Secara garis besar, arsitektur pada *convolutional neural network* terbagi menjadi 2, yaitu :

##### a. Feature Extraction Layer

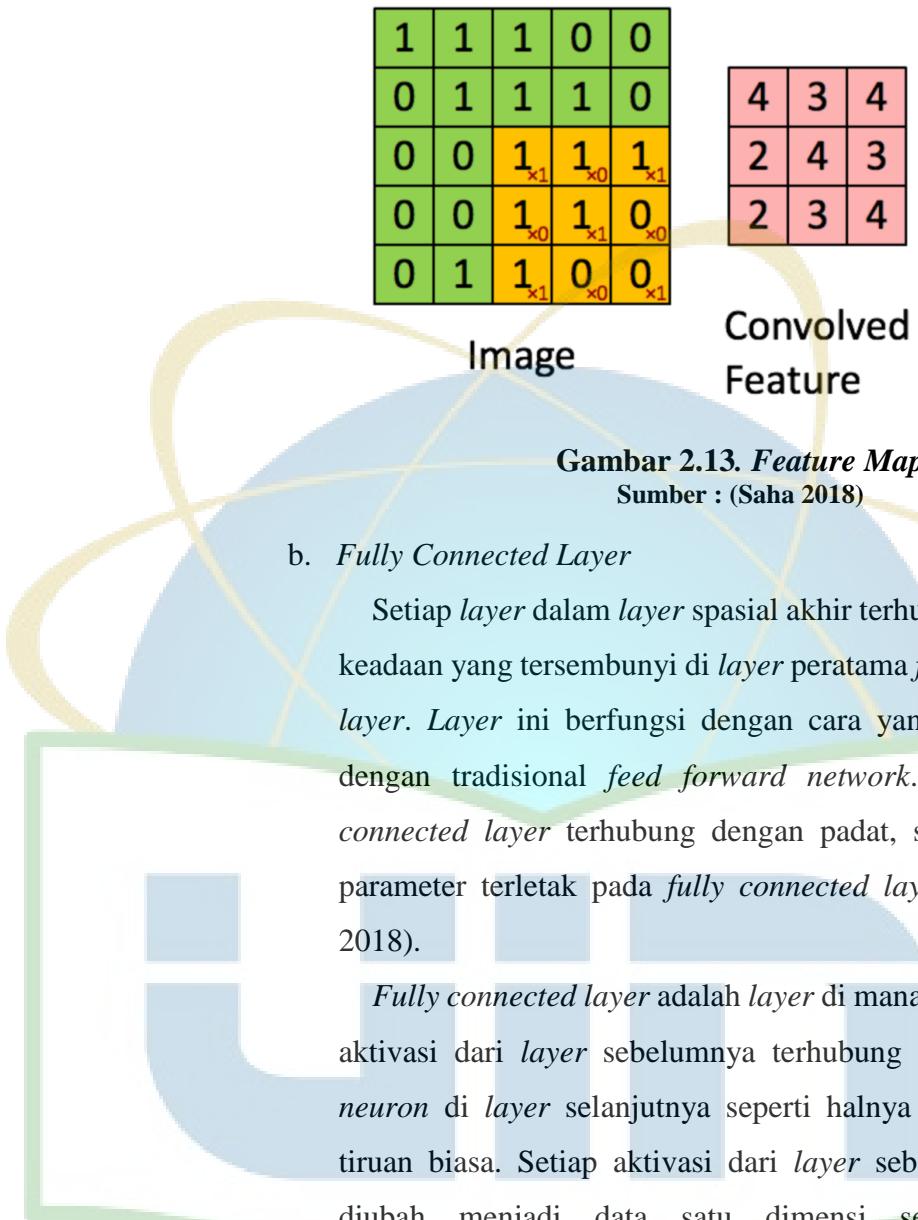
Pada arsitektur ini, dilakukan proses untuk merubah sebuah citra menjadi angka-angka yang mana angka-angka tersebut merepresntasikan citra itu sendiri. Proses ini disebut proses *encoding*.



**Gambar 2.12. Citra RGB**

Sumber :  
[https://commons.wikimedia.org/wiki/File:RGB\\_channels\\_separation.png](https://commons.wikimedia.org/wiki/File:RGB_channels_separation.png)

*Convolutional layer* terdiri dari sebuah *neuron* yang tersusun sehingga membentuk sebuah *filter* yang mempunyai panjang dan tinggi (*pixels*). Gambar 2.12 merupakan sebuah citra dengan ukuran  $32 \times 32$  *pixel* yang ukuran sebenarnya adalah  $32 \times 32 \times 3$  karena citra memiliki 3 buah *channel* yaitu *red*, *green* dan *blue* (*RGB*). Sebagai contoh, pada gambar 2.13 merupakan *feature extraction* dengan *convolution layer* berukuran  $3 \times 3 \times 3$ . Panjang 3 *pixels*, tinggi 3 *pixels* dan jumlah *channel* 3 buah sesuai dengan jumlah *channel* pada citra *input*. Tiga buah *channel* tersebut melakukan proses pergeseran pada seluruh bagian citra. Setiap pergeseran akan dilakukan operasi “dot” antara *input* dan nilai dari *filter* tersebut sehingga menghasilkan sebuah *output* atau biasa disebut dengan *activation map* atau *feature map*.



**Gambar 2.13. Feature Map**  
Sumber : (Saha 2018)

#### b. Fully Connected Layer

Setiap *layer* dalam *layer* spasial akhir terhubung ke setiap keadaan yang tersembunyi di *layer* peratama *fully connected layer*. *Layer* ini berfungsi dengan cara yang persis sama dengan tradisional *feed forward network*. Karena *fully connected layer* terhubung dengan padat, sebagian besar parameter terletak pada *fully connected layer*. (Aggarwal 2018).

*Fully connected layer* adalah *layer* di mana semua neuron aktivasi dari *layer* sebelumnya terhubung semua dengan *neuron* di *layer* selanjutnya seperti halnya jaringan saraf tiruan biasa. Setiap aktivasi dari *layer* sebelumnya perlu diubah menjadi data satu dimensi sebelum dapat dihubungkan ke semua neuron di lapisan (Dewi 2018).

Salah satu alternatif untuk menggunakan *fully connected layer* adalah dengan menggunakan pengumpulan rata-rata di seluruh area spasial dari set terakhir peta aktivasi untuk menciptakan nilai tunggal. Oleh karena itu, jumlah fitur yang dibuat dalam lapisan spasial akhir akan persis sama dengan jumlah filter (Aggarwal 2018).

## 2.11 Faster R-CNN

Masalah utama dengan pendekatan naif adalah bahwa Anda perlu menguji sejumlah besar jendela untuk dapat menemukan kotak terikat terbaik yang cocok. Mencari setiap lokasi yang mungkin tidak dapat dilakukan secara komputasi, karena sedang menguji semua rasio aspek dan ukuran jendela yang mungkin. Jadi Girshick et al.<sup>4</sup> mengusulkan metode di mana mereka menggunakan algoritma yang disebut *selective search* untuk pertama mengusulkan 2000 *regions* dari gambar (*region proposals*) dan kemudian, alih-alih mengklasifikasikan sejumlah besar wilayah, mereka mengklasifikasikan hanya 2.000 *regions* tersebut (Michelluci and Moolayil 2019).

*R-CNN* dan *fast R-CNN* keduanya menggunakan *selective search* untuk mengusulkan *region*, dan karena itu relatif lambat. Bahkan *fast R-CNN* membutuhkan sekitar dua detik untuk setiap gambar, membuat variasi ini tidak cocok untuk deteksi objek waktu nyata. Oleh karena itu, dikembangkan sebuah ide tentang sebuah model yang performanya lebih baik dari *fast R-CNN*, yaitu *faster R-CNN*.

Penggunaan *region proposal method* seperti *Selective Search* masih menjadi hambatan dalam proses deteksi objek menggunakan *R-CNN*, karena metode seperti ini membutuhkan waktu yang relatif lama untuk menghasilkan *regions*. Untuk mengatasi masalah tersebut, dikenalkan model *faster R-CNN* dengan metode *Region Proposal Network (RPN)*. *RPN* adalah sebuah *neural network* yang menggantikan peran *Selective Search* untuk mengajukan *region* (bagian-bagian mana dari sebuah gambar yang perlu “dilihat” lebih jauh). *RPN* menghasilkan beberapa *bounding box*, setiap *box* memiliki 2 skor probabilitas apakah pada lokasi tersebut terdapat objek atau tidak (Dharmadi 2018).

## 2.12 Tensorflow

*Tensorflow* merupakan 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* menggabungkan aljabar komputasi teknik pengoptimalan

kompilasi, mempermudah penghitungan banyak ekspresi *matematis* dimana masalahnya adalah waktu yang dibutuhkan untuk melakukan perhitungan (Dewi 2018). Tidak perlu secara eksplisit membutuhkan penggunaan *GPU*, alih-alih *TensorFlow* akan secara otomatis mencoba menggunakanannya jika Anda memilikinya. Jika Anda memiliki lebih dari satu *GPU*, Anda harus menetapkan operasi untuk setiap *GPU* secara *eksplisit*, atau hanya yang pertama yang akan digunakan (Vasilev 2019).

*Tensorflow library* telah jauh dari penampilan pertamanya. Terutama di tahun lalu, semakin banyak fitur yang tersedia yang dapat membuat kehidupan peneliti jauh lebih mudah. Hal-hal seperti eksekusi yang bersemangat dan *Keras* memungkinkan para ilmuwan untuk menguji dan bereksperimen lebih cepat dan men-debug model dengan cara yang tidak mungkin dilakukan sebelumnya (Michelluci and Moolayil 2019).

Menurut (Dewi 2018), beberapa fitur utama dari *tensorflow* diantaranya :

- a. Mendefinisikan, mengoptimalkan, dan menghitung secara efisien ekspresi *matematis* yang melibatkan *array multidimension (tensors)*
- b. Pemrograman pendukung jaringan syaraf dalam dan teknik pembelajaran mesin
- c. 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 khususnya lagi, *Tensorflow* akan mengetahui bagian perhitungan yang harus dipindahkan ke *GPU*.
- d. Skalabilitas komputasi yang tinggi di seluruh mesin dan kumpulan data yang besar.

### 2.13 *Centroid Tracking*

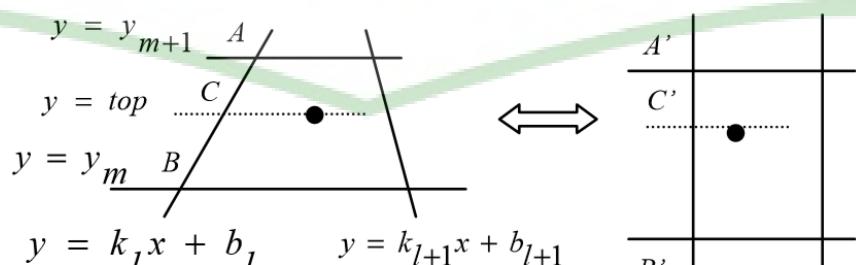
Algoritma *centroid tracking* mengasumsikan bahwa terdapat satu set *bounding box* atau kotak pembatas (x, y) yang akan dilewati dan

mengoordinasikan setiap objek yang terdeteksi disetiap frame tunggal. Setelah memiliki *bounding box*, langkah selanjutnya adalah menghitung *centroid* (titik pusat) untuk menetapkan *ID* suatu *boundin box* yang telah terdeteksi. Pada tiap frame selanjutnya dilakukan penghitungan kembali pada setiap *bounding box* untuk mendapatkan nilai *centroid*, namun berbeda pada pemberian *ID*, pada frame-frame selanjutnya, perlu ada proses pengaitan antara centroid pada frame saat ini dengan frame sebelumnya untuk diketahui apakah *bounding box* pada frame saat ini mendapatkan *ID* baru atau tidak. Permasalahan ini dapat diselesaikan menggunakan perhitungan jarak *Euclidean* pada objek tiap frame, yaitu menghitung jarak terdekat yang akan ditentukan sebagai objek yang sama pada tiap frame. Adapun rumus jarak *Euclidean* adalah sebagai berikut:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (2.5)$$

#### 2.14 Parallel Line Model

Dalam geometri, *parallel line* merupakan garis dalam bidang yang tidak bertemu. Gagasan utama dari model ini adalah perbandingan, dalam penelitian ini kasusnya adalah membandingkan jarak dan kecepatan kendaraan pada piksel dengan jarak dan kecepatan pada dunia nyata. Oleh karena itu penting untuk mengetahui kondisi sebenarnya keadaan lintasan yang akan dilewati oleh kendaraan dalam video. Ilustrasi penggunaan model *parallel line* dapat dilihat pada gambar berikut :



(a) Positions in the image plane

(b) Positions in the real world

### Gambar 2.14. Ilustrasi model Parallel Line

Sumber : (Zhiwei, Yuanyuan, and Xueyi 2007)

Rumus perbandingan untuk mendapatkan kecepatan pada dunia nyata berdasarkan ilustrasi diatas dapat dilihat pada formula 2.5

$$\frac{|A'C'|}{|A'B'|} = \frac{|AC|}{|AB|} = \frac{y_{m+1}-top}{y_{m+1}-y_m} \quad (2.6)$$

Dimana  $y_{m+1}$  dan top adalah jarak antara garis A dan kendaraan sedangkan  $y_{m+1}$  dan  $y_m$  adalah jarak antara garis A dan garis B yang telah ditentukan. Jarak antara garis yang dibuat harus diketahui terlebih dahulu jarak sebenarnya dalam dunia nyata.

### 2.15 Normalized Root Mean Square Error

*Normalized Root mean square error* merupakan normalisasi dari metode *root mean square error*. Jadi, rumus pada *root mean square error* dinormalisasi dengan cara dikali dengan hasil dari deteksi kendaraan. *RMSE* merupakan salah satu metode untuk mengevaluasi teknik prediksi yang digunakan untuk mengukur tingkat akuisisi hasil dari prediksi suatu model. adapun formula dari metode *root mean square error* sebagai berikut:

$$RSME = \sqrt{\sum_{i=1}^n \frac{(\hat{y}_i - y_i)^2}{n}} \quad (2.7)$$

Yang mana n merupakan jumlah seluruh data yang diprediksi,  $\hat{y}_i$  merupakan nilai dari kecepatan sebenarnya dan  $y_i$  merupakan nilai kecepatan dari hasil prediksi model. Sedangkan untuk normalisasi *RMSE*, formula *RMSE* dibagi dengan nilai hasil pengurangan y tertinggi dikurang y terendah. Adapun formula *normalized root mean square error* adalah sebagai berikut:

$$NRSME = \frac{\sqrt{\sum_{i=1}^n \frac{(\hat{y}_i - y_i)^2}{n}}}{y_{max} - y_{min}} \quad (2.8)$$

## 2.16 Penelitian Terdahulu

**Tabel 2 1. Tabel Literatur Sejenis**

Judul Penelitian	Penulis	Metode Penelitian	Hasil
<i>Speed Estimation and Abnormality Detection from Surveillance Cameras</i>	Panagiotis Giannakeris, dkk (2018)	<i>Faster R-CNN, KFC and calibration algorithm</i>	Penelitian ini dilatar belakangi oleh tren industri untuk meningkatkan hasil <i>traffic intelligence</i> . Menggunakan algoritma <i>detection</i> menggunakan <i>Faster R-CNN</i> dan <i>tracking</i> menggunakan algoritma <i>KFC</i> untuk mendapatkan lokasi kendaraan dalam video. Kemudian, <i>calibration algorithm</i> untuk menghitung kecepatan kendaraan. <i>Detection</i> dan <i>tracking</i> mendapatkan akurasi sebesar 89% untuk kedua pengaturan deteksi. Sedangkan untuk kecepatan mendapatkan skor 27,30 dengan skor <i>RMSE</i>
<i>Vehicle Tracking and Speed Estimation from Traffic Videos</i>	Shuai Hua, Manika Kapoor, dan David C. Anastasiu (2018)	<i>Model 3D Deformable,</i>	Latar belakang dari penelitian ini adalah menciptakan solusi untuk masalah kecelakaan lalu lintas dan dapat membantu meningkatkan

<i>Traffic Speed Estimation from Surveillance Video Data</i>	Tingting Huang (2018)	Metode <i>detect-then-track</i>  <i>Faster R-CNN</i> , <i>Histogram Comparison</i> , dan Metode <i>Warping</i>	manajemen lalu lintas yang lebih baik. Menggunakan metode <i>transfer learning</i> dengan model <i>3D Deformable</i> untuk deteksi objek. Estimasi kecepatan sangat bergantung pada hasil deteksi objek karena <i>tracking</i> objek menggunakan metode <i>detect-then-track</i> . Menghasilkan skor <i>DR</i> 1,0 dan skor <i>RMSE</i> 12,1094 serta memperoleh skor <i>S1</i> 0,6547
--	--------------------------	--	--

			gabungan memperoleh skor <i>RSME</i> 8,6089 dan <i>DR</i> 0,8148
<i>A Semi-Automatic 2D solution for Vehicle Speed Estimation from Monocular Videos</i>	Amit Kumar, dkk (2018)	<i>Mask-RCNN</i> , Metode <i>SORT and DeepSORT</i> dan <i>Kalman filter</i>	Latar belakang dari penelitian ini adalah menyajikan pendekatan baru untuk prediksi kecepatan dari video monokuler. Penelitian ini mencapai tingkat deteksi 1,0 pada deteksi dan pelacakan kendaraan, dan <i>Root Mean Square Error</i> 9,54 (mph) untuk tugas estimasi kecepatan kendaraan dalam video lalu lintas yang tidak dibatasi.
<i>Traffic Flow Analysis with Multiple Adaptive Vehicle Detectors and Velocity Estimation with Landmark-based Scanlines</i>	Minh-Triet Tran, dkk (2018)	<i>Faster R-CNN</i>	Penelitian ini dilatar belakangi dengan pencarian metode baru untuk meningkatkan akurasi prediksi kecepatan. Penelitian ini mencapai tingkat deteksi sempurna 100% dengan menggunakan algoritma <i>Faster R-CNN</i> , perbedaan kecepatan rata-rata 6,9762 mph di jalan raya, dan 8,9144 mph di jalan bebas hambatan dan jalan perkotaan.

Penelitian ini dibagi menjadi 3 langkah, yaitu deteksi objek, *tracking* objek dan perhitungan kecepatan yang dikonversi dari piksel ke dunia nyata. Penelitian ini menggunakan model *Faster R-CNN* untuk mendeteksi objek dalam sebuah video, *tracking* objek menggunakan *algoritma centroid tracking* berdasarkan jarak *Euclidean* dan menggunakan model *parallel line* untuk mengkonversi kecepatan dari piksel ke dunia nyata.



## BAB III

### METODE PENELITIAN

#### 3.1 Metode Pengumpulan Data

Penulis mengguakan 2 metode untuk pengumpulan data, yaitu metode studi pustaka dan metode observasi yang akan dijelaskan sebagai berikut :

##### 3.1.1 Studi Pustaka

Penulisan ini diawali dengan melakukan studi pustaka dengan mengumpulkan beberapa referensi terkait dengan penelitian ini yang dapat dijadikan acuan. Beberapa referensi yang digunakan antara lain buku-buku, skripsi, jurnal-jurnal maupun website. Dari sumber-sumber tersebut, penulis mempelajari berbagai macam informasi definisi, data, teori serta implementasi yang diperlukan penulis untuk membantu penyelesaian pada bab pendahuluan, landasan teori dan metode penelitian.

##### 3.1.2 Observasi

Penulis melakukan observasi untuk mengumpulkan data-data yang diperlukan untuk menyelesaikan sistem yang penulis buat. Data-data tersebut berupa dataset yang diperlukan dalam proses *training*, yaitu berupa video yang diambil melalui kamera *Samsung* yang penulis asumsikan sebagai kamera pengawas lalu lintas. Penulis juga mengambil data berupa variable-variabel yang dibutuhkan dalam perhitungan rumus kecepatan untuk mengkonversi kecepatan dari piksel ke dunia nyata seperti panjang lintasan, jarak antara garis yang telah penulis tetapkan dan sebagainya. Adapun observasi penulis lakukan di ITC permata Hijau, Jl. Arteri Permata Hijau, Grogol Utara Kebayoran Lama Jakarta Selatan sekitar pukul 16.00. Pada penelitian ini tidak ada acuan bagi penulis mengenai waktu pengambilan *dataset*. Penulis memilih lokasi tersebut karena lintasan pada lokasi datar, artinya tidak

menanjak ataupun menurun. Di lokasi juga terdapat jembatan penyebrangan orang (JPO). Di JPO tersebut tempat penulis merekam video dengan harapan hasil rekaman tersebut menyerupai hasil dari rekaman video pengawas lalu lintas yang sesungguhnya..

### 3.1.3 Wawancara

Penulis menggunakan metode wawancara untuk mengumpulkan data validasi dalam mengestimasi kecepatan kendaraan. Penulis melakukan wawancara kepada seorang ahli atau seseorang yang sudah terbiasa bekerja pada bidang lalu lintas. Penulis menanyakan pendapat dari orang tersebut mengenai kecepatan kendaraan yang nantinya, jawaban dari pertanyaan tersebut akan menjadi data validasi pada penelitian ini dalam mengestimasi kecepatan kendaraan dalam video pengawas lalu lintas.

## 3.2 Metode Pengembangan Sistem

Pada metode pengembangan sistem, penulis menggunakan model *prototyping* dengan asumsi bahwa penulis sendiri merupakan kliennya. Penulis tidak menggunakan semua tahapan yang ada pada model *prototyping*, tahapan pada penelitian ini telah disesuaikan dengan apa yang penulis butuhkan. Adapun tahap-tahap dalam pengembangan sistem dijelaskan sebagai berikut:

### 3.2.1 Analisis Kebutuhan

Pada tahap ini, penulis mengumpulkan kebutuhan secara lengkap kemudian menganalisis dan mendefinisikan kebutuhan yang harus dipenuhi oleh sistem yang akan penulis rancang. Analisis kebutuhan sistem sebagai bagian dari studi awal yang bertujuan mengidentifikasi masalah dan kebutuhan sistem. Kebutuhan sistem adalah spesifikasi mengenai hal-hal yang akan dilakukan sistem ketika diimplementasikan.

Analisis kebutuhan dilakukan dengan 2 metode, yaitu metode *document survey* dan metode observasi.

### 3.2.2 Desain *Prototyping*

Desain sistem menentukan bagaimana sistem akan memenuhi tujuan dari pengembangan sistem. Desain sistem terdiri dari aktivitas desain yang menghasilkan spesifikasi fungsional. Desain sistem dapat dikatakan sebagai desain *interface*, data dan proses dengan tujuan menghasilkan spesifikasi yang sesuai dengan yang ada pada tahap analisis kebutuhan, baik dari produk, metode, algortima, proses dan sebagainya.

Dalam tahap ini, penulis akan menghasilkan sebuah sistem secara keseluruhan dari tahap analisis yang telah dilakukan dan menentukan alur dari perngkat lunak hingga model dan algoritma yang penulis gunakan dalam merancang sistem. Pada tahap ini, penulis juga menentukan arsitektur informasi dan teknologi terapan seperti Bahasa pemrograman, perpustakaan kelas dan urutan program.

### 3.2.3 *Development*

Pada tahap ini seluruh desain diimplementasikan kedalam code program. Code-code program tersebut masih berupa modul-modul yang nantinya akan diintegrasikan menjadi sistem yang lengkap. Sistem lengkap berupa sebuah model untuk estimasi kendaraan yang melintas dari kamera pengawas lalu lintas. Pada tahap *development* terdiri dari beberapa langkah, adapun langkah-langkahnya adalah sebagai berikut:

- a *Annotasi dataset*
- b *Split dataset*
- c *Training*
- d *Tracking*
- e *Speed estimation*

### 3.2.4 *Testing*

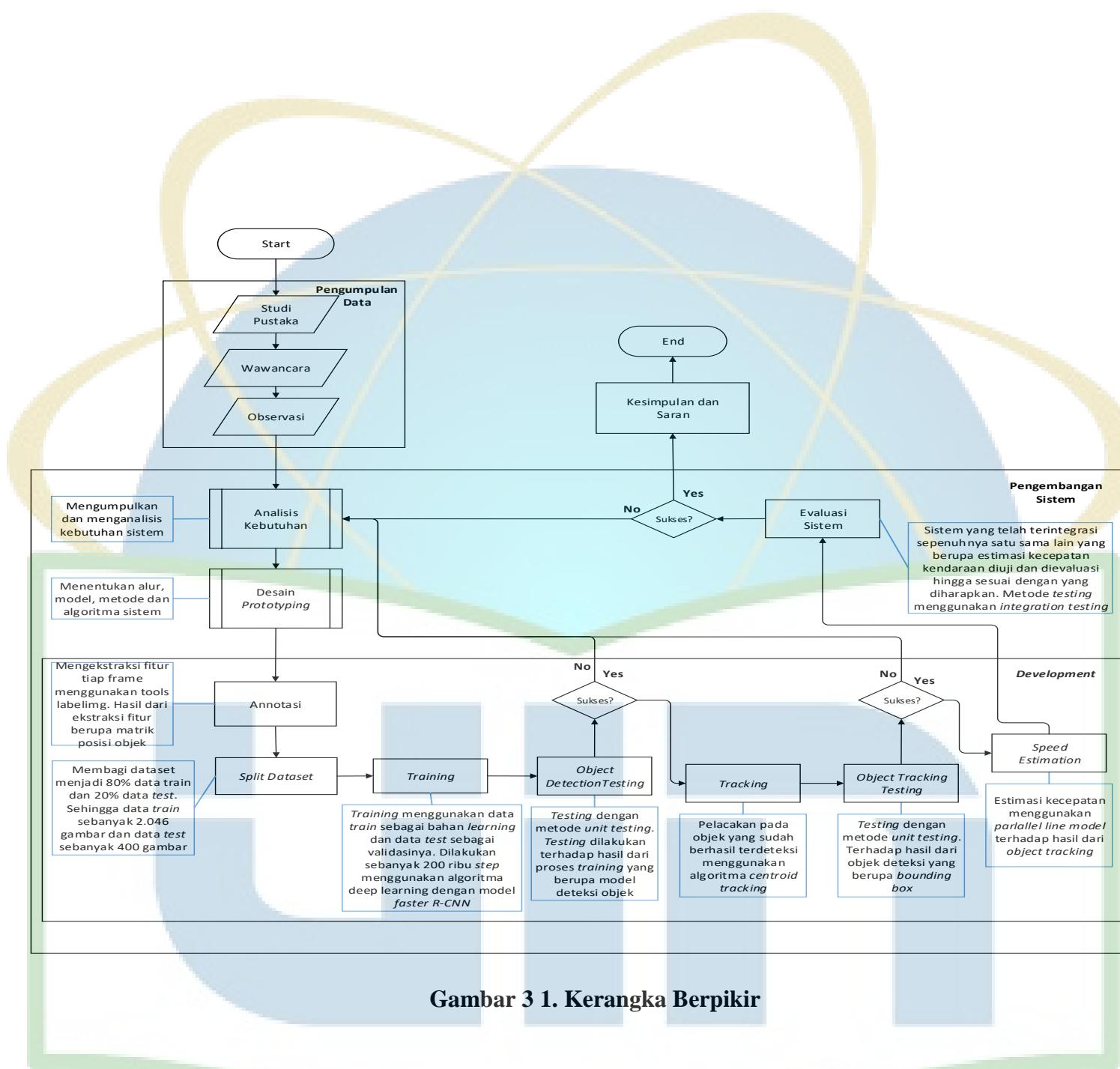
Ditahap ini, penulis menggabungkan modul-modul yang telah dibuat. Penulis juga melakukan pengujian untuk menganalisa apakah sistem yang telah dibuat sesuai dengan desainnya dan semua fungsi pada sistem terdapat error atau tidak. *Testing* dilakukan dengan metode *unit testing*, yaitu metode *testing* dengan proses pengujian individual, *class*, atau *komponen* sebelum mereka terintegrasi dengan sistem lain secara keseluruhan. Adapun *unit testing* terdiri dari *testing object detection* dan *testing object tracking*.

### 3.2.5 Evaluasi Sistem

Paket *software prototype* diuji, diimplementasikan dan dimodifikasi berulang-ulang hingga sistem sesuai dengan yang klien/penulis harapkan. Pengujian sistem bertujuan menemukan kesalahan-kesalahan yang terjadi pada sistem dan melakukan revisi sistem. tahap ini penting untuk memastikan bahwa sistem bebas dari kesalahan. Evaluasi sistem menggunakan metode *integration testing*.

### 3.3 Kerangka Berpikir

Dalam melakukan penelitian ini, penulis melakukakn tahapan-tahapan kegiatan yang dapat dilihat pada kerangka berpikir berpikir berikut ini :



Gambar 3 1. Kerangka Berpikir

## BAB IV

### IMPLEMENTASI

Pada bab ini akan dilakukan implementasi dan pembahasan dalam perancangan sistem sesuai dengan alur, metode, model dan algoritma yang digunakan pada penelitian ini dan telah dibahas pada bab sebelumnya.

#### 4.1 Analisis Kebutuhan

Metode *document survey* penulis lakukan untuk mengumpulkan data-data yang diperlukan untuk dipelajari dalam mengembangkan sistem. Penulis mempelajari berbagai macam teori yang berkaitan dengan cara penulis merancang sistem seperti mempelajari teori dan cara menggunakan *tensorflow*, *deep learning*, *Faster R-CNN*, algoritma *centroid tracking*, dan mempelajari model yang penulis gunakan untuk mengestimasi kecepatan, yaitu model *parallel line*. Penulis juga mempelajari langkah demi langkah dalam perancangan sistem seperti langkah demi langkah dalam mendeteksi objek, langkah demi langkah dalam menyelesaikan objek *tracking*, dan langkah demi langkah bagaimana sebuah sistem dapat mendeteksi kecepatan kendaraan melalui sebuah video, hingga akhirnya sistem benar-benar mampu mengestimasi kecepatan kendaraan melalui sebuah video.

Sedangkan metode observasi penulis lakukan untuk mengumpulkan *dataset*. *dataset* yang penulis kumpulkan sebanyak 2.446 gambar yang diproyeksikan dari sebuah video berdurasi 2 menit yang diambil dari sebuah kamera *Samsung 32 MP* milik penulis sendiri di lokasi ITC permata Hijau, Jl. Arteri Permata Hijau, Grogol Utara Kebayoran Lama Jakarta Selatan. *Dataset* ini penulis asumsikan sebagai kamera pengawas lalu lintas. Di lokasi tersebut, penulis juga menghitung panjang *landmark* dan panjang jarak antara *landmark* sebagai sebuah variabel yang harus penulis ketahui untuk menghitung estimasi kecepatan kendaraan dalam sebuah video. Panjang *landmark* di lokasi tersebut adalah 3 meter dengan jarak antara *landmark* 5 meter. Berikut adalah salah satu

contoh gambar *dataset* dan ilustrasi panjang *landmark* dan panjang jarak antar *landmark* :

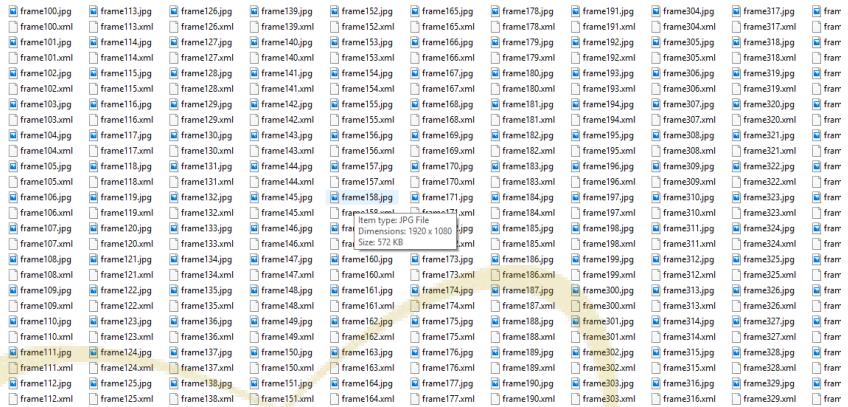


**Gambar 4.1. Contoh Gambar *Dataset***



**Gambar 4.2. Panjang *Landmark* dan Jarak Antara *Landmark***

Berikut merupakan sampel dataset dari video yang telah dipecah :



**Gambar 4.3. Hasil Split Dataset Video**

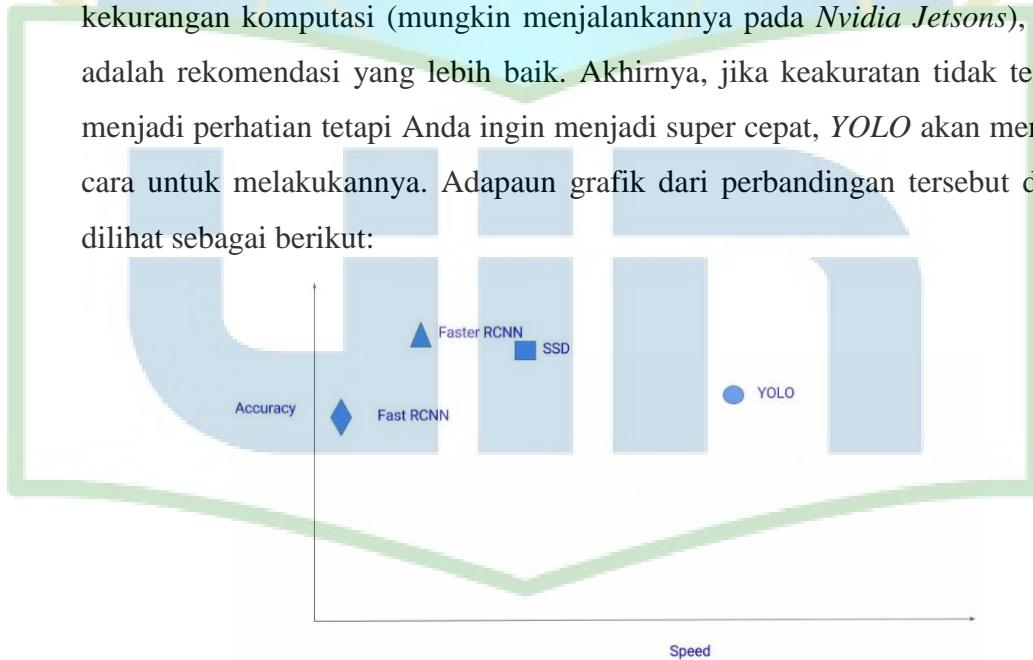
Penulis menggunakan metode wawancara untuk mengumpulkan data validasi dalam mengestimasi kecepatan kendaraan melalui video pengawas lalu lintas. Hasil dari wawancara dapat dilihat pada lampiran di halaman akhir pada laporan ini. Penulis mendatangi seseorang yang telah terbiasa bekerja pada transportasi dan lalu lintas untuk menanyakan kecepatan kendaraan melalui video yang menjadi data uji pada penelitian ini. Jawaban dari orang tersebut, akan penulis jadikan sebagai data validasi. Penulis mendatangi anggota SATLANTAS yang bernama Aiptu Tony yang berlokasi di daerah Relasi, Kebon Jeruk, Jakarta Barat. Hasil dari wawancara data validasi juga dapat dilihat pada halaman akhir laporan ini.

Karena membutuhkan komputasi yang berat dalam melakukan proses *training*, maka penulis menggunakan *platform* yang telah disediakan oleh Google, yaitu *Google Colaboratory* dengan pengaturan *runtime GPU* yang terhubung dengan *Google Drive* sebagai media penyimpanannya. Untuk menggunakan *Google Colaboratory*, penulis hanya membutuhkan koneksi yang stabil agar *runtime* saat menjalankan proses *training* tidak terputus. Penulis menggunakan *Google Drive* sebagai media penyimpanan *dataset*, karena untuk mengolah data dari *Google Colab*, data harus diupload terlebih dahulu kedalam *Google Drive*.

## 4.2 Desain *Prototyping*

Penulis menggunakan 3 tahapan dalam merancang sistem, yaitu tahap deteksi, tahap *tracking* dan tahap perhitungan estimasi kecepatan kendaraan. pada tahap deteksi, penulis menggunakan algoritma *deep learning* dengan model *Faster R-CNN*. Penulis menggunakan model *Faster R-CNN* karena model tersebut sesuai dengan masalah pada penelitian ini. model *Faster R-CNN* sangat cepat dalam masalah mendeteksi objek, dan memiliki tingkat akurasi yang paling baik. Hal tersebut sesuai pada penelitian ini yang mana akurasi dari estimasi kecepatan sangat dipengaruhi oleh akurasi pada proses-proses sebelumnya.

Terdapat beberapa model yang sangat *power full* dalam menyelesaikan masalah deteksi objek jika dilihat dari segi kecepatan membaca objek, dari segi komputasi, dan dari segi akurasi. Dikutip dari website (Sachan 2018), *Faster R-CNN* adalah pilihan jika Anda fanatik dengan angka akurasi. Namun, jika Anda kekurangan komputasi (mungkin menjalankannya pada *Nvidia Jetsons*), *SSD* adalah rekomendasi yang lebih baik. Akhirnya, jika keakuratan tidak terlalu menjadi perhatian tetapi Anda ingin menjadi super cepat, *YOLO* akan menjadi cara untuk melakukannya. Adapaun grafik dari perbandingan tersebut dapat dilihat sebagai berikut:

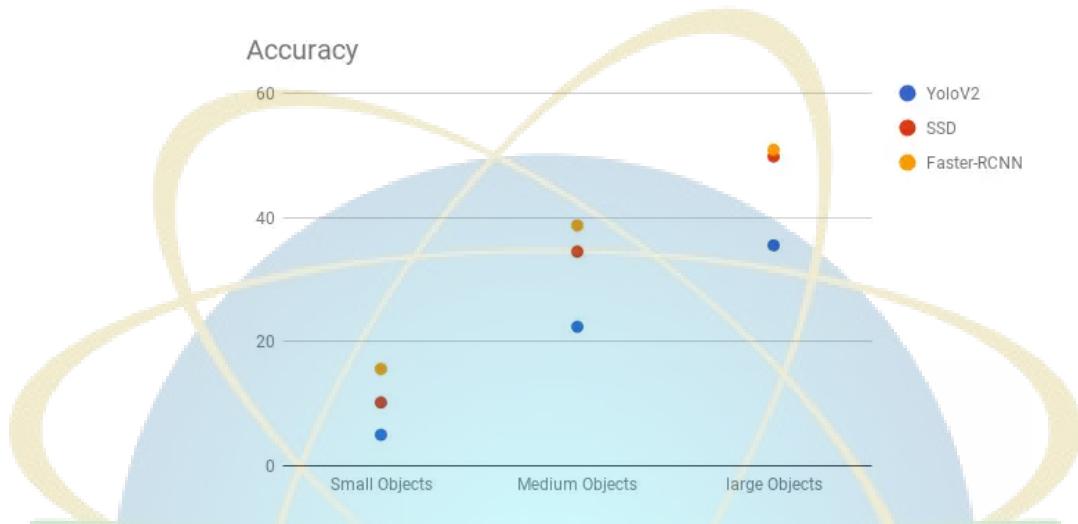


**Gambar 4.4. Perbandingan Model Objek Deteksi**

Sumber : (Sachan 2018)

Dari gambar tersebut, dapat dilihat bahwa model *Faster R-CNN* mengungguli semua model dalam hal akurasi. Model *YOLO* mengungguli

semua model dalam hal kecepatan dalam mendeteksi objek. Jika diperhatikan, model *SSD* memiliki akurasi yang hampir sama dengan model *Faster R-CNN* dan memiliki kecepatan mendeteksi yang lebih baik. Namun, model *SSD* sangat buruk dari sisi deteksi akurasi pada objek yang kecil. Hal tersebut dapat dilihat lebih teliti pada gambar berikut:



**Gambar 4.5. Perbandingan Model Objek Deteksi**

Sumber : (Sachan 2018)

Pada gambar 4.4, dapat disimpulkan bahwa model deteksi objek *SSD* akurasinya menyamai model *Faster R-CNN* hanya dalam hal deteksi yang objeknya besar. Hal tersebut membuat model *SSD* tidak sesuai jika digunakan dalam penelitian ini. *Faster R-CNN* masih memungkinkan model-model deteksi objek yang lainnya dalam hal akurasi, baik pada objek yang besar, objek yang sedang maupun objek yang kecil. Dari penjelasan tersebut, dapat disimpulkan bahwa model deteksi objek *Faster R-CNN* yang paling sesuai digunakan dalam penelitian ini.

Pada tahap *tracking*, penulis menggunakan algoritma *centroid tracking*. Alasan penulis menggunakan algoritma tersebut karena algoritma *centroid tracking* merupakan algoritma yang sederhana namun masih sangat akurat untuk digunakan pada penelitian ini. Untuk menggunakan metode ini, diperlukan objek deteksi yang sangat akurat, hal tersebut sudah dimiliki dalam penelitian ini. permasalahan lainnya dalam menggunakan algoritma ini adalah

jika kotak pembatas (*bounding box*) pada tiap frame berdekatan atau tumpeng tindih, maka algoritma ini bisa salah memberikan *ID* yang seharusnya pada objek. Pada penelitian ini, sedikit kemungkinan menemukan objek yang tumpeng tindih karena objek yang berupa kendaraan akan terus berjalan maju sampai kendaraan tersebut hilang dari frame. Dengan demikian, maka algoritma ini masih sangat sesuai jika digunakan pada penelitian ini.

Untuk menghitung estimasi kecepatan kendaraan, penulis menggunakan model *parallel line* untuk mengkonversi kecepatan dari piksel ke dunia nyata. Model *parallel line* juga merupakan model yang sederhana, konsep perhitungan hanya menggunakan perbandingan antara piksel dan dunia nyata. Untuk menggunakan model ini, penulis harus mengetahui jarak lintasan pada dunia nyata. Dari jarak lintasan yang telah diketahui tersebut, penulis akan menentukan acuan yang sama untuk mendapatkan jarak dalam piksel. Pada penelitian ini, jarak lintasan pada dunia nyata menggunakan acuan *landmark* atau marka jalan. Untuk mendapatkan jarak tersebut, penulis dapat dari aturan KM 60 tahun 1993 yang mengatur ukuran dari marka jalan Indonesia. Adapun aturannya secara rinci sebagai berikut:

1. Lebar garis utuh maupun putus-putus pada marka membujur sekurang-kurangnya 0,10 meter.
2. Panjang masing-masing garis pada garis putus-putus harus sama, berdasarkan kecepatan rencana :
  - a. kurang dari 60 km per jam, panjang garis putus-putus 3,0 meter.
  - b. 60 km per jam atau lebih, panjang garis putus-putus 5,0 meter.
3. Panjang celah diantara garis putus-putus sebagaimana dimaksud dalam ayat (1) harus sama, berdasarkan kecepatan rencana :
  - a. kurang dari 60 km per jam, panjang celah garis putus-putus 5,0 meter.
  - b. 60 km per jam atau lebih, panjang celah garis putus-putus 8,0 meter.

Untuk lebih meyakinkan penulis mengenai jarak lintasan pada dunia nyata, penulis melakukan observasi ke lokasi tempat pengambilan *dataset* yang telah disebutkan sebelumnya untuk menghitung secara langsung jarak lintasan

dengan *landmark* sebagai acuannya. Penulis mendapatkan jarak yang sama dengan yang telah diatur pada aturan KM 60 tahun 1993.

Penulis menggunakan code editor *Sublime Text* dengan Bahasa pemrograman *Python*. *Python* merupakan bahasa pemrograman yang banyak komunitasnya, sehingga menjadi lebih mudah dalam menyelesaikan masalah-masalah saat proses *coding*. Penulis juga menggunakan *anaconda* sebagai *packaging* bahasa *python* yang memudahkan penulis merancang sistem dengan *library-library* yang telah disediakan.

### 4.3 Development

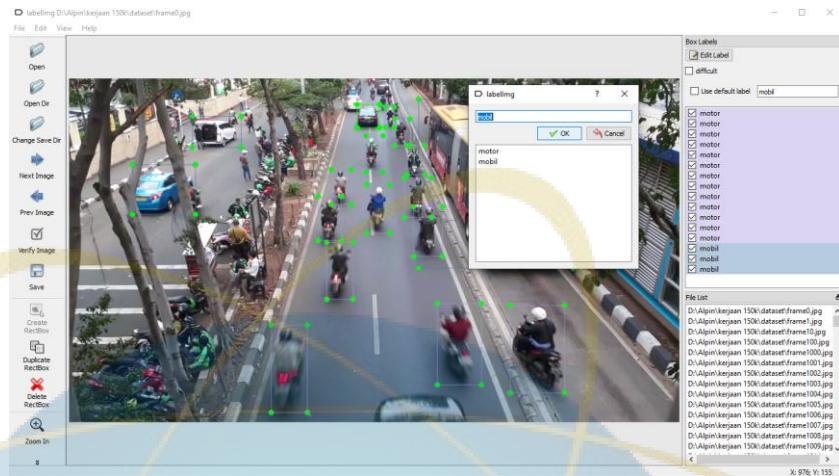
Pada tahap *development*, terdapat beberapa langkah hingga sistem mampu mengestimasi kecepatan kendaraan melalui sebuah video. adapun langkah-langkah yang penulis lakukan adalah sebagai berikut:

#### 4.3.1 Annotasi

Dalam tahap ini, penulis melakukan *annotasi* pada *dataset*. *Annotasi* dilakukan untuk mendapatkan ciri khusus yang merepresentasikan masing-masing objek yang terdapat dalam gambar. Ciri khusus tersebut nantinya akan dijadikan bahan *learning* dalam proses *training*. *Dataset* dibagi menjadi 2 kelas, yaitu kelas mobil dan kelas motor. *Annotasi* dilakukan untuk mendapatkan fitur-fitur yang dianggap penting pada dataset dalam mendeteksi objek. Fitur-fitur yang diambil pada proses *annotasi* diantaranya adalah nama folder, direktori penyimpanan *dataset*, tinggi layer, lebar layer, kedalaman layer, tinggi objek, lebar objek.

*Annotasi* dilakukan dengan menggunakan *tools labelimg*. Tahap awal dalam proses *annotasi* menggunakan *labelimg* adalah memasukkan data yang akan diannotasi. Selanjutnya tentukan objek yang nantinya akan dideteksi, pada penelitian ini objek adalah objek mobil dan motor. Terakhir beri tanda dengan memberi kotak pembatas (*bounding box*)

pada area sekitar objek. Berikut adalah contoh proses *annotasi* menggunakan tools labelimg :



**Gambar 4.6. Proses Annotasi**

Dari data *frame* 0 diatas, penulis menandai objek sebanyak 16 objek, objek dengan kelas motor sebanyak 13 objek dan objek dengan kelas mobil sebanyak 3 objek. Hasil dari *annotasi* tersebut berupa file XML yang merepresentasikan koordinat objek dan nama kelas dari objek tersebut. Hasil dari proses *annotasi* menggunakan *tools lebelimg* berupa file dengan ekstensi *.xml*. dalam file tersebut terdapat beberapa variabel yang merepresentasikan objek seperti yang telah disebutkan sebelumnya. Beberapa variable tersebut diantaranya lebar objek, tinggi, kedalaman, label dan koordinat objek tersebut. Hasil proses *annotasi* menggunakan tools labelimg lebih rinci lagi dapat dilihat pada gambar 4.4 :

```

<annotation>
  <folder>dataset</folder>
  <filename>frame0.jpg</filename>
  <path>D:\Alpin\kerjaan 150k\dataset\frame0.jpg</path>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>1920</width>
    <height>1080</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>motor</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>810</xmin>
      <ymin>515</ymin>
      <xmax>893</xmax>
      <ymax>694</ymax>
    </bndbox>
  </object>
</annotation>

```

**Gambar 4.7. Hasil Proses *Annotasi***

Berikut merupakan contoh proses *annotasi* pada tiap *frame* terhadap objek yang sama untuk mengetahui perbedaan koordinat objek :

**Tabel 4.1. Annotasi Terhadap Objek Yang Sama**

Frame	Gambar	Koordinat	Array
Frame 0		<xmin>933</xmin> <ymin>338</ymin> <xmax>1006</xmax> <ymax>480</ymax>	[[[ 87 95 125] [- 87 95 125] [ - 90 98 128] ... [ 51 53 47] [ 47 49 43] [ 44 45 41]] ... [[ 143 128 126] [ 143 128 126] [ 141 128 126] ... [ 57 55 54] [ 58 56 55] [ 58 56 55]] ...
Frame 1		<xmin>933</xmin> <ymin>338</ymin> <xmax>1003</xmax> <ymax>475</ymax>	[[ 178 163 160] [ 178 163 160] [ 178 163 160] ... [ 173 158 155] [ 173 158 155] [ 173 158 155]] ... [[ 157 138 133] [ 155 136 131] [ 155 136 131] ... [ 154 141 133] [ 154 141 133] [ 155 142 134]] ...

Frame 2		<pre> &lt;xmin&gt;935&lt;/xmin&gt; &lt;ymin&gt;330&lt;/ymin&gt; &lt;xmax&gt;1003&lt;/xmax&gt; &lt;ymax&gt;473&lt;/ymax&gt; </pre>	<pre> [[[178 163 160]   [[156 137 132]  [174 162 158]   [156 137 132]  [174 162 158]   [155 136 131]  ...  [175 158 155]   [156 141 138]  [175 158 155]   [156 141 138]  [174 157 154]] ... [157 142 139]]] </pre>
Frame 3		<pre> &lt;xmin&gt;935&lt;/xmin&gt; &lt;ymin&gt;327&lt;/ymin&gt; &lt;xmax&gt;1001&lt;/xmax&gt; &lt;ymax&gt;461&lt;/ymax&gt; </pre>	<pre> [[[175 163 159]   [[159 140 135]  [174 162 158]   [156 137 134]  [174 162 158]   [155 136 133]  ...  [176 156 155]   [153 140 132]  [176 156 155]   [154 141 133]  [176 156 155]] ... [156 143 135]]] </pre>
Frame 4		<pre> &lt;xmin&gt;933&lt;/xmin&gt; &lt;ymin&gt;319&lt;/ymin&gt; &lt;xmax&gt;1001&lt;/xmax&gt; &lt;ymax&gt;452&lt;/ymax&gt; </pre>	<pre> [[[175 163 159]   [[160 141 136]  [175 163 159]   [160 141 136]  [175 163 159]   [159 140 135]  ...  [175 155 154]   [158 145 137]  [175 155 154]   [158 145 137]  [175 155 154]] ... [159 146 138]]] </pre>

Tahap ini merupakan tahap yang melelahkan dan menghabiskan banyak waktu walaupun sudah dibantu menggunakan *software labeling*. Hal itu disebabkan karena pada tahap ini, seluruh *dataset* yaitu gambar sebanyak 2.446 buah dilakukan proses *annotasi* terhadap objek yang terdapat pada tiap-tiap gambar secara manual satu persatu. Pada penelitian ini, proses *annotasi* menghabiskan waktu sebanyak 7 hari.

#### 4.3.2 Split Dataset

Pada tahap ini, *dataset* dibagi menjadi dua bagian, yaitu data *train* dan data *test*. Sesuai dengan konsep *split dataset*, data *train* harus lebih banyak dari data *test*. Pada penelitian ini, pembagian *dataset* merujuk pada penelitian yang dilakukan (Dewi 2018), *dataset* dibagi menjadi 80% data *train* dan 20% data *test*. Pembagian tersebut penulis lakukan secara manual dan acak karena penulis belum menemukan *papper* atau rujukan tentang berapa persen pembagian *dataset* menjadi data *train* dan data *test* yang proporsional. Pada penelitian ini, data *train* berjumlah

2.046 gambar dan data *test* berjumlah 400 gambar. Berikut merupakan beberapa contoh data *train* dan data *test*:



**Gambar 4.8. Data *train***



**Gambar 4.9. Data *test***

Konsep dasar *deep learning* adalah bagaimana sebuah sistem dapat belajar untuk mencapai suatu tujuan tertentu, pada penelitian ini, tujuannya adalah untuk bisa mendeteksi objek dalam sebuah video. Untuk mencapai tujuan tersebut, maka sistem harus diberi pengetahuan tentang data mana yang harus dicapai dan data mana yang bisa digunakan untuk mencapai tujuan tersebut. Data *train* adalah data yang digunakan untuk mencapai tujuan sedangkan data *test* adalah data yang ingin dicapai.

Oleh sebab itu, maka pembagian dataset ini harus dilakukan sebelum melakukan proses *training*. Data *train* berfungsi sebagai bahan ajar untuk sistem agar sistem mampu belajar (*learning*) menggunakan data

yang telah disediakan pada data *train*. Sedangkan data *test* bertujuan agar penulis dapat melihat sampai sejauh mana sistem mempelajari bahan pada data *train* dengan membandingkan nilai *loss* pada setiap *step* saat proses *training*.

#### 4.3.3 *Training*

Setelah melakukan proses ekstraksi fitur dan pembagian *dataset*, maka tahap selanjutnya adalah *training*. *Training* menggunakan *google colaboratory*, sebuah *platform* yang telah disediakan secara gratis oleh *google* dengan *google drive* sebagai media penyimpanannya. Berikut adalah *source code* untuk menghubungkan *google colab* dengan *google drive*:

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

Setelah perintah dijalankan, maka *google colaboratory* akan meminta *authorization code* dari akun *google drive* yang ingin dihubungkan untuk memastikan bahwa akun tersebut setuju untuk terhubung dengan *google colaboratory*. Untuk mendapatkan *code authorization* tersebut, setelah perintah dijalankan maka akan disediakan link yang akan alihkan ke halaman pemilihan akun *google drive*. Setelah *google colaboratory* dan *google drive* terhubung, maka data-data yang tersimpan dalam *google drive* sudah bisa diakses dan diolah di *goole colaboratory*.

Langkah selanjutnya adalah mengupload *dataset* ke akun *google drive* yang telah dihubungkan ke *google colaboratory*. setelah itu mengkonversi semua *dataset* dalam ekstensi file *XML* menjadi *CSV*. Berikut adalah *source code* untuk mengkonversi file berekstensi *XML* menjadi *CSV*:

```
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', 'ymin', 'xmax', 'ymax']
    xml_df = pd.DataFrame(xml_list,
                          columns=column_name)
    return xml_df
```

```
def main():
    for folder in ['train', 'test']:
        image_path = os.path.join(os.getcwd(),
        ('images/' + folder))
        xml_df = xml_to_csv(image_path)
        xml_df.to_csv(('images/' + folder +
        '_labels.csv'), index=None)
        print('Successfully converted xml to csv.')
main()
```

Konversi file *XML* menjadi file *CSV* diperlukan untuk tahapan selanjutnya, yaitu mengkonversi file *CSV* menjadi file *TFRecord* (*Tensorflow Record*). File tersebut digunakan untuk *feeding* data atau sebagai bahan belajar sistem pada proses *training*. Alasan *dataset* perlu diubah menjadi format *TFRecord* karena pada *TFRecord*, penyimpanan *dataset* diubah kedalam bentuk *biner*. Dengan demikian, penyimpanan *dataset* kedalam disk lebih sedikit memakan ruang, lebih cepat membaca dan menyimpan data dan alasan lainnya adalah karena *TFRecord* sangat *powerfull* jika digunakan dengan *Tesnorflow* yang mana sangat sesuai jika digunakan pada penelitian ini. Berikut adalah source code untuk mengkonversi file *CSV* ke file *TFRecord*:

```

def main(_):
    writer =
    tf.python_io.TFRecordWriter(FLAGS.output_path)
    path = os.path.join(os.getcwd(), FLAGS.image_dir)
    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))

```

Dalam *source code generateTRecord* juga terdapat fungsi untuk mendefinisikan kelas-kelas kedalam sebuah tipe data *integer* sebagai *ID* dari masing-masing kelas. Kelas-kelas tersebut sesuai dengan proses *annotasi* pada *dataset*. Pada penelitian ini, kelas dibagi menjadi dua, yaitu kelas mobil dan kelas motor. *Source code* fungsi untuk mengkonversi label objek dari *string* menjadi *integer* adalah sebagai berikut :

```

def class_text_to_int(row_label):
    if row_label == 'mobil':
        return 1
    elif row_label == 'motor':
        return 2
    else:
        return None

```

Langkah terakhir sebelum melakukan proses *training* adalah membuat *label map* dan mengedit file konfigurasi *training*. *Label map* menginformasikan pelatihan apa yang dilakukan pada masing-masing

objek dengan mendefinisikan pemetaan nama kelas ke nomor *ID*. Nomor *ID label map* harus sesuai dengan apa yang didefinisikan dalam *source code TFRecord*. File *label map* dibuat dalam format ekstensi file “.pbtxt”, yang nantinya akan dibutuhkan pada saat konfigurasi sebelum melakukan *training*.

```
item {  
    id: 1  
    name: 'mobil'  
}  
item {  
    id: 2  
    name: 'motor'  
}
```

Konfigurasi pada *pipeline training* mendefinisikan model mana dan parameter apa yang akan digunakan untuk proses *training*. *API* deteksi objek *tensorflow* menggunakan berkas *protobuf* untuk mengkonfigurasi proses *training* dan evaluasi. File konfigurasi dibagi menjadi 5 bagian :

- a. Model : mendefinisikan jenis model apa yang akan dilatih, penelitian ini menggunakan model *Faster R-CNN* dengan beberapa konfigurasi seperti tipe *feature extractor* ‘*Faster R-CNN Inception V2*’, *Stride 16*, *score converter* menggunakan *SOFTMAX*. Berikut konfigurasi model secara rinci:

```
model {
  faster_rcnn {
    num_classes: 2
    image_resizer {
      keep_aspect_ratio_resizer {
        min_dimension: 600
        max_dimension: 1024
      }
    }
    feature_extractor {
      type: 'faster_rcnn_inception_v2'
      first_stage_features_stride: 16
    }
    first_stage_anchor_generator {
      grid_anchor_generator {
        scales: [0.25, 0.5, 1.0, 2.0]
        aspect_ratios: [0.5, 1.0, 2.0]
        height_stride: 16
        width_stride: 16
      }
    }
    first_stage_box_predictor_conv_hyperparams {
      op: CONV
      regularizer {
        l2_regularizer {
          weight: 0.0
        }
      }
      initializer {
        truncated_normal_initializer {
          stddev: 0.01
        }
      }
    }
    first_stage_nms_score_threshold: 0.0
    first_stage_nms_iou_threshold: 0.7
    first_stage_max_proposals: 300
    first_stage_localization_loss_weight: 2.0
    first_stage_objectness_loss_weight: 1.0
    initial_crop_size: 14
    maxpool_kernel_size: 2
    maxpool_stride: 2
    second_stage_box_predictor {
      mask_rcnn_box_predictor {
        use_dropout: false
        dropout_keep_probability: 1.0
        fc_hyperparams {
          op: FC
          regularizer {
            l2_regularizer {
              weight: 0.0
            }
          }
        }
        initializer {
          variance_scaling_initializer {
            factor: 1.0
          }
        }
      }
    }
  }
}
```

```

        uniform: true
        mode: FAN_AVG
    }
}
}
}
second_stage_post_processing {
    batch_non_max_suppression {
        score_threshold: 0.0
        iou_threshold: 0.6
        max_detections_per_class: 100
        max_total_detections: 300
    }
    score_converter: SOFTMAX
}
second_stage_localization_loss_weight: 2.0
second_stage_classification_loss_weight: 1.0
}
}

```

- b. *Train Config* : konfigurasi parameter apa saja yang digunakan dalam proses *training*. Berikut adalah konfigurasinya :

```

train_config: {
    batch_size: 1
    optimizer {
        momentum_optimizer: {
            learning_rate: {
                manual_step_learning_rate {
                    initial_learning_rate: 0.0002
                    schedule {
                        step: 900000
                        learning_rate: .000002
                    }
                    schedule {
                        step: 1200000
                        learning_rate: .000002
                    }
                }
                momentum_optimizer_value: 0.9
            }
            use_moving_average: false
        }
        gradient_clipping_by_norm: 10.0
        fine_tune_checkpoint:
            "C:/tensorflow1/models/research/object_detection/faster_rcnn_inception_v2_coco_2018_01_28/model.ckpt"
            from_detection_checkpoint: true
            load_all_detection_checkpoint_vars: true
    }
    num_steps: 200000
    data_augmentation_options {
        random_horizontal_flip { } }
}

```

- c. *Train Input Config* : konfigurasi untuk menentukan *dataset* yang akan dilatih dengan model. Berikut adalah konfigurasinya :

```
train_input_reader: {
    tf_record_input_reader {
        input_path:
        "C:/tensorflow1/models/research/object_detection/train
        .record"
    }
    label_map_path:
    "C:/tensorflow1/models/research/object_detection/training/labelmap.pbtxt"
}
```

- d. *Eval Config* : konfigurasi untuk menentukan metrik pengukuran evaluasi. Berikut adalah konfigurasinya :

```
eval_config: {
    metrics_set: "coco_detection_metrics"
    num_examples: 7
}
```

- e. *Eval Input Config* :konfigurasi untuk menentukan *dataset* yang akan dievaluasi dengan model. Berikut adalah konfigurasinya :

```
eval_input_reader: {
    tf_record_input_reader {
        input_path:
        "C:/tensorflow1/models/research/object_detection/test.
        record"
    }
    label_map_path:
    "C:/tensorflow1/models/research/object_detection/training/labelmap.pbtxt"
    shuffle: false
    num_readers: 1
}
```

Langkah selanjutnya adalah proses *training*. *Training* dilakukan sebanyak 200 ribu *step* yang mana setiap *step* ditampilkan hasil dari pembelajaran (*learning*) sistem. Setiap *step* akan menampilkan sebuah variabel yang disebut *loss*. *Loss* menunjukkan apakah model berlatih dengan baik atau tidak. Model dapat dikatakan bagus apabila telah mendapatkan *loss* konsisten dibawah 0,05. Berikut merupakan *source code* dalam melakukan *training* :

```

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)

```

Pada saat proses *training*, sistem akan menghasilkan sebuah *checkpoint* yang dibuat secara otomatis oleh *Tensorflow* yang berbentuk *graph tensor*. *Checkpoint* ini bertujuan untuk menyimpan informasi pada *step* tertentu saat proses *training* yang telah diatur sebelumnya. penelitian ini menghabiskan waktu untuk proses *training* selama 2 hari. Penyebab proses *training* menjadi lama karena *runtime Google Colab* sering terputus akibat tidak stabilnya koneksi internet. Jika proses *training* telah selesai, maka tahap terakhir adalah mengkonversi *checkpoint* terakhir atau *graph tensor* menjadi sebuah model yang berkekstensi file ‘.pb’ atau file *protobuf*. Adapun *source codenya* adalah sebagai berikut :

```
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)

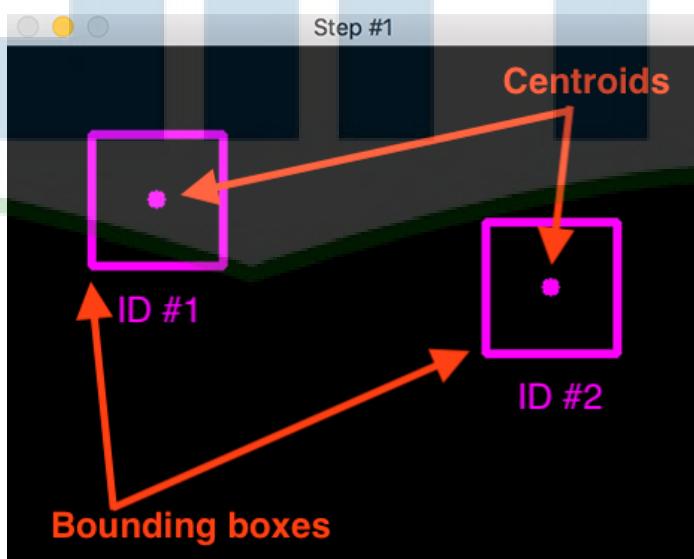
        if FLAGS.input_shape:
            input_shape = [
                int(dim) if dim != '-1' else None
                for dim in FLAGS.input_shape.split(',')
            ]
        else:
            input_shape = None

    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)
```

#### 4.3.4 Tracking

Setelah proses *training*, maka sistem telah mampu mendeteksi objek dan menentukan koordinat dimana objek dalam video itu berada. Objek yang terdeteksi ditandai dengan sebuah kotak pembatas (*bounding box*). Pada tahap ini, dilakukan *tracking* pada objek yang telah terdeteksi. *Tracking* dilakukan menggunakan algoritma *centroid tracking*, alasan penulis menggunakan algoritma ini, karena penulis memerlukan algoritma yang sederhana sesuai dengan batasan penelitian penulis.

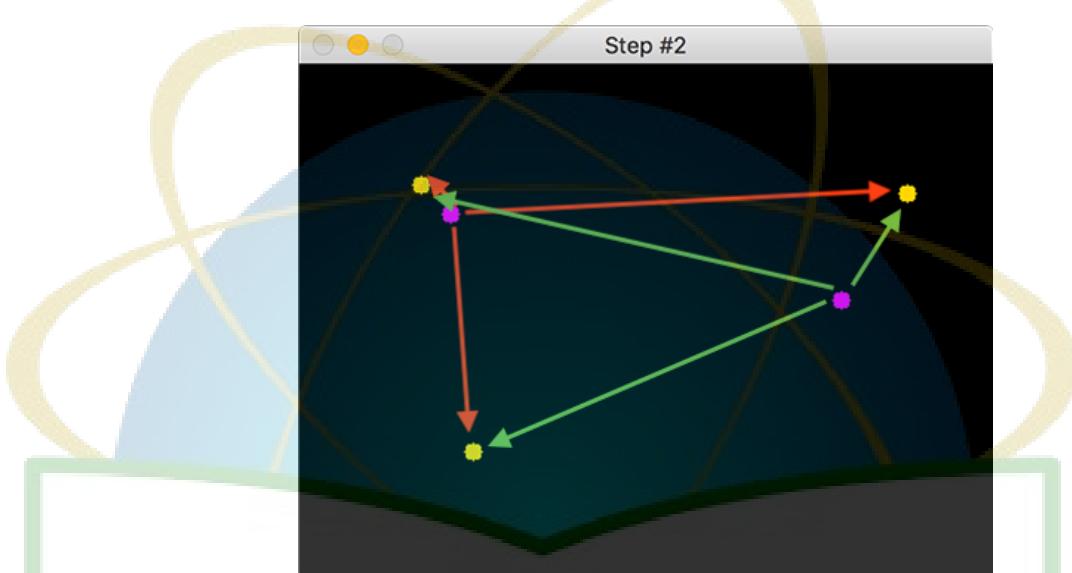
Langkah pertama dalam melakukan *tracking* objek menggunakan algoritma *centroid tracking* adalah dengan menghitung nilai *centroid* atau titik tengah dari setiap *bouding box* pada masing-masing *frame*. Setiap *bounding box* memiliki koordinat dalam *frame* yang mana dengan koordinat itu, penulis dapat menghitung nilai *centroid*. Pada penelitian ini, nilai perhitungan nilai *centroid* diganti dengan nilai pada sisi bawah *bounding box*. Alasan penulis melakukan hal tersebut karena objek kendaraan bergerak dari bawah *frame* ke atas *frame*, sehingga penulis menggunakan sisi bawah *bounding box* sebagai acuan perhitungan karena lebih dekat ke tanah. Alasan lainnya karena untuk meminimalisir objek tumpang tindih. Adapun ilustrasi pada langkah ini dapat dilihat pada gambar berikut:



**Gambar 4.10. Ilustrasi Langkah Perhitungan Nilai *Centroid***

Sumber : (Rosebrock 2018)

Langkah selanjutnya adalah menghitung jarak terdekat antar *centroid* dari *frame* satu ke *frame* selanjutnya dengan menggunakan rumus *Euclidean distance*. Berikut merupakan gambar dari langkah tersebut:

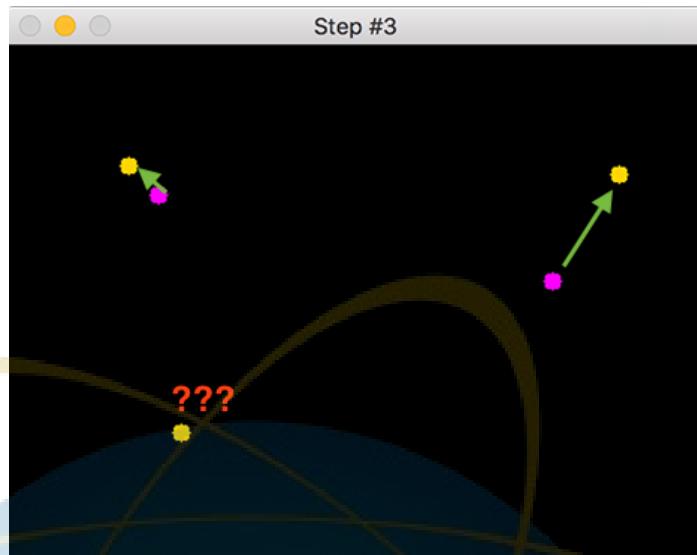


**Gambar 4.11. Gambar Menghitung Jarak Centroid Terdekat**

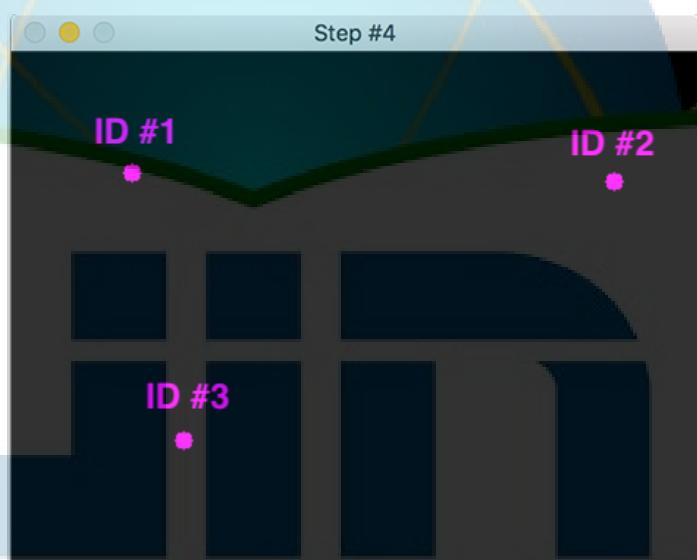
**Antar Frame**

Sumber : (Rosebrock 2018)

*Centroid* yang memiliki jarak terdekat ke *centroid* lain pada tiap-tiap *frame* ditentukan sebagai objek yang sama, maka diberi *ID* yang sama sampai objek tersebut hilang dari *frame*. Sedangkan objek yang tidak memiliki jarak *centroid* terdekat, ditentukan sebagai *ID* baru. Adapun gambaran proses tersebut sebagai berikut:



**Gambar 4.12. Menentukan Centroid Terdekat**  
Sumber : (Rosebrock 2018)



**Gambar 4.13. Menentukan ID Pada Objek Yang Terdeteksi**  
Sumber : (Rosebrock 2018)

Berikut *source code* penentuan koordinat objek dan algoritma *centroid tracking*:

```

box = np.squeeze(boxes)

score2 = np.squeeze(scores)
min_score_thresh2 = 0.8
bbox = box[score2>min_score_thresh2]
rects = []
carLocation1 = {}

for i in range(len(bbox)):
    boxx = box[i, 0:4] * np.array([im_height,
        im_width, im_height, im_width])
    rects.append(boxx.astype("int"))

objects = ct.update(rects)
for i, (objectID, centroid) in enumerate(objects.items()):
    carLocation1[i] = (centroid[0], centroid[1],
        objectID, frameCounter)

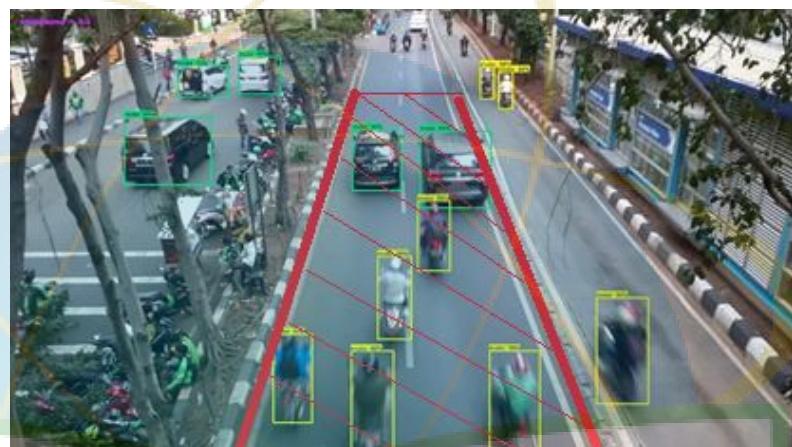
```

#### 4.3.5 Speed Estimation

Perhitungan estimasi kecepatan menggunakan model *parallel line*. Model ini digunakan untuk mengkonversi kecepatan dari piksel ke dunia nyata. Model ini dapat digunakan dengan syarat penulis harus mengetahui kondisi sebenarnya pada lintasan. Penulis menggunakan 5 *line* yang penulis buat sendiri sebagai *landmark* atau batasan untuk setiap kendaraan yang melintas. Kelima *line* tersebut penulis buat sesuai dengan *landmark* pada dunia nyata. Hal tersebut bertujuan untuk perbandingan agar mendapatkan nilai kecepatan sebenarnya pada dunia nyata. Pada setiap kendaraan melewati *line*, maka variabel perhitunganpun akan berbeda.

Pada penelitian ini, penulis tidak mengestimasi kecepatan kendaraan pada semua lintasan. Penulis mengestimasi kecepatan kendaraan hanya pada lintasan yang tidak mempunyai gangguan. Gangguan tersebut

dapat mempengaruhi keakuratan dari estimasi kecepatan kendaraan. Beberapa gangguan yang dapat ditemukan diantaranya adanya objek yang menjadi penghalang antara kendaraan yang melintas dengan kamera, buruknya kualitas piksel pada objek yang berada di lintasan yang banyak gangguan tersebut dan sebagainya. Berikut gambar dari lintasan yang penulis gunakan untuk mengestimasi kecepatan kendaraan:



**Gambar 4.14. Lintasan Estimasi Kecepatan Kendaraan**

Penulis hanya menggunakan lintasan yang penulis arsir dengan warna merah. Jika diperhatikan, pada gambar tersebut lintasan sebelah kiri memiliki banyak gangguan, yaitu banyaknya objek yang menghalangi penglihatan kamera terhadap objek kendaraan yang melintas. Sedangkan pada lintasan busway, penulis tidak gunakan dalam menghitung estimasi kendaraan karena pada saat penulis melakukan wawancara kepada Aiptu Tony untuk mendapatkan validasi estimasinya, beliau tidak mengizinkan kendaraan yang melanggar jalur lalu lintas untuk diestimasi. Berikut adalah *source code* estimasi kecepatan kendaraan :

```

def estimateSpeed(location1, location2):

    d_pixels = math.sqrt(math.pow(location1[0] -
location2[0], 2) + math.pow(location1[1] -
location2[1], 2))

    if (location2[3] == 1):
        d_real = (d_pixels/376)*8
        frame = (location1[2] - location2[2])
        if not(frame == 0):
            speed = (d_real*fps*3.6)/frame
        else:
            speed = 0

    elif (location2[3] == 2):
        d_real = (d_pixels/163)*8
        frame = (location1[2] - location2[2])
        if not(frame == 0):
            speed = (d_real*fps*3.6)/frame
        else:
            speed = 0

    elif (location2[3] == 3):
        d_real = (d_pixels/96)*8
        frame = (location1[2] - location2[2])
        if not(frame == 0):
            speed = (d_real*fps*3.6)/frame
        else:
            speed = 0

    return speed

```

Terdapat kelemahan dari model *parallel line* yang menjadi rujukan penulis dalam menghitung estimasi kecepatan kendaraan pada penelitian ini. Oleh karena itu, penulis melakukan beberapa perbaikan pada formula model *parallel line*. Pada formula model *parallel line* sebelumnya, perhitungan jarak antara kendaraan dengan *line* yang telah dibuat hanya berdasarkan jarak vertikal, hal tersebut menjadi suatu

kelemahan jika diterapkan di Indonesia. Para pengendara di Indonesia kebanyakan selalu menyalip kendaraan yang berada di depannya, sehingga kendaraan tidak melaju secara vertikal sepenuhnya.

Pada model *parallel line* yang telah penulis perbaiki, jarak antara kendaraan dengan *line* tidak dihitung hanya dengan jarak vertikal, namun juga dikombinasikan dengan jarak horizontal. Hal tersebut penulis lakukan dengan menggunakan formula jarak *euclidean*. Sehingga rumus model *parallel line* yang telah penulis perbaiki menjadi:

$$\frac{|A'C'|}{|A'B'|} = \frac{|AC|}{|AB|} = \frac{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}{y_{m+1}-y_m} \quad (4.1)$$

#### 4.4 Testing

Tahap *testing* dilakukan untuk menguji sistem yang telah dirancang. Tahap *testing* diperlukan agar penulis mengetahui apakah sistem telah berjalan sesuai dengan desain dan fungsinya. Penjelasan dan pemaparan mengenai *testing* akan dijelaskan pada BAB V skripsi ini. yaitu pada BAB hasil dan pembahasan. Pembahasan *testing* pada BAB V, masing-masing akan dibahas pada 3 tahap utama dalam perancangan sistem pada penelitian ini sesuai dengan metode *unit testing*.

#### 4.5 Evaluasi Sistem

Pada tahap ini, seluruh sistem yang telah terintegrasi satu sama lain diuji dan dianalisis menggunakan metode *integration testing*. Penulis mengevaluasi sepenuhnya apakah sistem telah sesuai dengan yang diharapkan atau belum. Tahap evaluasi sistem akan dibahas pada BAB V, yaitu mengenai hasil atau *score* yang didapat dari sistem yang telah dibuat.

## BAB V

### HASIL DAN PEMBAHASAN

Pada bab ini akan dibahas mengenai hasil dari setiap proses dalam merancang sistem pada bab IV, yaitu bab implementasi. Penulis merancang sistem pada penelitian ini dengan 3 tahap utama, yaitu deteksi, *tracking* dan menghitung estimasi kecepatan kendaraan. Oleh sebab itu, maka akurasi ketepatan dari perhitungan estimasi kecepatan kendaraan sangat bergantung pada tahap-tahap sebelum perhitungan, yaitu tahap deteksi dan *tracking*.

#### 5.1 Dataset

Pengumpulan dataset diperoleh dari sebuah video berdurasi 2 menit yang dipisahkan tiap framenya sehingga menjadi gambar sebanyak 2.446 gambar dengan ekstensi file ‘.jpg’. Selanjutnya, *dataset* diberi label (*annotasi*). Hasil dari proses *annotasi* ini adalah setiap file dalam *dataset* akan mempunyai satu file lagi dengan ekstensi file ‘.xml’, file ini berisi fitur atau ciri khusus yang merepresentasikan masing-masing file dalam *dataset*. Setelah diannotasi, data displit menjadi 2 bagian, yaitu 80% data train dan 20% data test. Agar dapat diproses lebih lanjut, semua file XML tadi harus dikonversi menjadi satu file dengan ekstensi file ‘.csv’. Berikut merupakan contoh dataset yang telah siap untuk digunakan dalam proses training :

**Tabel 5.1. Tabel Pembagian Dataset**

No	Dataset	Split	Jumlah
1	Data Train	80%	1956
2	Data Test	20%	490
<b>Total</b>		100%	2446

**Tabel 5 2. Tabel Hasil conversi dataset XML menjadi CSV**

filename	width	height	class	xmin	ymin	xmax	ymax
img_20181221_wa0007.jpg	1032	581	mobil	465	148	621	454
img_20181221_wa0008.jpg	1032	581	mobil	451	79	578	330
img_20181221_wa0009.jpg	1032	581	mobil	444	81	595	401
img_20181221_wa0010.jpg	581	1032	motor	225	550	423	657
img_20181221_wa0011.jpg	581	1032	mobil	119	429	466	609
img_20181221_wa0013.jpg	581	1032	motor	103	601	436	757
img_20181221_wa0014.jpg	581	1032	mobil	269	348	408	465
img_20181221_wa0015.jpg	581	1032	mobil	109	546	455	728
img_20181221_wa0017.jpg	1032	581	motor	372	123	524	445
img_20181221_wa0018.jpg	1032	581	motor	217	197	421	457
img_20181221_wa0019.jpg	581	1032	motor	101	518	459	703
img_20181221_wa0020.jpg	581	1032	motor	205	459	428	567
img_20181221_wa0021.jpg	581	1032	mobil	103	446	470	617
img_20181221_wa0023.jpg	581	1032	mobil	84	590	448	796
img_20181221_wa0024.jpg	1032	581	mobil	363	126	532	482
img_20181221_wa0026.jpg	1032	581	mobil	321	194	497	509
img_20181221_wa0028.jpg	581	1032	mobil	166	436	441	575
img_20181221_wa0029.jpg	581	1032	mobil	125	454	478	640
img_20181221_wa0030.jpg	581	1032	motor	80	418	506	626

## 5.2 Training

Setelah memiliki file CSV data *train* dan data *test*, maka langkah selanjutnya adalah mengkonversi masing-masing file tersebut menjadi file *TFRecord* (*TensorFlow Record*). File ini berfungsi untuk melakukan proses *training*. *Training* dilakukan menggunakan algoritma *deep learning* dengan model *Faster R-CNN*. *Training* dilakukan hingga nilai *loss* konsisten dibawah 0,05. Jika

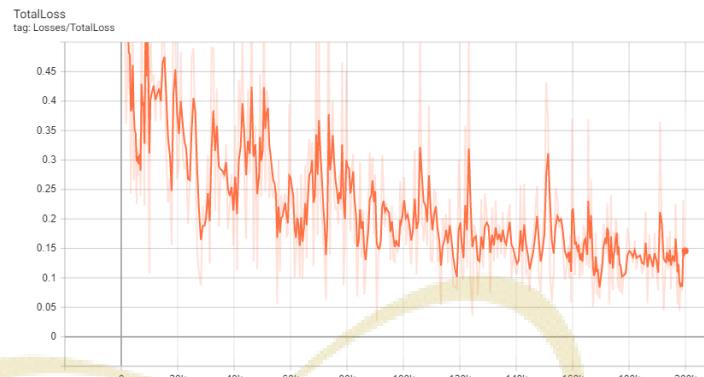
selama proses *training* tidak mendapatkan hasil tersebut, maka penulis memberi batasan *training* sebanyak 200 ribu *step training*. Setiap *step* proses *training*, terdapat juga lamanya waktu dalam memproses *step* tersebut. Pada penelitian ini, hasil proses training mendapatkan nilai *loss* rata-rata dibawah 0,2. Berikut merupakan gambaran proses *training* :

```
I1215 13:44:58.44981/ 140582152066944 learning.py:507] global step 199991: loss = 0.0815 (0.344 sec/step)
INFO:tensorflow:global step 199992: loss = 0.0907 (0.317 sec/step)
I1215 13:44:58.76868 140582152066944 learning.py:507] global step 199992: loss = 0.0907 (0.317 sec/step)
INFO:tensorflow:global step 199993: loss = 0.1422 (0.337 sec/step)
I1215 13:44:59.107574 140582152066944 learning.py:507] global step 199993: loss = 0.1422 (0.337 sec/step)
INFO:tensorflow:global step 199994: loss = 0.1515 (0.314 sec/step)
I1215 13:44:59.424055 140582152066944 learning.py:507] global step 199994: loss = 0.1515 (0.314 sec/step)
INFO:tensorflow:global step 199995: loss = 0.0653 (0.342 sec/step)
I1215 13:44:59.767975 140582152066944 learning.py:507] global step 199995: loss = 0.0653 (0.342 sec/step)
INFO:tensorflow:global step 199996: loss = 0.1635 (0.322 sec/step)
I1215 13:45:00.091226 140582152066944 learning.py:507] global step 199996: loss = 0.1635 (0.322 sec/step)
INFO:tensorflow:global step 199997: loss = 0.1470 (0.351 sec/step)
I1215 13:45:00.443545 140582152066944 learning.py:507] global step 199997: loss = 0.1470 (0.351 sec/step)
INFO:tensorflow:global step 199998: loss = 0.1528 (0.367 sec/step)
I1215 13:45:00.812329 140582152066944 learning.py:507] global step 199998: loss = 0.1528 (0.367 sec/step)
INFO:tensorflow:global step 199999: loss = 0.0850 (0.330 sec/step)
I1215 13:45:01.143643 140582152066944 learning.py:507] global step 199999: loss = 0.0850 (0.330 sec/step)
INFO:tensorflow:global step 200000: loss = 0.2180 (0.357 sec/step)
I1215 13:45:01.502332 140582152066944 learning.py:507] global step 200000: loss = 0.2180 (0.357 sec/step)
INFO:tensorflow:Stopping Training.
I1215 13:45:01.503287 140582152066944 learning.py:777] Stopping Training.
INFO:tensorflow:Finished training! Saving model to disk.
I1215 13:45:01.503537 140582152066944 learning.py:785] Finished training! Saving model to disk.
/usr/local/lib/python3.6/dist-packages/tensorflow_core/python/summary/writer/writer.py:386: UserWarning: /warnings.warn("Attempting to use a closed FileWriter.")
```

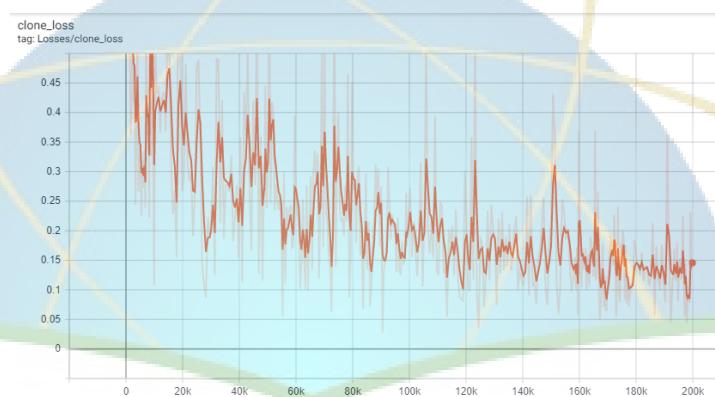
### Gambar 5.1. Proses Training

Proses *training* juga dapat diamati menggunakan *tools tensorboard*. *Tensorboard* dibuat karena *neural network* merupakan sebuah proses yang biasa dikenal sebagai *black box*, artinya *programmer* tidak mengetahui secara detail proses apa yang terjadi pada sistem *neural network*. Dengan bantuan *tensorboard* *programmer* dapat melakukan *debug* dan meningkatkan model. *Tensorboard* membantu untuk memahami ketergantungan antara operasi, perhitungan bobot, menampilkan fungsi *loss* dan banyak informasi lainnya.

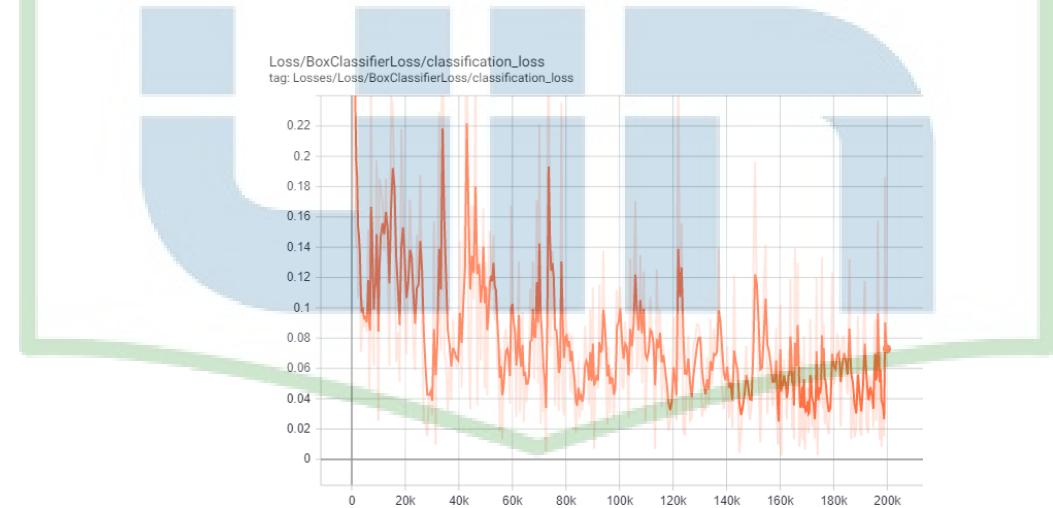
Melalui *tools* tersebut, penulis dapat mengamati progress dari proses *training*. Pada *dashboard tensorboard*, terdapat 10 informasi yang ditampilkan menggunakan grafik. Adapun beberapa grafik utama yang menjadi acuan penulis dalam mengamati proses *training* diantaranya adalah grafik *box classifier loss* dan grafik *RPN loss*. *RPN loss*, merupakan nilai hasil perkalian dari masing-masing tingkat dampak dan kemungkinan yang terjadi. Berikut merupakan gambaran grafik dalam dashboard *tensorboard* :



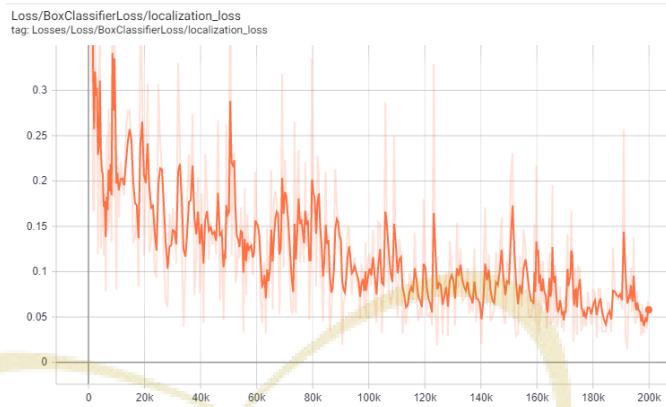
**Gambar 5.2. Grafik Total Loss**



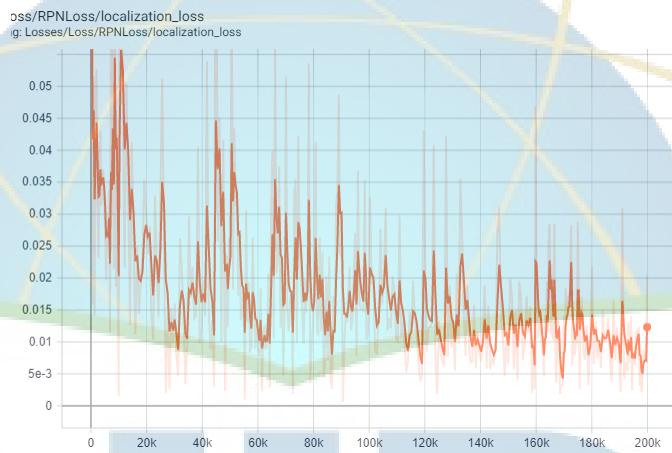
**Gambar 5.3. Grafik Clone Loss**



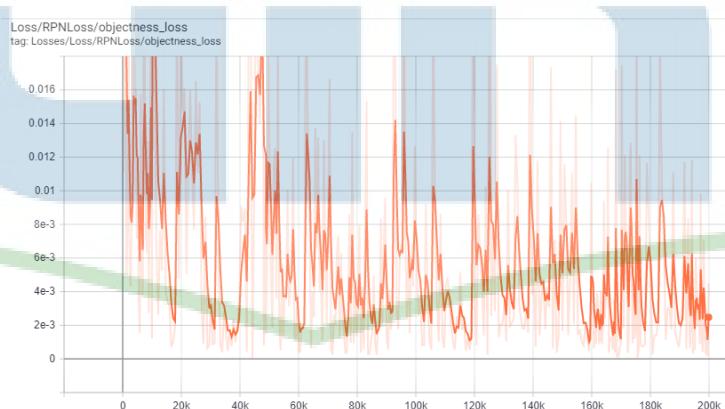
**Gambar 5.4. Grafik Box Classifier Loss/Classification**



**Gambar 5.5. Grafik Box Classifier Loss/Localization**



**Gambar 5.6. Grafik RPN Loss/Localization**



**Gambar 5.7. Grafik RPN Loss/Objectness**

Pada gambar 5.2, grafik total *loss* berguna untuk menghitung seberapa baik model telah belajar dari setiap *step* pada proses *training*. Fungsi *loss* menggambarkan *error* atau kesalahan yang dilakukan model saat belajar, semakin kecil fungsi *loss* maka semakin baik model belajar. Adapun formula dari fungsi *loss* adalah sebagai berikut:

$$\text{Loss} = \frac{1}{2} (y_{pred} - y_{truth})^2 \quad (5.1)$$

Variabel  $y_{pred}$  merupakan nilai dari hasil atau *output* dari proses *training*, sedangkan  $y_{truth}$  merupakan nilai kebenaran. Dikalikan dengan  $\frac{1}{2}$  agar saat diturunkan, fungsi *loss* akan menjadi satu kali *loss* atau menetralisir turunan fungsi kuadrat. Pada penelitian ini, dari grafik tersebut nilai *loss* didapat rata-rata dibawah 0.2 Sedangkan gambar grafik 5.3, yaitu grafik *clone loss*, merupakan hasil salinan dari grafik *loss*. *Clone loss* hanya relevan jika melakukan proses *training* lebih dari satu *GPU*. Fungsinya adalah *tensorflow* akan membuat salinan dari model untuk melatih di setiap *GPU* dan menginformasikan *loss* pada setiap salinannya. Pada penelitian ini, grafik *clone loss* sama dengan grafik *loss*.

Pada gambar 5.4, yaitu grafik *box classifier loss / classification* mendapatkan nilai rata-rata dibawah 0.8 mendekati *step* akhir. Grafik ini menunjukkan kesalahan yang dilakukan model dalam mengklasifikasi objek yang terdeteksi menjadi berbagai macam kelas lain. Dengan demikian, model mendapatkan nilai kesalahan rata-rata dibawah 0.8 dalam mendeteksi objek menjadi kelas yang seharusnya selama proses *training*. Grafik *box classifier loss / localization* yang ditunjukkan pada gambar 5.5 menginformasikan kesalahan yang model lakukan dalam menentukan lokasi *bounding box* pada saat proses *training*.

Grafik *RPN loss / localization* yang ditunjukkan pada gambar 5.6 menginformasikan kesalahan model dalam menentukan lokasi atau kesalahan *bounding box regressor* untuk *RPN* (*Region Proposal Network*). Pada penelitian

ini, nilai *RPN loss / localization* pada saat proses *training* mendekati *step* akhir sebesar 0.015. dengan demikian, model melakukan sedikit kesalahan pada saat proses *training*. Sedangkan pada grafik *RPN loss / Objectness* yang ditunjukkan pada gambar 5.7 menginformasikan kesalahan *classifier* dalam mengklasifikasikan *bounding box* adalah sebuah objek atau sebuah latar belakang.

Selama proses *training*, sistem akan merekam semua proses yang terjadi pada saat proses tersebut dan menyimpannya setiap 5 menit sekali. File tersebut berextensi ‘.ckpt’ diikuti dengan *step* yang terekam pada saat menit tersebut. File yang tersimpan dan merekam *step* terakhir akan dikonversi menjadi sebuah model hasil pelatihan dengan format *protobuf*, dengan ekstensi file ‘.pb’. Setelah mendapatkan model berupa file dengan format *protobuf*, langkah selanjutnya adalah menguji hasil pelatihan tersebut dengan cara mengujinya pada sebuah video.

### 5.3 Testing

Pada sub bab ini, akan dibahas hasil dari masing-masing tahapan dalam perancangan sistem yang telah dijelaskan sebelumnya. Tahap *testing* dibagi menjadi 3 sesuai dengan tahapan dalam perancangan sistem. Yaitu *testing* pada tahap *object detection*, *testing* pada tahap *object tracking*, dan *testing* pada tahap *speed estimation*.

#### 5.3.1 Object Detection

Tahap awal sebelum menghitung estimasi kecepatan kendaraan adalah tahap deteksi objek dalam video. tahap ini menjadi sangat penting karena akurasi dalam memperkirakan kecepatan kendaraan bergantung dari tahap ini. Output dari tahap ini adalah sistem mampu membedakan mana objek yang telah ditentukan untuk dideteksi dan mana yang bukan objek.

Sistem diuji dengan inputan sebuah video berdurasi 17 detik di lokasi yang sama dengan lokasi pengambilan dataset. Sistem bekerja dengan cara memecah video tersebut pada tiap framenya. Selanjutnya mendeteksi objek kendaraan yaitu motor dan mobil pada tiap framenya. Terakhir, semua *frame* yang telah dilakukan proses deteksi disatukan kembali menjadi sebuah video baru. Berikut merupakan hasil dari pengujian sistem pada beberapa *frame* dalam video :



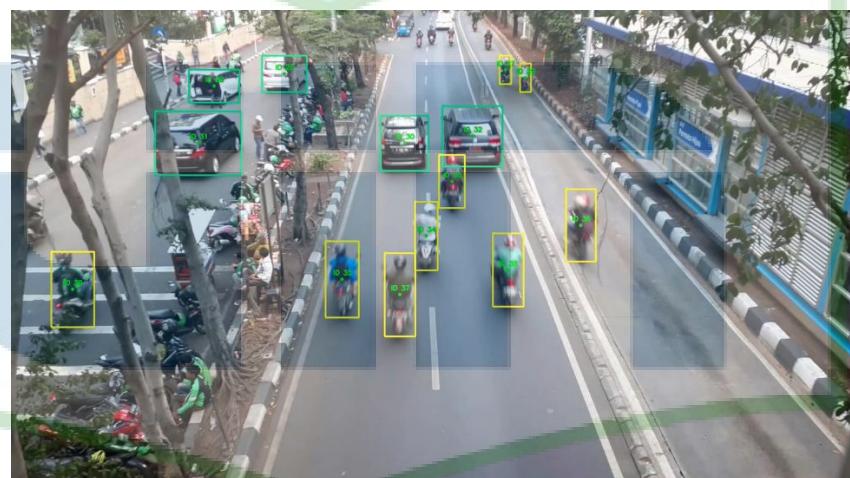
**Gambar 5.8. Gambar Hasil Deteksi Kendaraan Pada Video**

Keberhasilan sistem dalam mendeteksi kendaraan dapat dilihat ketika sistem mampu membedakan mana objek dan mana yang bukan objek yang harus dideteksi. Objek yang terdeteksi ditandai dengan

*bounding box* (kotak pembatas) disekitar objek. Pada sistem ini, objek kendaraan motor ditandai dengan *bounding box* berwarna kuning dan *bounding box* berwarna hijau untuk kendaraan mobil. Pada setiap *bounding box* juga terdapat akurasi dari hasil sistem mendeteksi objek.

### 5.3.2 Object Tracking

Setelah sistem mampu mendeteksi objek dalam video, tahap selanjutnya adalah melacak objek tersebut (*object tracking*) selama objek yang sama berada dalam video. Pada tahap ini, sistem memberi *ID* pada masing-masing objek yang terdeteksi. Akurasi estimasi kecepatan kendaraan juga bergantung pada tahap ini. jika sistem tidak mampu melacak objek atau akurasi *tracking* buruk, maka objek yang sama dapat diberi *ID* yang berbeda pada tiap *frame*, ataupun sebaliknya. Hal ini menjadi sangat berpengaruh pada akurasi estimasi kecepatan kendaraan. Berikut merupakan hasil dari *tracking* objek yang diambil dari frame dalam video :



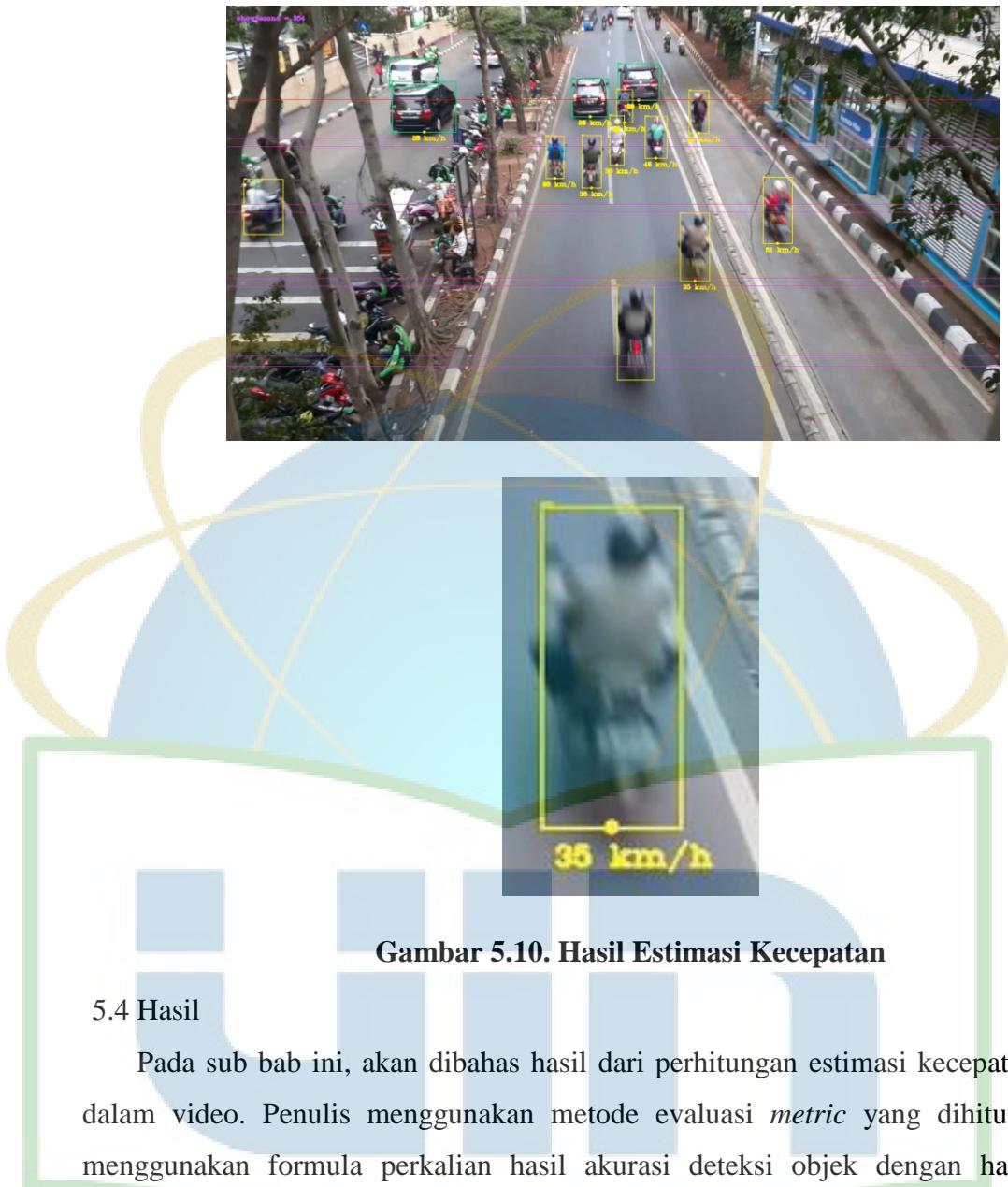


**Gambar 5.9. Hasil Tracking Objek**

Setiap objek yang terdeteksi atau memiliki *bounding box*, diberikan *ID*. Keberhasilan sistem dalam melacak objek yang sebelumnya telah terdeteksi dalam video, ditandai dengan tidak berubahnya *ID* yang telah ditetapkan pada masing-masing objek, sampai objek tersebut menghilang dari frame dalam video.

### 5.3.3 Speed Estimation

Tahap terakhir atau tahap inti dalam perancangan sistem pada penelitian ini adalah memperdiksi kecepatan kendaraan yang melintas dalam video. Estimasi kecepatan menggunakan model *parallel line*. Penulis membuat garis pada video yang telah diketahui jarak sebenarnya pada dunia nyata sebagai acuan. Tiap garis yang dilewati kendaraan mempunyai variabel yang berbeda-beda pada formulanya. Berikut merupakan hasil dari perhitungan estimasi kecepatan:



#### 5.4 Hasil

Pada sub bab ini, akan dibahas hasil dari perhitungan estimasi kecepatan dalam video. Penulis menggunakan metode evaluasi *metric* yang dihitung menggunakan formula perkalian hasil akurasi deteksi objek dengan hasil estimasi kecepatan yang menggunakan metode *NRMSE* (*Normalized Root Mean Square Error*). Adapun formula dari evaluasi *metric* yang penulis gunakan adalah sebagai berikut:

$$S1 = DR \times (1 - NRMSE) \quad (5.2)$$

Yang mana *DR* merupakan hasil dari deteksi objek (*detection rate*), *DR* merepresentasikan rasio antara kendaraan yang terdeteksi dengan kendaraan sebenarnya. Kendaraan dianggap terdeteksi jika setidaknya dilokalisasi 30% dari

seluruh waktu dalam video. Rata-rata akurasi pada hasil deteksi dalam penelitian ini sebesar 99%. Berikut merupakan data hasil deteksi kendaraan yang didapat dari inputan video berdurasi 17 detik.

**Tabel 5.3. Hasil Deteksi Kendaraan**

frame	objek	akurasi
1	motor	99%
1	motor	99%
1	mobil	99%
1	motor	99%
1	motor	99%
1	mobil	99%
1	mobil	99%
1	motor	97%
2	mobil	99%
2	motor	99%
2	mobil	99%
.	.	.
.	.	.
.	.	.
504	mobil	99%
504	motor	99%

Pada tabel hasil deteksi tersebut, video berdurasi 17 detik menghasilkan frame sebanyak 504 yang mana masing-masing frame terdapat hasil dari kendaraan-kendaraan yang terdeteksi beserta dengan akurasinya. Seluruh akurasi dalam tabel tersebut jika dirata-ratakan maka didapat hasil sebesar 99%. Adapun *score detection rate (DR)* dari video tersebut sebesar 1,0. *Score*

*detection rate* didapat dari perhitungan manual penulis terhadap jumlah kendaraan yang terdeteksi benar.

Penulis mencatat secara manual hasil estimasi kecepatan masing-masing kendaraan pada tiap frame dalam video. Variabel kecepatan kendaraan pada dunia nyata, penulis dapatkan dengan bantuan seorang ahli atau seseorang yang sudah terbiasa bekerja pada bidang transportasi dan lalu lintas, sehingga nilai yang didapat dalam variabel tersebut menjadi lebih akurat. Adapun nilai dari hasil estimasi kecepatan dan nilai kecepatan pada dunia nyata dijabarkan pada tabel berikut:

**Tabel 5.4. Tabel Estimasi Kecepatan Kendaraan**

ID	Estimasi	Kecepatan
0	38	28
0	38	28
0	38	28
0	28	28
0	38	28
11	23	38
11	36	38
11	34	38
11	39	38
11	38	38
11	38	38
11	38	38
11	38	38
11	38	38
11	38	38
11	38	38
11	38	38
14	44	50
.	.	.
.	.	.
.	.	.
65	48	45
65	48	45
65	39	45

65	38	45
65	38	45

Berdasarkan table estimasi kecepatan diatas, skor kecepatan pada dunia nyata dan skor estimasi kecepatan dikumpulkan dan dievaluasi secara kolektif dengan menggunakan skor evaluasi *metric*. Dari skor evaluasi *metric* dapat diketahui seberapa akurat hasil estimasi kecepatan dengan kecepatan yang sebenarnya pada dunia nyata. Untuk mendapatkan skor tersebut, dibutuhkan beberapa skor sebagai variabel perhitungan untuk mendapatkan skor evaluasi *metric*. berikut merupakan hasil skor yang didapat dalam penelitian ini :

**Tabel 5.5. Tabel Skor Hasil**

Evaluasi	Skor
Root Mean Square Error	7,116
Normalize Root Mean Square Error	0,103
Evaluasi Metric	0,897

Berdasarkan score-score tersebut, dapat disimpulkan bahwa hasil dari penelitian ini cukup baik. Karena secara teori, semakin mendekati 0 score-score diatas maka sistem semakin baik atau semakin sedikit terdapat kesalahan saat melakukan estimasi kecepatan.

## BAB VI

### PENUTUP

#### 6.1 Kesimpulan

Secara garis besar perancangan sistem pada penelitian ini dibagi menjadi 3 tahapan utama, yaitu tahap deteksi kendaraan, tahap *tracking* kendaraan dan tahap estimasi kecepatan kendaraan. Deteksi kendaraan menggunakan algoritma *deep learning* dengan model *Faster R-CNN*, Untuk *tracking* kendaraan menggunakan algoritma *centroid tracking*. Pada tahap estimasi kecepatan menggunakan model *parallel line*. *Training* sebanyak 200 ribu *step* dengan hasil *loss* rata-rata dibawah 0,2. *Dataset* didapat dari video berdurasi 2 menit yang diambil di lokasi ITC permata Hijau, Jl. Arteri Permata Hijau, Grogol Utara Kebayoran Lama Jakarta Selatan.

Sistem hanya bisa digunakan dengan video yang telah dikonfigurasi pada saat perancangan sistem. Jadi, apabila sistem diuji dengan video yang berbeda, kemungkinan besar akurasi estimasi yang dihasilkan buruk. Sistem bisa digunakan pada video yang berbeda dengan syarat mengkonfigurasi ulang parameter yang terdapat dalam perhitungan estimasi kecepatan kendaraan.

Sistem diuji dengan inputan video berdurasi 17 detik dan menghasilkan *frame* sebanyak 504 *frame* dengan hasil akurasi rata-rata deteksi mencapai 99%. *Score detection rate* sebesar 1,0, yang berarti sistem mendapatkan *score* sempurna dalam hal deteksi kendaraan. Pada estimasi kecepatan kendaraan, Sistem mendapatkan *score RMSE* sebesar 7,116 dan mendapatkan *score NRMSE* sebesar 0,103. Dengan kedua *score* tersebut, sistem berhasil mendapatkan *score evaluasi metric* sebesar 0,897.

#### 6.2 Saran

Dalam penelitian ini masih terdapat keterbatasan, oleh karena itu, penulis mengharapkan adanya pengembangan pada penelitian selanjutnya. Berdasarkan

penelitian yang telah dilakukan, maka penulis dapat memberikan beberapa saran sebagai berikut:

- a. Menggunakan model estimasi kecepatan yang lebih baik, yang mencakup seluruh lintasan dalam video, sehingga estimasi kecepatan yang dihasilkan lebih luas dan lebih akurat.
- b. Menggunakan metode validasi estimasi kecepatan yang lebih akurat, sehingga dapat diketahui secara lebih tepat tingkat keberhasilan dari model estimasi.
- c. Menggunakan kamera pengawas yang sesungguhnya.
- d. Menggunakan model yang dapat diuji ke video-video yang bebeda, dengan kamera dan lokasi yang berbeda.

## DAFTAR PUSTAKA

- Aggarwal, Charu C. 2018. *Neural Networks and Deep Learning*.
- Dani. 2019. “ARTI ESTIMASI: Pengertian, Menurut Para Ahli, Manfaat & Contohnya.” <https://thegeekhost.com/arti-estimasi/> (January 31, 2020).
- Dewi, Syarifah Rosita. 2018. “Deep Learning Object Detection Pada Video.”
- Dharmadi, Richard. 2018. “Faster R-CNN.” <https://medium.com/nodeflux/convolutional-neural-net-untuk-deteksi-objek-f14d72f11ba6> (November 19, 2019).
- Eckroth, Joshua. 2018. *Python Arti Fi Cial Intelligence Projects for Beginners*.
- Giannakeris, Panagiotis, and Alexia Briassouli. “Speed Estimation and Abnormality Detection from Surveillance Cameras.” : 93–99.
- Gong, Shengrong et al. 2019. *Advanced Image and Video Processing Using MATLAB*.
- Himanshu Singh, Santanu Pattanayak. 2019. *Practical Machine Learning and Image Processing*.
- Howse, Joseph. 2015. *Android Application Programming with OpenCV 3*.
- Hua, Shuai, Manika Kapoor, and David C Anastasiu. “Vehicle Tracking and Speed Estimation from Traffic Videos San Jos’.” : 153–60.
- Huang, Tingting. “Traffic Speed Estimation from Surveillance Video Data Institute for Transportation , Iowa State University.” : 161–65.
- Kompas.com. 2019. “Ingat Lagi Batas Kecepatan Kendaraan Di Jalan.” <https://otomotif.kompas.com/read/2019/04/14/113100015/ingat-lagi-aturan-batas-kecepatan-kendaraan-di-jalan> (January 14, 2020).
- Kumar, Amit et al. “A Semi-Automatic 2D Solution for Vehicle Speed Estimation from Monocular Videos.” : 137–44.

- Michelluci, Umberto, and Jojo Moolayil. 2019. *Advanced Applied Deep Learning*.
- Moeslund, Thomas B. 2019. *Introductionn to Video and Image Processing*. 2017.
- Mohri, Mehryar, Afshin Rostamizadeh, and Ameet Talwalker. 2018. *Foundation Of Machine Learning*.
- Redmon, Joseph Chet. 2018. “YOLO: Real-Time Object Detection.”  
<https://pjreddie.com/darknet/yolov1/> (January 17, 2020).
- Rohimah, Wudda. 2017. “Pengenalan Plat Nomor Kendaraan Indonesia Menggunakan Principal Component Analysis Dan Metode K-Nearest Neighbor.” <http://repository.usu.ac.id/handle/123456789/2378>.
- Rosebrock, Adrian. 2018. “Simple Object Tracking.”  
<https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/> (January 23, 2020).
- Sachan, Ankit. 2018. “Zero to Hero: Guide to Object Detection Using Deep Learning: Faster R-CNN,YOLO,SSD.” <https://cv-tricks.com/object-detection/faster-r-cnn-yolo-ssd/> (January 14, 2019).
- Saha, Sumit. 2018. “A Comprehensive Guide to Convolutional Neural Networks — the ELI5 Way.” <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53> (January 18, 2020).
- Sofia, Nadhifa. 2018. “Convolutional Neural Network.”  
<https://medium.com/@nadhifasofia/1-convolutional-neural-network-convolutional-neural-network-merupakan-salah-satu-metode-machine-28189e17335b> (November 22, 2019).
- Techinasia, ID. 2018. “Apa Itu Machine Learning Dan Perbedaanya Dengan Deep Learning.” <https://inixindo.co.id/index.php/it-forum/79-pemrogramman/680-apa-itu-machine-learning-dan-perbedaannya-dengan-deep-learning> (January 17, 2020).

Tran, Minh-triet et al. “Traffic Flow Analysis with Multiple Adaptive Vehicle Detectors and Velocity Estimation with Landmark-Based Scanlines.” : 100–107.

Vasilev, Ivan. 2019. *Python Deep Learning: Exploring Deep Learning Techniques and Neural Network Architectures with PyTorch, Keras, and TensorFlow*.

Zhiwei, He, Liu Yuanyuan, and Ye Xueyi. 2007. “Models of Vehicle Speeds Measurement with a Single Camera.” (2): 283–86.



## DAFTAR LAMPIRAN

### Lampiran 1

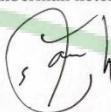
#### Surat Pernyataan Data Validasi

**SURAT PERNYATAAN**

Yang bertanda tangan dibawah ini:

Nama	:	Alfi Salim
NIK	:	3173050305980003
Alamat	:	Jl. KPBD RT.001/02 Sukabumi Selatan, Kebon Jeruk, Jakarta Barat
Tempat, Tanggal Lahir	:	Jakarta, 03 Mei 1998
Agama	:	Islam
Kewarganegaraan	:	Indonesia
Nomor Induk Mahasiswa	:	11160910000014
Nomor Telepon	:	0895615140683

Menyatakan bahwa benar saya telah minta keterangan masalah perkiraan kecepatan kendaraan dalam sebuah video dengan pihak terkait yang mengerti masalah tersebut.  
 Demikian surat pernyataan ini saya buat dengan sebenar-benarnya untuk keperluan pembuatan skripsi di Universitas Islam Negeri Syarif Hidayatullah Jakarta.

Yang memberikan keterangan,  
  
TONY SUATRAT  
 NIPN No.: 76090204

Jakarta, 30 Desember 2019  
  
 Alfi Salim

## Lampiran 2

### Data Validasi

ID	Kecepatan	ID	Kecepatan	ID	Kecepatan	ID	Kecepatan
0	28	14	50	16	49	18	36
0	28	14	50	16	49	18	36
0	28	15	29	16	49	21	36
0	28	15	29	16	49	21	36
0	28	15	29	16	49	21	36
11	38	15	29	16	49	21	36
11	38	15	29	17	56	21	36
11	38	15	29	17	56	21	36
11	38	15	29	17	56	21	36
11	38	15	29	17	56	21	36
11	38	15	29	17	56	21	36
11	38	15	29	17	56	21	36
11	38	15	29	17	56	21	36
11	38	15	29	17	56	21	36
11	38	15	29	17	56	21	36
11	38	15	29	17	56	21	36
11	38	15	29	17	56	21	36
11	38	15	29	17	56	21	36
11	38	15	29	17	56	21	36
11	38	15	29	17	56	21	36
11	38	15	29	17	56	21	36
11	38	15	29	17	56	21	36
11	38	15	29	17	56	21	36
11	38	15	29	17	56	21	36
11	38	15	29	17	56	21	36
14	50	16	49	18	36	23	39
14	50	16	49	18	36	23	39
14	50	16	49	18	36	23	39
14	50	16	49	18	36	23	39
14	50	16	49	18	36	23	39
14	50	16	49	18	36	23	39
14	50	16	49	18	36	23	39
14	50	16	49	18	36	23	39

ID	Kecepatan	ID	Kecepatan	ID	Kecepatan	ID	Kecepatan
23	39	30	26	34	29	35	28
	49	32	38	34	29	35	28
25	49	32	38	34	29	35	28
25	49	32	38	34	29	35	28
25	49	32	38	34	29	35	28
25	49	32	38	34	29	35	28
25	49	32	38	34	29	35	39
25	49	32	38	34	29	36	39
25	49	32	38	34	29	36	39
25	49	32	38	34	29	36	39
25	49	32	38	34	29	36	39
25	49	32	38	34	29	36	39
25	49	32	38	34	29	36	39
25	49	32	38	34	29	36	39
30	26	32	38	34	29	36	39
30	26	32	38	34	29	36	39
30	26	32	38	34	29	36	39
30	26	32	38	34	29	36	39
30	26	33	39	34	29	36	39
30	26	33	39	34	29	36	39
30	26	33	39	35	28	36	39
30	26	33	39	35	28	36	39
30	26	33	39	35	28	37	28
30	26	33	39	35	28	37	28
30	26	33	39	35	28	37	28
30	26	33	39	35	28	37	28
30	26	34	29	35	28	37	28

ID	Kecepatan	ID	Kecepatan	ID	Kecepatan	ID	Kecepatan
37	28	43	38	47	46	55	45
37	28	43	38	47	46	55	45
37	28	43	38	47	46	55	45
37	28	43	38	47	46	55	45
37	28	43	38	47	46	55	45
37	28	43	38	47	46	55	45
37	28	43	38	47	46	55	45
37	28	43	38	47	46	60	45
37	38	43	38	47	46	60	45
41	38	46	38	47	46	60	45
41	38	46	38	47	46	60	45
41	38	46	38	51	40	60	45
41	38	46	38	51	40	60	45
41	38	46	38	51	40	60	45
41	38	46	38	51	40	60	45
41	38	46	38	51	40	60	45
41	38	46	38	51	40	60	45
41	38	46	38	51	40	60	45
41	38	46	38	51	40	60	45
41	38	46	38	51	40	60	45
41	38	46	38	51	40	60	45
41	38	46	38	51	40	60	45
41	38	46	38	51	40	60	45
41	38	46	38	51	40	60	45
41	38	46	38	51	40	60	45
41	38	46	38	51	40	60	45
41	38	46	38	51	40	60	45
41	38	46	38	51	40	60	45
43	38	46	38	55	45	63	45
43	38	46	38	55	45	63	45
43	38	46	46	55	45	63	45
43	38	47	46	55	45	63	45
43	38	47	46	55	45	63	45
43	38	47	46	55	45	63	45

### Lampiran 3

#### Surat Pernyataan Wawancara



## Lampiran 4

### Daftar Pertanyaan Wawancara

#### DAFTAR PERTANYAAN WAWANCARA

Tanggal : 30 Januari 2020

Waktu : 11.00 – 12.00

Narasumber : Tony Suatrat

Jabatan : Ajun Inspektur Polisi Satu (Aiptu)

Daftar pertanyaan wawancara ini berfungsi untuk menjawab rumusan masalah pada penelitian yang berjudul “**Estimasi Kecepatan Kendaraan Melalui Video Pengawas Lalu Lintas Menggunakan Parallel Line Model**”. Berikut daftar pertanyaan seberapa penting penelitian ini dilakukan untuk memberikan kontribusi terhadap kemajuan lalu lintas di Indonesia.

#### Daftar Pertanyaan :

1. Bagaimana tanggapan Bapak mengenai prilaku pengendara dewasa ini?
2. Apakah lalu lintas tempat bapak bertugas semakin padat setiap tahunnya?
3. Apakah semakin sulit mengatur lalu lintasnya?
4. Berapa lama waktu Bapak bertugas di lalu lintas?
5. Apakah ada yang melanggar lalu lintas saat Bapak bertugas?
6. Bila ada, Berapa rata-rata pelanggaran lalu lintas yang Bapak temukan pada saat bertugas?
7. Apakah saat tidak ada yang bertugas di lalu lintas, pelanggaran lalu lintas meningkat?
8. Mengapa hal tersebut bisa terjadi?
9. Apa saja contoh pelanggaran lalu lintas?
10. Apakah pelanggaran lalu lintas dalam hal melanggar batas kecepatan sering terjadi?

11. Apakah diperlukan sebuah sistem yang dapat mengawasi perilaku pengendara melalui kamera pengawas lalu lintas, khususnya dalam hal kecepatan?
12. Berapa persentase sistem tersebut diperlukan?
13. Mengapa sistem tersebut sangat diperlukan?

Jawaban :

1. Dewasa ini, kesadaran akan kedisiplinan bagi para pengendara semakin berkurang. Hal ini sangat memprihatinkan.
2. Benar, lalu lintas yang saya atur terus meningkat setiap tahunnya. Saya memperkirakan peningkatan tersebut sekitar 2% sampai 3% kendaraan.
3. Tidak, karena kami mempunyai kerja sama yang baik antar anggota saat melakukan pengawasan lalu lintas, ditambah lagi dengan adanya rambu-rambu dan lampu pengatur lalu lintas.
4. Petugas lalu lintas berada di lapangan sekitar 6 sampai 8 jam persifit
5. Masih ada saja yang masih melanggar walaupun ada petugas yang sedang bertugas. Saya juga tidak mengerti, saya asumsikan bahwa pelanggaran masih terjadi karena pengendara berpikir saat itu tidak ada petugas yang bertugas.
6. Biasanya saya menemukan sekitar 20 sampai 30 pelanggaran baik itu sepeda motor maupun mobil.
7. Jelas meningkat, padahal ada ataupun tidak ada petugas, peraturan lalu lintas harus tetap ditaati.
8. Hal tersebut bisa terjadi karena kultur masyarakat pengguna lalu lintas yang kurang sadar akan disiplin lalu lintas. Kebanyakan dari mereka hanya mementingkan ego masing-masing. Padahal hal tersebut sangat berbahaya untuk dirinya sendiri maupun orang lain.
9. Contoh-contoh pelanggaran lalu lintas yang terjadi seperti menerobos lampu lalu lintas, melanggar rambu-rambu lalu lintas, tidak menggunakan helm, tidak menyalakan lampu kendaraan dan lain sebagainya.

10. Pelanggaran dalam hal batas kecepatan juga sering terjadi, namun pelanggaran tersebut tidak terlalu terlihat jelas sehingga banyak masyarakat yang kurang peduli akan hal tersebut
11. Pasti, saya rasa sistem seperti itu sangat diperlukan seperti sistem Et-le seperti yang sudah diterapkan di Jalan Jendral Sudirman hingga ke Jalan Gajah Mada.
12. Saya rasa itu tidak bisa diukur dengan tingkat persen karena kita memang sangat membutuhkan sistem tersebut dan juga sistem-sistem serupa seperti pelanggaran jalur kendaraan, penerobosan lampu lalu lintas dan sebagainya.
13. Karena masyarakat Indonesia hanya menaati peraturan lalu lintas jika terdapat petugas yang sedang bertugas, mungkin karena takut diberikan sanksi tilang. Bila menggunakan sistem, perilaku pengendara dapat terpantau selama 24 jam yang artinya, semakin banyak sistem ini diterapkan, maka akan semakin berkurang juga pelanggaran lalu lintas yang terjadi. Bila menggunakan sistem juga peanggaran lalu lintas akan ditindak lanjuti tanpa pandang bulu.