BAB I

PENDAHULUAN

A. Latar Belakang

Data volume kendaraan yang lewat di jalan raya sangat dibutuhkan baik oleh Jasa Marga sebagai pihak pengembang dan pengelola jalan tol, dinas lalu lintas jalan raya dan pemerintah setempat. Data ini dapat digunakan untuk proses penghitungan kepadatan kendaraan, penghitungan frekuensi kendaraan, penghitungan jumlah polusi rata-rata, juga dapat digunakan sebagai referensi untuk perbaikan jalan yang sudah ada, pelebaran jalan, penambahan jalan baru pembuatan atau penataan rute baru, penataan rambu-rambu lalu lintas dan lainlain. Untuk memperoleh data volume kendaraan yang lewat di jalan raya masih dilakukan dengan cara manual yaitu dengan menugaskan beberapa orang untuk berada di lapangan (tempat survei) dan menghitung setiap kendaraan yang lewat, kemudian dibagi dengan rentang waktu tertentu. Biasanya pengambilan data ini dilakukan pada dua kondisi yaitu pada jam sibuk (peak time) dan jam tidak sibuk (off time). Dalam penghitungan secara manual ini masih terdapat banyak kelemahan, diantaranya yaitu tingkat keakuratan data yang masih kurang, data masih berada di tempat penghitungan dan untuk mengumpulaknnya masih diperlukan waktu yang relatif lama. Berdasarkan pada kondisi tersebut, fokus penelitian ini adalah melakukan pengembangan yang sudah ada, adapun fokus tersebut dititik beratkan pada perancangan jaringan penghitung kendaraan secara otomatis berbasis client server dengan tujuan untuk merancang penyedia informasi volume kendaraan

secara real time dan untuk membantu menyediakan informasi volume kendaraan kepada pihak terkait secara cepat dan akurat.

B. Rumusan Masalah

Adapun rumusan masalah pada latar belakang yaitu Bagaimana merancang sebuah aplikasi Sistem perhitungan jenis dan jumlah kendaraan secara realtime berbasis android.

C. Tujuan Penelitian

Adapun tujuan yang diharapkan adalah :

1. Akan membuat sebuah aplikasi monitoring kendaraan lewat.
2. situasi arus kendaraan lewat pada ruas jalan yang telah ditentukan yang diperoleh dari hasil perkaman, dengan aplikasi ini Dinas Perhubungan bisa menentukan kebijakan apa yang harus dilakukan sehubungan dengan mendapatkannya informasi hasil pemantauan kendaraan yang lewat

D. Manfaat Penelitian

Adapun manfaat pada penelitian ini adalah sebagai berikut :

1. Meningkatkan kemampuan dalam membuat dan merancang aplikasi sesuai dengan kebutuhan
2. Memberikan informasi jumlah kendaraan yang telah didteksi oleh kamera

E. Batasan Masalah

Batasan masalah dalam pembuatan aplikasi yang akan dibangun adalah:

1. Aplikasi hanya diuji coba di lingkup kampus universitas muhammadiyah parepare.

2. Aplikasi ini hanya medeteksi jumlah kendaraan yang melewati kamera
3. Aplikasi ini mendeteksi jenis kendaraan dengan menggunakan kamera

F. Sistematika Penulisan

Sistematika penulisan dibagi menjadi beberapa bagian yaitu.

BAB I PENDAHULUAN

Menjelaskan tentang latar belakang masalah, maksud dan tujuan yang akan dicapai dalam pelaksanaan skripsi ini. Penulis juga memberikan batasan-batasan masalah pada bagian akhir dan Sistematika Penulisan.

BAB II TINJAUAN PUSTAKA

Menguraikan teori yang erat hubungannya dengan cara kerja komponen yang digunakan dan penjelasan umum dalam pembuatan Aplikasi dengan.

BAB III METODOLOGI PENELITIAN

Berisi tentang metode penelitian dengan menggunakan metode survey dimana ingin diketahui secara langsung sistem informasi, metode yang digunakan untuk pengumpulan data adalah wawancara, pengamatan dan metode pustaka serta alat dan bahan penelitian.

BAB II

TINJAUAN PUSTAKA

A. Penelitian Terdahulu

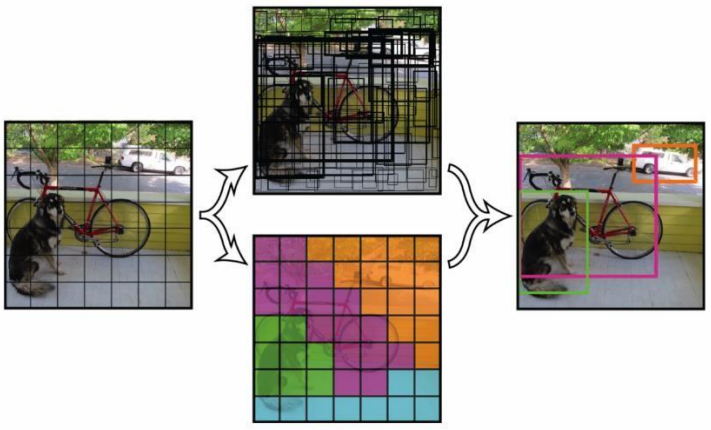
Implementasi Object Tracking Untuk Mendeteksi Dan Menghitung Jumlah Kendaraan Secara Otomatis Menggunakan Metode Kalman Filter Dan Gaussian Mixture Model, Havez Vazirani Al Kautsar, Universitas dipunegoro 2017. Kepadatan lalu lintas dapat dikontrol dengan cara memperoleh dan mengelola data volume kendaraan di jalan raya. Pada umumnya, proses akuisisi data volume kendaraan yang lewat di jalan raya masih dilakukan dengan cara manual yaitu dengan menugaskan beberapa orang untuk berada di lapangan dan menghitung setiap kendaraan yang lewat, kemudian dibagi dengan rentang waktu tertentu. Dalam penghitungan secara manual ini masih terdapat banyak kelemahan seperti pengumpulan data yang lama, dan jumlah sumber daya manusia yang dibutuhkan banyak. Berdasarkan pada kondisi tersebut dibutuhkan sistem penghitung dan deteksi kendaraan otomatis yang akurat sebagai pemantau kontrol lalu lintas dan analisa lalu lintas. Pada saat ini telah dikembangkan sistem deteksi kendaraan menggunakan sensor, Radio Frequency Identifier atau perangkat keras lain yang diintegrasikan oleh perangkat lunak pada mikrokontroller dan bekerja otomatis untuk mendeteksi kecepatan serta menghitung jumlah kendaraan yang melintas di jalan raya. Kelemahan detektor tersebut hanya dapat mendeteksi pada jangkauan yang sempit, perancangan sistem dan operasional yang rumit, serta biaya operasional yang besar. Berdasarkan pada beberapa hal tersebut, penelitian ini dikembangkan dengan fokus perancangan sistem deteksi dan penghitung jumlah

kendaraan dengan metode Kalman Filter dan Gaussian Mixture Model (GMM). Sistem ini mendapatkan hasil yang paling akurat pada pagi hari (pencahayaan 10.000- 25.000 lux) dengan nilai F1 Score sebesar 0,91111, sedangkan penghitungan kendaraan yang paling tidak akurat terjadi pada malam hari (pencahayaan 0,27-1,0 lux) dengan F1 Score 0,16071.

Rancang Bangun Penghitung Kendaraan Secara Otomatis Berbasis Client Server, Tarnoto**, Universitas Gunadarma 2019.** Saat ini, untuk memperoleh data volume kendaraan yang lewat di jalan raya masih dilakukan dengan cara manual yaitu dengan menugaskan beberapa orang ke lapangan (tempat survei), dalam penghitungan ini masih terdapat banyak kelemahan, diantaranya yaitu tingkat keakuratan data yang masih kurang, data masih terdapat di beberapa tempat penghitungan dan untuk mengumpulkannya diperlukan waktu yang relatif lama serta masih ada kemungkinan duplikasi data. Tujuan dari penelitian ini adalah untuk merancang sebuah penyedia informasi volume kendaraan secara otomatis berbasis client server. Metode penelitian yang dipakai adalah peneliti mencari data pada sumber literatur dan dari objek penelitian serta merancang model alatnya kemudian mengujinya dan mengevaluasi hasil pengujian. Dalam perancangan, untuk mendeteksi kendaraan yang lewat menggunakan LDR dan laser kemudian data diproses oleh mikrokontroler dan dikirimkan ke server. Selanjutnya data tersebut disimpan dalam database informasi volume kendaraan oleh aplikasi server serta ditampilkan dalam tabel dan grafik. Dari hasil pengujian, dapat disimpulkan bahwa rancangan dapat diimplementasikan dengan baik karena dapat menghitung secara otomatis dan waktu yang dibutuhkan menjadi singkat.

B. YOLO Object Detection

YOLO (*you only look once*) merupakan algoritma *real object detection* yang baru–baru ini sangat popular untuk dikembangkan. YOLO menggunakan pendekatan yang sangat berbeda dengan algoritma sebelumnya, yakni menerapkan jaringan syaraf tunggal pada keseluruhan gambar. Pendeteksian objek dilakukan dengan membingkai objek yang akan dideteksi sebagai *regression problem* dan memisahkan special pada *bounding boxes* dan *class probabilities*. Dengan menggunakan *single neural network* untuk memprediksi *bounding boxes* dan *class probabilities* dari seluruh gambar pada satu kali evaluasi. Karena metode ini menggunakan *single neural network* untuk semua *detection pipeline*, maka permforma deteksi ini bisa dioptimasi dari *end-to-end*



YOLO memiliki kemampuan pada base model yang bisa memproses *45 frame per second* secara *real time* pada versi yang lebih kecil bisa memproses *155 frame per second* secara *real-time*. YOLO juga memiliki arsitektur yang sederhana yaitu jaringan syaraf convolutional.

Layer	kernel	stride	output shape
Input			(416, 416, 3)
Convolution	3×3	1	(416, 416, 16)
MaxPooling	2×2	2	(208, 208, 16)
Convolution	3×3	1	(208, 208, 32)
MaxPooling	2×2	2	(104, 104, 32)
Convolution	3×3	1	(104, 104, 64)
MaxPooling	2×2	2	(52, 52, 64)
Convolution	3×3	1	(52, 52, 128)
MaxPooling	2×2	2	(26, 26, 128)
Convolution	3×3	1	(26, 26, 256)
MaxPooling	2×2	2	(13, 13, 256)
Convolution	3×3	1	(13, 13, 512)
MaxPooling	2×2	1	(13, 13, 512)
Convolution	3×3	1	(13, 13, 1024)
Convolution	3×3	1	(13, 13, 1024)
Convolution	1×1	1	(13, 13, 125)

Layer	kernel	stride	output shape
Input			(416, 416, 3)
Convolution	3×3	1	(416, 416, 16)
MaxPooling	2×2	2	(208, 208, 16)
Convolution	3×3	1	(208, 208, 32)
MaxPooling	2×2	2	(104, 104, 32)
Convolution	3×3	1	(104, 104, 64)
MaxPooling	2×2	2	(52, 52, 64)
Convolution	3×3	1	(52, 52, 128)
MaxPooling	2×2	2	(26, 26, 128)
Convolution	3×3	1	(26, 26, 256)
MaxPooling	2×2	2	(13, 13, 256)
Convolution	3×3	1	(13, 13, 512)
MaxPooling	2×2	1	(13, 13, 512)
Convolution	3×3	1	(13, 13, 1024)
Convolution	3×3	1	(13, 13, 1024)
Convolution	1×1	1	(13, 13, 125)

Jaringan saraf ini hanya menggunakan jenis lapisan standar; konvolusi dengan kernel 3 x 3 dan *max pooling* dengan 2 x 2 kernel. Lapisan konvolusional terkahir 1 x 1 kernel digunakan untuk mengecilkan data ke bentuk 13 x 13 x 125.

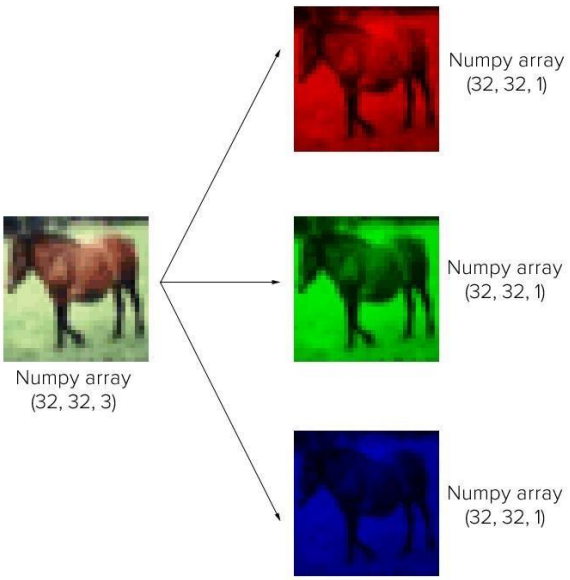
13 x 13 ini seharusnya terlibat familiar: itu adalah ukuran *grid* yang dibagi menjadi gambar. 125 merupakan *channel* untuk setiap *grid*. 125 ini berisi data

untuk kotak pembatas dan prediksi kelas. Setiap sel *grid* memprediksi 5 kotak sekeliling dan dijelaskan oleh 25 elemen data.

- X, Y untuk lebar dan tinggi kotak pembatas (dua elemen data)
- Skor keyakinan (satu elemem data)
- Distribusi probabilitas yang lebih dari 20 kelas (20 elemen data)

Berikut ini urutan rumus atau cara yang digunakan untuk bisa menerapkan metode YOLO *object detection*.

a. Convolutional Layer



Gambar 2. 1 Pemisahan RGB pada Gambar

Gambar diatas adalah RGB (*Red, Green, Blue*) *image* berukuran 32 x 32 pixels yang sebenarnya adalah multidimensional array dengan ukuran 32 x 32 x 3 (3 adalah jumlah *channel*). Convolutional layer terdiri dari neuron yang tersusun sedemikiran rupa sehingga membentuk sebuah filter dengan Panjang dan tinggi (*pixels*). Sebagai contoh, layer pertama pada *feature extraction layer* biasanya

adalah *convolutional layer* dengan ukuran 5 x 5 x 3. Panjang 5 *pixels*, tinggi 5 *pixels*, dan tebal/jumlah 3 buah sesaui dengan *channel* dari gambar tersebut.

Untuk menghitung dimensi dari *feature map* bisa menggunakan rumus seperti dibawah ini:

$$\text{Output} = \frac{W - N + 2P}{S} + 1$$

- W = Panjang/Tinggi Input
- N = Panjang/Tinggi Filter
- P = Padding
- S = Stride

Stride adalah parameter yang menyatukan beberapa jumlah pergeseran filter. Jika nilai stride adalah 1, maka convolutional filter akan bergeser sebanyak 1 pixels secara horizontal lalu vertical. Semakin kecil stride maka akan semakin detail informasi yang kita dapatkan dari sebuah input, namun membutuhkan komputasi yang lebih jika dibandingkan dengan stride yang besar. Padding atau zero padding adalah parameter yang membutuhkan jumlah pixels (berisi nilai 0) yang akan ditambahkan di setiap sisi input. Hal ini digunakan dengan tujuan untuk memanipulasi dimensi output dari convolutional layer (feature map). Dimensi output dari convolutional layer selalu lebih kecil dari inputnya (kecuali penggunaan 1 x 1 filter dengan stride 1). Padding dapat mengatur dimensi output agar tetap sama seperti dimensi input atau setidaknya tidak berkurang secara drastic.

b. Fully Connected Layer (FC Layer)

Feature map yang dihasilkan dari *feature extraction layer* masih berbentuk *multidimensional array*, sehingga harus melakukan "*flatten*" atau *reshape feature map* menjadi sebuah vektor agar bisa kita gunakan sebgaai *input* dari *fully- commected layer.*

c. Deep Learning

Deep structured learning adalah salat satu cabang dari ilmu pembelajaran mesin (*Machine Learning*) yang terdiri algoritma pemodelan abstarksi tingkat tinggi pada data menggunakan sekumpulan fungsi transformasi *non-linear* yang ditata berlapis-lapis dan mendalam. Teknik dan algoritma dalam pembelajaran dalam dapat digunakan baik untuk kebutuhan pembelajaran terarah (*supervised learning*), pembelajaran tak terarah (*unsupervised learning*) dan semi terarah (*semi- supervised learning*) dalam bebagai aplikasi seperti pengenalan citra, pengenalan suara, klasifikasi teks, dan sebagainya. *Deep leraning* disebut sebagai *deep* (dalam) karena struktur dan jumlah jaringan saraf pada algoritmanya sangat banyak bisa mencapai hingga ratusan lapisan.

Deep learning adalah salah satu jenis algoritma jaringan saraf tiruan yang menggunakan metadata sebagai input dan mengolahnya menggunakan sejumlah lapisan tersembunyi (*hidden layer*) transformasi non linear dari data masukan untuk menghitung nilai output. Algoritma pada *deep learning* memiliki fitur yang unik yaitu sebuah fitur yang mampu mengekstraksi

secara otomatis. Hal ini berarti algoritma yang dimilikinya secara otomatis dapat menangkap fitur yang relevan sebagai keperluan dalam pemecahan suatu masalah. Algoritma semacam ini sangat penting dalam sebuah kecerdasan buatan karena mampu mengurangi beban pemrograman dalam memilih fitur yang eksplisit. Dan, algoritma ini dapat digunakan untuk memecahkan permasalahan yang perlu pengawasan (*supervised*), tanpa pengawasan (*unsupervised*), dan semi terawasi (*semi supervised*).

Jenis *Deep Learning* :

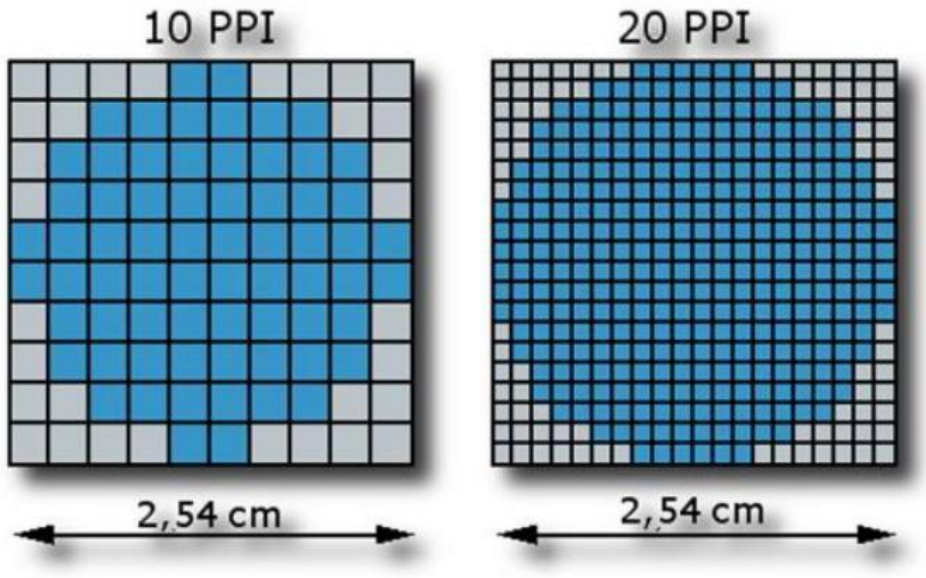
- *Deep Learning* untuk pembelajaran tanpa pengawasan (*Unsupervised Learning*): tipe ini digunakan pada saat tabel variabel target tidak tersedia dan korelasi nilai yang lebih tinggi harus dihitung dari unit yang diamati untuk menganalisa polanya.
- *Hybrid Deep Networks* (*Deep Learning* Gabungan): pendekatan tipe ini bertujuan agar dapat dicapai hasil yang baik dengan menggunakan pembelajaran yang diawasi untuk melakukan Analisa pola atau dapat juga dengan menggunakan pembelajaran tanpa pengawasan.

d. Komputer Vision

Komputer vision merupakan bidang yang mencakup metode untuk memperoleh, mengolah, menganalisis, dan memahami data visual seperti gambar dan video. Tujuan utamanya adalah agar komputer atau mesin dapat meniru kemampuan memahami dan menginterprestasikan informasi sensorik atau kemampuan intelektual untuk mencari makna yang diterima oleh panca indera mata manusia dan otak.

Bidang yang berkaitan erat dengan *komputer vision* adalah *image processing* (pengolahan citra) dan machine vision (visi mesin). Ada tumpeng tindih yang signifikan dalam berbagai terknik dan aplikasi yang mencakup tiga bidang tersebut. Hal ini menunjukan terknik dasar yang diguakan dan dikembangkan kurang lebih sama (*identic*). Komputer vision mencakup teknologi utama untuk menganalisis citra (visual) secara otomatis yang digunakan dalam bidang lain. Sedangkan machine vision biasanya mengacu pada proses menggabungkan analisis citra otomatis dengan metode atau teknologi lain baik berupa *software* maupun *hardware* untuk mencapai tujuan tertentu.

C. Pixel Per Inch (PPI)



Gambar 2. 2. PPI

Kepadatan piksel atau dalam bahasa Inggris *pixel density* yang biasanya dinyatakan dalam satuan *pixel per inch* (ppi) adalah indikasi penting yang memberitahu Anda tentang seberapa jelas layar dari sebuah perangkat elektronik.

Semakin besar nilai ppi maka bisa diartikan kepadatan atau kerapatan piksel dalam satuan area inci semakin tinggi. Hasilnya, layar yang akan Anda lihat akan semakin halus dan jelas.

Sedikit keliru jika ppi didefinisikan sebagai kepadatan piksel, ini sama dengan Anda mengartikan “cm” sebagai panjang, padahal panjang sendiri tidak hanya bisa dinyatakan dalam “cm” tetapi bisa dengan meter, km dan lain sebagainya.

Lebih tepatnya, ppi adalah satuan yang menyatakan kepadatan piksel sebuah layar. Bayangkan layar ponsel adalah sebuah grid, di mana piksel akan menempati setiap baris dan kolom di dalamnya.

Layar HD diukur dari segi jumlah piksel yang melintas sepanjang layarnya, sehingga dalam contoh Samsung Galaxy Mega 6.3 ini disebut sebagai layar 720p. Sebuah layar Full HD terdiri dari 1080 piksel sepanjang layarnya, sehingga Sony Xperia Z Ultra digambarkan memiliki tampilan Full HD 1080p.

1. Cara Menghitung PPI Sebuah Layar

Dalam istilah awam, ppi mengacu pada jumlah piksel yang tersebar di seluruh area permukaan layar, tetapi karena layar diukur secara diagonal maka tidak bisa sesederhana mengalikan jumlah pixel mendatar dengan jumlah piksel yang menurun dan kemudian membaginya dengan pengukuran diagonal.

Agar mudah dan cepat, Anda dapat menggunakan kalkulator ppi siap pakai untuk mengetahui nilai ppi dari layar apapun tanpa terlibat dalam Teorema Pythagoras. Uniknya ppi adalah, bahwa semakin besar layar maka semakin rendah ppi tersebut. Hal ini disebabkan karena layar yang lebih besar membuat tiap individu piksel menjadi lebih besar (meregang) agar dapat mengisi seluruh ruang. meskipun memiliki jumlah piksel yang sama, tetapi ketika **ukuran layarnya** berbeda maka akan berbeda pula nilai kepadatan piksel. Dengan resolusi yang sama, semakin kecil layar maka piksel akan terkonsentrasi pada area yang lebih kecil sehingga akan semakin tinggi nilai kepadatan piksel.

2. Cara Manual Menghitung ppi

Secara teoritis, ppi dapat dihitung dengan mengetahui ukuran diagonal dari sebuah layar dalam satuan inci dan resolusi layar dalam satuan piksel (lebar dan tinggi). Menghitung ppi dapat dilakukan dengan dua langkah.

a. **Menghitung resolusi diagonal dalam satuan piksel menggunakan teorema Pythagoras**

$$d_p = \sqrt{w_p^2 + h_p^2}$$

b. **Menghitung ppi dengan rumus berikut**

$$PPI = \frac{d_p}{d_i}$$

dimana:

- **dp** adalah resolusi diagonal dalam satuan piksel
- **wp** adalah resolusi lebar (horisontal) layar dalam satuan piksel
- **hp** adalah resolusi tinggi (vertikal) layar dalam satuan piksel
- **di** adalah ukuran diagonal dalam satuan inci

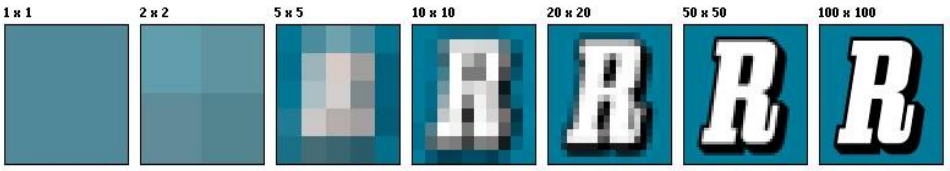
Contoh:

1. Sebuah perangkat dengan ukuran layar 21,5 inci dan resolusi Full HD (1920×1080) akan memiliki kepadatan piksel 102,46 ppi (di mana **wp** = 1920, **hp** = 1080 dan **di** = 21,5) → *akar (1920^2 + 1080^2) / 21,5* = **102,46 ppi.**
2. Perangkat dengan ukuran layar 10,1 inci dan resolusi 1024×600 piksel akan memiliki kepadatan piksel 117,5 ppi (di mana **wp** = 1024, **hp** = 600 dan **di** = 10,1) → *akar (1024^2 + 600^2) / 10,1* = **117,5 ppi.**
3. Dengan cara yang sama, perangkat dengan layar 27 inci dan resolusi 2560×1440 piksel akan menghasilkan kepadatan piksel **109 ppi**.

D. *Pixel* dan Resolusi Gambar

Pixel atau picture elements merupakan unsur gambar atau representasi sebuah titik terkecil dalam sebuah gambar grafis yang dihitung per inci. Semakin tinggi jumlah piksel dalam suatu citra, maka semakin tajam citra tersebut.

Selain piksel, terdapat pula resolusi gambar, yang menentukan kualitas suatu citra. Resolusi gambar mendeskripsikan tentang banyaknya detil gambar yang tersimpan, atau sering didefinisikan sebagai jumlah piksel dalam pencitraan gambar digital. Semakin besar nilai resolusi, maka semakin tajam gambar yang diperlihatkan seperti Gambar 2.5.



Gambar 2.3. Contoh Perbedaan Resolusi Gambar

1. *Grayscale*

Suatu citra *grayscale* adalah suatu citra yang hanya memiliki warna tingkat keabuan. Penggunaan citra *grayscale* dikarenakan membutuhkan sedikit informasi yang diberikan pada tiap *pixel* dibandingkan dengan citra berwarna. Warna abu-abu pada citra keabuan adalah warna R *(Red)*, G *(Green)*, B *(Blue)* yang memiliki intensitas yang sama. Sehingga dalam *grayscale image* hanya membutuhkan nilai intensitas tunggal dibandingkan dengan citra berwarna membutuhkan tiga intensitas untuk tiap *pixel*nya.

Pada umumnya citra *grayscale* memiliki penyimpanan 8 bit sehingga dapat menampung 256 tingkatan skala abu-abu dimana tiap *pixel* memiliki intensitas 0

hingga 255 dengan 0 menjadi hitam dan 255 menjadi putih. Contoh dari *grayscale image* dapat dilihat pada Gambar 2.6.

Gambar 2.4. Grayscale Image

Grayscale dilakukan dengan cara mengambil semua pixel pada gambar kemudian warna tiap pixel akan diambil informasi mengenai 3 warna dasar yaitu merah, hijau dan biru, ketiga warna dasar ini akan dijumlahkan kemudian dibagi tiga sehingga didapat nilai rata-rata. Nilai rata-rata inilah yang akan dipakai untuk memberikan warna pada pixel gambar sehingga warna menjadi *grayscale* seperti pada persamaan dibawah ini.

$$S = \frac{r+g+b}{3}$$

Dimana : S = Citra *grayscale*

R = nilai *red* dari sebuah piksel

G = nilai *green* dari sebuah piksel

B = nilai *blue* dari sebuah piksel

2. *Thresholding*

Thresholding merupakan teknik binerisasi yang digunakan untuk mengubah citra keabuan menjadi citra biner. *Thresholding* dapat digunakan dalam proses segmentasi citra untuk mengidentifikasi objek yang diinginkan dari *background* berdasarkan distribusi tingkat keabuan atau tekstur citra. Proses *thresholding* menggunakan nilai batas (*threshold*) untuk mengubah nilai piksel pada *grayscale image* menjadi hitam atau putih.

Citra hasil berupa citra hitam dan putih, citra hitam (0) diperoleh apabila nilai piksel citra keabuan lebih kecil dari nilai *threshold* sebaliknya citra putih (1) diperoleh jika nilai piksel citra keabuan lebih besar dari nilai *threshold*. Pada proses *thresholding* citra dibagi kedalam beberapa bagian, masing-masing dari bagian tersebut memiliki ukuran yang sama besar dengan sebuah citra dibagi dengan derajat keabuan (Aruan, 2017). Persamaan yang digunakan terdapat pada pesamaan dibawah ini.

$$g(x,y)\begin{cases}1\ if\ (x,y) > T\\0\ if\ F\ (x,y) \leq\end{cases}$$

Dimana : g(x,y) = piksel citra hasil binerisasi

f(x,y) = piksel citra asal

T = nilai *threshold*

3. *Feature Extraction*

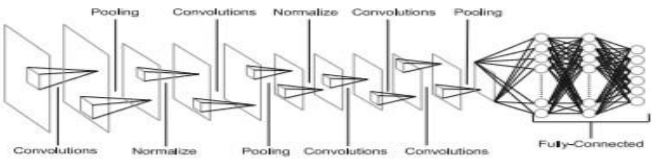
Fitur merupakan karakteristik unik dari suatu objek. Fitur dibedakan menjadi dua yaitu fitur “alami” merupakan bagian dari gambar, misalnya kecerahan dan tepi objek. Sedangkan fitur “buatan” merupakan fitur yang diperoleh dengan operasi

tertentu pada gambar, misalnya histogram tingkat keabuan. Sehingga ekstraksi fitur adalah proses untuk mendapatkan ciri-ciri pembeda yang membedakan suatu objek dari objek yang lain (Putra,2010).

4. *Convolutional Neural Network*

Convolutional Neural Network (CNN) adalah pengembangan dari *Multilayer Perceptron* (MLP) yang didesain untuk mengolah data dua dimensi yang terdiri dari lapisan input dan output, serta beberapa lapisan tersembunyi (Yao, 2019). CNN menjadi sangat populer setelah memenangkan *ImageNet Large Scale Recognition Challenge* ILSVRC 2012. Dalam makalah tersebut, mereka menggunakan lebih dari 600.000 neuron dan 7 lapisan tersembunyi untuk memberikan model data yang baik untuk menghindari overfting.

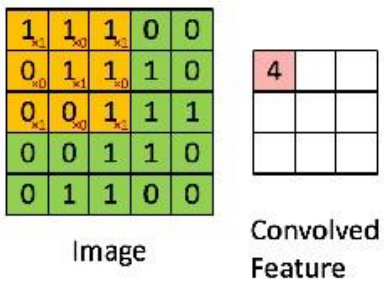
Pada CNN, setiap neuron direpresentasikan dalam bentuk dua dimensi, tidak seperti MLP yang setiap neuron hanya berukuran satu dimensi. CNN termasuk dalam *Deep Neural Network* karena kedalaman jaringan yang tinggi. *Convolutional Neural Network* (CNN) merupakan salah satu jenis neural network yang berisi kombinasi beberapa layer yaitu *convolutional layer, pooling layer* dan *fully connected layer* (Hu, Huang, Wei, Zhang, & Li, 2015) seperti pada Gambar 2.7.



Gambar 2.5 Convolutional neural network

5. *Convolutional Layer*

Konvolusi merupakan suatu istilah matematis yang berarti mengaplikasikan sebuah fungsi pada output fungsi lain secara berulang (Vista Lab, 2013) seperti Gambar 2.6.



Gambar 2.6 Operasi Konvulasi

Convolutional Layer adalah sebuah inti utama dari CNN, dimana layer ini memiliki sebuah kumpulan filter yang dapat digunakan untuk mempelajari citra masukan. Melalui layer ini, fitur akan di ekstraksi dan kemudian di lanjutkan ke layer berikutnya dengan tujuan untuk mengekstraksi fitur yang lebih kompleks (Bui & Chang, 2017).

Convolutional Layer melakukan operasi konvolusi terhadap input ataupun output dari layer sebelumnya. Tujuan dilakukannya konvolusi pada data citra adalah untuk mengekstraksi fitur dari citra input. Konvolusi akan menghasilkan transformasi linear dari data input sesuai informasi spasial pada data. Bobot pada layer tersebut menspesifikasikan kernel konvolusi yang digunakan, sehingga kernel konvolusi dapat dilatih berdasarkan input pada CNN seperti pada persamaan rumus.

$$C_{i,j} = A \text{ x } P_1 + B \text{ x } P_2 + C \text{ x } P_3 + D \text{ x } P_4 + E \text{ x } P_5$$

$$+ F \text{ x } P_6 + G \text{ x } P_7 + H \text{ x } P_8 + I \text{ x } P_9$$

Keterangan :

$C_{i,j}$ = Nilai *feature map* yang dihasilkan proses konvolusi

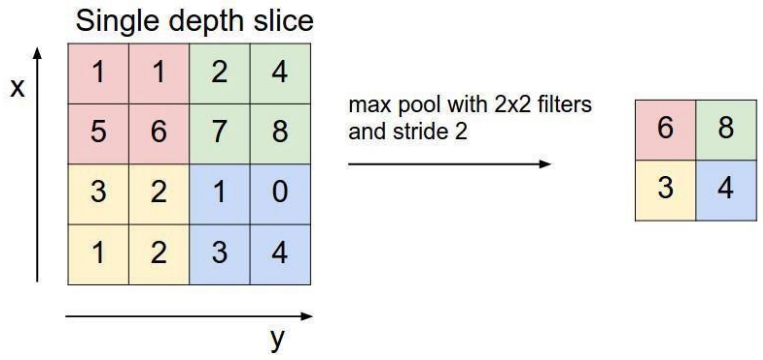
A-I = Nilai matriks gambar *input*

$P_{1\text{-}9}$ = Nilai matriks *feature detector*

6. *Pooling Layer*

Pooling Layer merupakan proses resizing yaitu proses untuk mengubah ukuran citra input yang berbeda,salah satunya dengan menggunakan operasi *MAX*. Hal ini bertujuan untuk membantu mengurangi jumlah parameter dan waktu perhitungan yang dibutuhkan saat melatih network.

Max pooling membagi output dari *convolutional layer* menjadi beberapa grid kecil lalu mengambil nilai maksimal dari setiap grid untuk menyusun matriks citra yang telah direduksi seperti yang ditunjukkan pada Gambar. Grid yang berwarna merah, hijau, kuning dan biru merupakan kelompok grid yang akan dipilih nilai maksimumnya. Sehingga hasil dari proses tersebut dapat dilihat pada kumpulan grid disebelah kanannya. Proses tersebut memastikan fitur yang didapatkan akan sama meskipun objek citra mengalami translasi (pergeseran).



Gambar 2.7 Operasi Max Polling

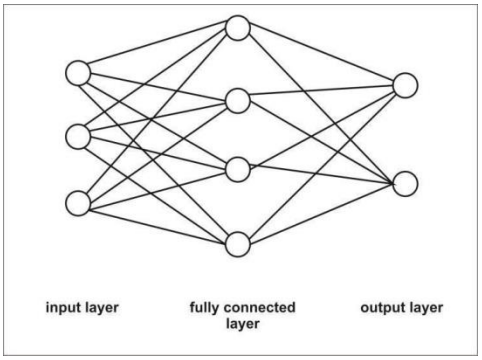
7. *Fully Connected Layer*

Dan layer ketiga pada CNN adalah Fully Connected Layer, dimana layer ini mengambil seluruh neuron pada layer sebelumnya (Convolutional Layer dan MAX Pooling Layer) dan menghubungkannya ke setiap single neuron yang ada (Devikar, 2016).

Fully connected layer adalah layer yang biasanya digunakan dalam penerapan MLP dan bertujuan untuk melakukan transformasi pada dimensi data agar data dapat diklasifikasikan secara linear (Suartika et al., 2016).

Setiap neuron pada convolution layer perlu ditransformasi menjadi data satu dimensi terlebih dahulu sebelum dapat dimasukkan kedalam sebuah fully connected layer. Karena hal tersebut menyebabkan data kehilangan informasi spesialnya dan tidak reversibel, *fully connected layer* hanya dapat diimplementasikan diakhir jaringan seperti pada Gambar.

Dalam sebuah jurnal oleh Lin et al. (2014), dijelaskan bahwa *convolutional layer* dengan ukuran kernel 1 x 1 melakukan fungsi yang sama dengan sebuah *fully connected layer* namun dengan tetap mempertahankan karakter spasial dari data. Hal tersebut membuat penggunaan *fully connected layer* pada CNN sekarang tidak banyak dipakai.



Gambar 2.8 Proses Fully Connected Layer

E. Pengenalan Jenis Mobil

1. Jenis mobil Sedan

Berdasarkan Ballast, bentuk merupakan raut (*shape)* dan konfigurasi dasar dari sebuah objek. Ballast menyatakan bahwa bentuk seringkali merupakan cara pertama kali seseorang membedakan sebuah objek dengan lainnya. Jenis mobil sedan atau sering juga disebut saloon. Jenis mobil sedan ini terbilang sangat familiar diseluruh dunia. Hampir seluruh orang mengenal jenis mobil sedan ini. Jenis mobil sedan adalah mobil yang memiliki 3 pilar utama untuk kabin penumpang (pilar A,B,C) serta dilengkapi dengan kompartemen mesin dibagian depan dan kompartemen bagasi dibagian belakang.

Pada dasarnya, mobil sedan mempunyai kapasitas 5 orang penumpang, 2 dibagian depan, dan 3 dibagian belakang. Selain itu, jumlah pintu yang dimiliki umumnya ada 5, 4 pintu khusus untuk akses keluar masuk penumpang, dan 1 pintu khusus untuk memasukkan barang ke bagasi.

Jenis mobil sedan yang digunakan untuk *sample training*, mobil dipilih berdasarkan perbedaan bentuk mobil dari berbagai sisi yaitu depan, samping dan belakang.

Gambar 2.9 *Sample Training* Jenis Sedan

2. Jenis mobil MPV

Multi Purpose *Vehicle* (MPV) adalah sebuah mobil multi fungsi dengan dua kompartemen, satu kompartemen untuk mesin, satu untuk kabin penumpang dengan pintu bagasi yang menyatu dengan kaca belakang. Bagian untuk penumpang terdiri dari 3 baris tempat duduk dengan kapasitas 7 orang. Bahkan pada beberapa kelas MPV yang lebih mewah bisa mengangkut lebih dari itu. Kegunaan utama adalah untuk mengangkut penumpang dan barang.

MVP yang cukup populer di Indonesia tersedia dalam berbagai ukuran. MVP yang cukup besar (muat hingga 7 orang) misalnya Daihatsu Xenia, Mitsubishi Xpander, Toyota Avanza. Sedangkan MPV yang berukuran kecil seperti Datsun

Go, Daihatsu Sigra dan Picanto. Pada Gambar 2.10 merupakan *sample training* citra yang dipilih untuk jenis mobil MPV.

Gambar 2.10 *Sample Training* Jenis MPV

3. Jenis mobil SUV

Jenis mobil *Sport Utility Vehicle* atau akrab disebut SUV ini merupakan mobil yang menggabungkan utilitas mobil jeep, pikap dengan mobil sedan yang elegan. Prioritas utama dari jenis mobil SUV ini adalah kemampuannya yang bisa digunakan diberbagai medan berat namun dengan kenyamanan layaknya mobil sedan.

SUV adalah sebuah jenis mobil penumpang dengan dua kompartemen, satu kompartemen untuk mesin, satu untuk kabin penumpang yakni mobil penumpang

yang dibangun diatas kerangka truk ringan. Rata-rata, SUV memiliki kapasitas bagasi sebesar 500 hingga 600 liter dengan kondisi kursi baris kedua tegak. Mobil yang mampu berjalan di “dua alam” ini mampu menampung 5-7 orang penumpang dan memiliki 2 baris kursi. Kegunaan utama adalah untuk mengangkut penumpang pada medan yang lebih berat dari jalan biasa. Pada Gambar 2.11 adalah contoh *sample training* mobil SUV.

Gambar 2.11 *Sample Training* Jenis SUV

F. *Object Tracking*

Dalam visi komputer, objecttracking (pelacakan objek) adalah suatu proses untuk melacak satu objek atau lebih dari suatu citra. Objecttracking termasuk dalam salah satu fungsi yang sangat penting di bidang visi komputer. Ada tiga langkah penting dalam analisa video: deteksi objek yang bergerak, mendeteksi beberapa

objek di setiap frame, dan analisa objek yang dilacak untuk mengenali pergerakaan objek pada citra.Dalam bentuk yang paling sederhana, tracking dapat didefinisikan sebagai suatu masalah untuk memperkirakan lintasan dari sebuah objek yang bergerak dalam sebuah gambar. Secara konsisten, pelacak memberikan label pada objek yang dilacak pada frame-frame yang berbeda dalam sebuah video. Berdasarkan dari pelacakan domain, sebuah pelacak juga dapat memberikan informasi suatu objek, seperti sebuah orientasi gerak, area, atau bentuk dari objek. Satu hal yang dapat memudahkan pelacakan, yaitu dengan memberikan batasan secara paksa pada pergerakan dan/atau bentuk dari objek. Sebagai contoh, hampir semua algoritma tracking berasumsi bahwa objek bergerak secara tetap tanpa ada perubahan yang mendadak serta memaksakan kecepatan atau percepatan objek menjadi konstan berdasarkan informasi sebelumnya. Pembelajaran sebelumnya mengenai ukuran, tampilan, dan bentuk objek, juga dapat digunakan untuk memudahkan masalah

G. Pengertian Aplikasi

Aplikasi menurut Jogiyanto (2018:12) adalah penggunaan dalam suatu komputer, instruksi (instructiom) atau pernyataan (statement) yang disusun sedemikian rupa sehingga komputer dapat memprosesinput menjadi output . Menurut Kamus Kamus Besar Bahasa Indonesia (2018 : 52) , “Aplikasi adalah penerapan dari rancang sistem untuk mengolah data yang menggunakan aturan atauketentuan bahasa pemrograman tertentu”. Aplikasi adalah suatu program komputer yang dibuat untuk mengerjakan danmelaksanakan tugas khusus dari pengguna. Aplikasi merupakan rangkaian kegiatan atauperintah untuk dieksekusi

oleh komputer. Program merupakan kumpulan instruction set yang akan dijalankan olehpemroses, yaitu berupa software. Bagaimana sebuah sistem komputer berpikir diatur olehprogram ini. Program inilah yang mengendalikan semua aktifitas yang ada pada pemroses. Program berisi konstruksi logika yang dibuat oleh manusia, dan sudah diterjemahkan ke dalam bahasa mesin sesuai dengan format yang ada pada instructionset .Program aplikasi merupakan program siap pakai. Program yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain. Contoh-contoh aplikasiialah program pemproses kata dan Web Browser. Aplikasi akan menggunakan sistemoperasi (OS) komputer dan aplikasi yang lainnya yang mendukung.Istilah ini mulai perlahan masuk ke dalam istilah Teknologi Informasi semenjak tahun 1993, yang biasanya juga disingkat dengan app. Secara historis, aplikasi adalah software yang dikembangkan oleh sebuah perusahaan. App adalah sofware yang dibeli perusahaan dari tempat pembuatnya. Industri PC tampaknya menciptakan istilah ini untuk merefleksikan medan pertempuran persaingan yang baru, yang paralel dengan yang terjadi antar sistem operasi yang dimunculkan.

H. *Java*

Menurut Shalahuddin (2008:1), *java* adalah bahasa pemrograman yang berorientasi objek (OOP) dan dapat dijalankan pada berbagai *platform* sistem operasi. Perkembangan *Java* tidak hanya terfokus pada satu sistem operasi, tetapi dikembangkan untuk berbagai sistem operasi dan bersifat *open source.*

1. Sebagian Fitur dari *Java*

a. *Java Virtual Machine (JVM)*

Menurut Shalahuddin (2008:10), *JVM* adalah sebuah mesin imajiner*(maya)* yang bekerja dengan menyerupaiaplikasi pada sebuah mesin nyata. *JVM* menyediakan spesifikasi *hardware* dan *platform* dimana kompilasi kode *Java* terjadi. Spesifikasi inilah yang membuat aplikasi berbasis *Java* menjadi bebas dari *platform* manapun karena proses kompilasi diselesaikan oleh JVM.Aplikasi program *Java*diciptakan dengan *file* teks berekstensi.*Java.* Program ini dikompilasimenghasilkan satu berkas *bytcode* berekstensi *.class* atau lebih. *Bytecode* adalah serangkaian instruksi serupa instruksi kode mesin. Perbedaannya adalah kode mesin harus dijalankan pada sistem komputer dimana kompilasi ditujukan, sementara *bytecode* berjalan pada *Java interpreter* yang tersedia di semua *platform* sistem komputer dan sistem operasi.

b. *Garbage Collection*

Banyak bahasa pemrogramanlain yang mengijinkan seorang programmer mengalokasikan memori pada saat dijalankan. Program *Java* melakukan *garbage collection* yang berarti program tidak perlu menghapus sendiri objek–objek yang tidak digunakan lagi.

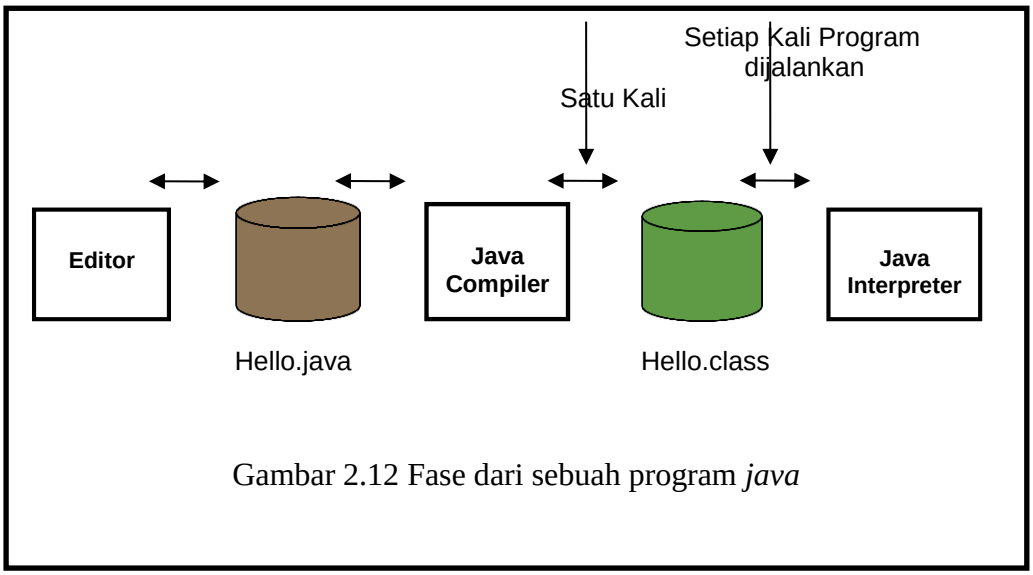
c. *Code Security*

*Code Security*terimplementasipada *Java* melalui penggunaan *JavaRuntime Environment(JRE). Java* menggunakan model pengamanan 3 lapis untuk melindungi sistem dari *untrusted Java* Code.

1. Pertama, *class-loader* menangani pemuatan kelas *Java* ke runtime *interpreter*. Proses ini menyediakanpengamanan dengan memisahkan kelas–kelas yang berasal dari *localdisk* dengan kelas–kelas yangdiambil dari jaringan. Hal ini membatasi aplikasi Trojan karena kelas–kelas yang berasal dari *localdisk* yang dimuat terlebih dahulu.
2. Kedua, *bytecode verifier* membaca *bytecode* sebelum dijalankan danmenjamin *bytecode* memenuhi aturan–aturan dasar bahasa *Java.*
3. Ketiga, manajemen keamanan menangani keamanan tingkat aplikasi dengan mengendalikan apakah program berhak mengakses sumber daya seperti sistem *file, port* jaringan, proses eksternal dan sistem *windowing*.

d. Fase-fase pemrogram *Java*

Gambar di bawah ini menjelaskan aliran proses kompilasi dan eksekusi sebuah program *Java*:



Gambar 2.12 Fase dari sebuah program *java*

1. *Editor* merupakan aplikasi untukmembuat atau mengedit aplikasi *java*.

2. *Hello.java* merupakan *file java* yang sudah mempunyai kode-kode *java* dan di simpan dengan extensi *java*.
3. *Java compiler* merupakanmerubah *file java* yang extensi *java* menjadi *file* yang extensi *class*.
4. *Hello.class* merupakan hasil *file java compiler* yang asalnyaberupa *file extensi java*.
5. Java interpreter merupakan menterjemahkan baris per baris kode ke dalam bahasa mesin dan menampilkan hasilnya.

I. Android

Irawan 2018. Android adalah sebuah sistem operasi untuk perangkat mobile berbasis linuk yang mencakup sistem operasi, *middleware* dan aplikasi. Androidmenyediakan platform yang terbuka bagi para pengembang untuk menciptakan aplikasi baru. Android merupakan generasi baru *platformmobile*, *platform* yang memberikanpengembangan untuk melakukan pengembangan sesuai dengan yang diharapkan. Pada saat perilisan perdana android, 5 November 2007, android bersama *open handset alliance* menyatakan mendukung pengembangan *open source* pada perangkat lunak *mobile*. Di lain pihak, google meriliskode-kode android dibawah lisensi *apache*, sebuah lisensi perangkat lunakdan *open platform* perangkat seluler. Safaat (2011:1).

1. Android SDK (*SoftwareDevelopment Kit*)

Android SDK adalah tools API (*Application Programming Interface*) yang

diperlukan untuk memulai pengembangan aplikasi pada *platform*Android menggunakan bahasa pemrograman *Java*. Android merupakan subset perangkat lunak untuk ponsel yang meliputi, sistem operasi, *middleware* dan aplikasi yang di *release* oleh *google*. Saat ini disediakan android SDK (*Software Development Kit*) sebagai alat bantu dan API untuk mulai mengembangkan aplikasi platform android menggunakan bahasa pemrograman *java*. Sebagai platform aplikasi-netral, android memberi anda kesempatan untuk membuat aplikasi yang kita butuhkan yang buka merupakan aplikasi bawaan *handphone*. Safaat.

J. Database SQLite

Menurut Jay A. Kreibich (2018,12) SQLite merupakan paket perangkat lunak yang bersifat public domain yang menyediakan sistem manajemen basis data relasional atau RDBMS. Sistem basis data relasional digunakan untuk menyimpan record yang didefinisikan oleh pengguna pada ukuran tabel yang besar dan memproses perintah query yang kompleks dan menggabungkan data dari berbagai tabel untuk menghasilkan laporan dan rangkuman data. Kata ‘Lite’ pada SQLite tidak menunjuk pada kemampuannya, melainkan menunjuk pada sifat dari SQLite, yaitu ringan ketika dihubungkan dengan kompleksitas pengaturan, administrative overhead, dan pemakaian sumber. SQLite memiliki fitur-fitur sebagai berikut :

1. Tidak memerlukan server Arsitektur SQLite tidak memiliki arsitektur client server. Kebanyakan sistem database skala besar memiliki paket server yang besar yang membentuk mesin database.
2. Single File Database SQLite mengemas seluruh database ke dalam suatu single file. Single file tersebut berisi layoutdatabase dan data aktual yang

berada pada tabel dan indeks yang berbeda. Format file dapat digunakan pada banyak platform dan dapat diakses pada mesin manapun tanpa memperhatikan native byte order ataupun ukuran kata. Pengemasan database kedalam suatu file tunggal memudahkan pengguna untuk membuat, menyalin, ataupun mem-backupimage database yang berada di dalam media penyimpanan

3. Zero Configuration SQLite tidak membutuhkan apapun untuk melakukan instalasi dan konfigurasi. Dengan mengeliminasi server dan menggabungkan database secara langsung ke dalam aplikasi, maka pengguna tidak perlu mengetahui bahwa mereka sedang menggunakan database.
4. Embedded Device Support Ukuran code dari SQLite bersifat kecil dan penggunaan sumber daya yang konservatif membuatnya cocok digunakan untuk embedded system yang berjalan terbatas pada sistem operasi.
5. Fitur-fitur yang unik SQLite menggunakan sistem dengan tipe dinamis untuk tabel-tabel. SQLite memungkinkan pengguna untuk memasukkan nilai ke dalam kolom tanpa memperhatikan tipe data. Pada beberapa cara pemakaiannya, sistem yang bertipe dinamis pada SQLite mirip dengan sistem yang ditemukan pada bahasa scripting yang populer, yang sering memiliki sebuah tipe skalar yang dapat menerima semua tipe data dari integer sampai string. Fitur lainnya adalah kemampuan untuk memanipulasi lebih dari satu basis data pada satu waktu. SQLite mempunyai kemampuan dalam menghubungkan sebuah koneksi database tunggal dengan banyak file

basis data secara bersamaan. Hal ini memungkinkan SQLite umtuk memproses SQL statement yang menjembatani beberapa basis data sekaligus.

6. Compatible License SQLite dan SQLite code tidak memiliki lisensi pengguna dan tidak dilindungi oleh GNU's Not Unix (GNU) General Public License (GPL) atau lisensi open source sejenisnya. Hal ini berarti pengguna dapat melakukan apapun dengan source code SQLite, sehingga library code dapat digunakan dengan berbagai cara, dimodifikasi dengan berbagai cara dan didistribusikan dengan berbagai cara.
7. Highly reliable Sejumlah tes telah dilakukan sebelum library SQLite masing-masing dirilis. Hal ini dilakukan untuk mempertahankan tingkat kehandalan yang tinggi.

K. UML (Unified Modeling Language)

Menurut Adi nugroho dalam buku “Rekayasa Perangkat Lunak Beriorentasi Objek” (2017:6) UML (Unified Modeling Language) adalah ‘bahasa’ pemodelan untuk system atau perangkat lunak yang beerparadigma ‘beriorentasi objek’. Pemodelang (Modeling) sesungguhnya digunakan untuk penyederhanaan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajari dan dipahami.

a. Use Case Diagram

Use case diagram bersifat statis, memperlihatkan himpunan use-case dan actor-aktor. Diagram ini sangat penting terutama untuk memodelkan ataupun mengorganisasikan perilkau dari system yang dibutuhkan pengguna.

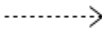
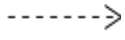
b. Activity Diagram

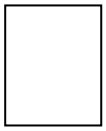
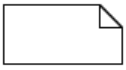
Activity Diagram bersifat dinamis, merupakan tipe khusus dari diagram state yang memperlihatkan aliran dari suatu aktivitas ke aktivitas lainya dalam suatu system.

c. Class Diagram

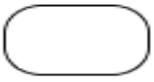
Class Diagram bersifat statis tetapi sering pula memuat kelas-kelas, antarmuka-antarmuka, kolaborasi-kolaborasi, serta relasi-relasi.

Tabel 2.1 Simbol Diagram *Use Case*

NO	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
2		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri*(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*).
3		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
4		*Include*	Menspesifikasikan bahwa *use case* sumber secara *eksplisit*.
5		*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use*

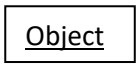
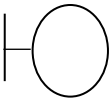
NO	GAMBAR	NAMA	KETERANGAN
			case sumber pada suatu titik yang diberikan.
6		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
8		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor
9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi

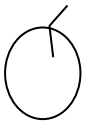
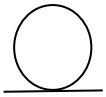
Tabel 2.2 Simbol Diagram *Activity*

NO	GAMBAR	NAMA	KETERANGAN
1		*Actifity*	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain
2		*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi

NO	GAMBAR	NAMA	KETERANGAN
3		*Initial Node*	Bagaimana objek dibentuk atau diawali.
4		*Actifity Final Node*	Bagaimana objek dibentuk dan dihancurkan
5		*Fork Node*	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran

Tabel 2.3 Simbol Sequence Diagram

No.	GAMBAR	NAMA	KETERANGAN
1		*Object (Partisipan)*	Object atau biasa juga disebut partisipan merupakan instance dari sebuah class dan dituliskan tersusun secara horizontal. Digambarkan sebagai sebuah class (kotak) dengan nama objek didalamnya yang diawali dengan sebuah titik koma.
2		*Actor*	Actor jug dapat berkomunikasi dengan object, maka actor juga dapat diurutkan sebagai kolom.
3		*Life line*	Life line mengidentifikasikan keberadaan sebuah object dalam basis waktu. Notasi untuk lifeline adalah garis putus-putus vertical yang ditarik oleh sebuah object.
4		*Collaboration*	Activication dinotasikan sebagai kotak segi empat yang digambar pada sebuah lifeline action mengidentifikasikan sebuah object yang akan melakukan sebuah aksi.
5		*Boundary*	Boundary terletak diantara system dengan dunia sekelilingnya. Semua form, laporan-laporan, antar muka ke

			perangkat keras seperti printer atau scanner dan antar muka ke system lainya adalah termasuk dalam kategor.
6		*Control*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri
7		*Entity*	Entity digunakan menangani informasi yang munkin akan disimpan secara permanen. Entity bias juga merupakan sebuah table.

BAB III

METODE PENELITIAN

A. Jenis Penelitian

Jenis penelitian yang digunakan adalah penelitian deskriptif dimana memberikan gambaran mengenai apa yang sesungguhnya terjadi. Dalam pembuatan Proposal ini digunakan metode deskripsif yang menggambarkan fakta-fakta dan informasi secara sistematis, faktual dan akurat.

Penelitian ini dilakukan melalui internet yang dapat memberikan sumber data dan pengetahuan mengenai sistem yang diteliti, kemudian mencocokkan dengan kemungkinan yang terjadi dalam usaha penyelesaian masalah.

B. Waktu dan Tempat Penelitian

Adapun waktu dan tempat penelitian yang dilaksanakan pada bulan juni tahun 2021

C. Metode Pengumpulan Data

Untuk memperoleh data-data yang dibutuhkan dalam rangka melakukan penelitian, maka penulis mengumpulkan data melalui beberapa cara yaitu :

1. Analisis Data

   Menganalisa data-data yang sebelumnya telah dikumpulkan.

2. Perancangan Program

   Sebagai pedoman dalam penulisan program atau kode-kode agar berjalan sesuai rencana.

3. Uji Coba Program

   Pengujian program dilakukan untuk memastikan bahwa program yang dibuat dapat berjalan dengan baik.

4. Evaluasi

   Sistem yang telah selesai dibangun perlu adanya evaluasi untuk menemukan kelemahan yang terdapat pada program yang telah dibangun tadi, yang nantinya bisa digunakan sebagai acuan untuk memperbaiki program sehingga lebih sempurna.

D. Jenis Data

Data yang digunakan dalam penelitian ini berupa data-data yang telah dikumpulkan melalui Penelitian Pustaka (*Library Research*). Adapun jenis data primer dan data sekunder yang relevan dengan masalah yang akan dibahas.

a. Data primer

   Data Primer adalah data yang berasal atau data yang diperoleh langsung dari sumber data dan pengetahuan.

b. Data Sekunder

   Data sekunder adalah data yang diperoleh tidak secara langsung dari objek penelitian. Peneliti mendapatkan data yang sudah jadi dari internet, website dan jurnal.

E. Tahapan Penelitian

Tahapan penelitian yang dimaksud dalam penelitian ini ada beberapa tahapan yaitu persiapan penelitian, pengumpulan data, analisis perancangan, pengujian dan implementasi. Adapun Uraian dari tahapan tersebut adalah sebagai berikut :

1. Persiapan Penelitian

   Pada tahapan ini peneliti melakukan persiapan penelitian. Persiapan penelitian yang dimaksud adalah menyiapkan buku-buku, artikel-artikel tentang topik penelitian serta software yang digunakan selama penelitian.

2. Pengumpulan Data

   Pada tahap ini peneliti melakukan observasi dengan peninjauan, pencatatan dan pengamatan langsung di tempat penelitian.

3. Analisis

   Pada tahap analisis, peneliti melakukan analisa terhadap sistem yang di terapkan sekarang berdasarkan kemudian merumuskan masalah yang menjadi pokok penelitian sehingga dapat dibuat alternatif pemecahan masalah.

4. Perancangan

   Peneliti kemudian merancang aplikasi yang ingin dibuat berdasarkan alternatif pemecahan masalah.

5. Pengujian

   Setelah melakukan perancangan, peneliti kemudian menguji hasil perancangan yang telah dibuat. Jika hasil perancangan terdapat kekurangan atau kelemahan maka kembali ke tahap analisis.

6. Implementasi

   Setelah pada perancangan tidak terdapat kekurangan maka aplikasi siap untuk di gunakan oleh user.

F. Metode Pengujian Aplikasi

Beberapa *test-case* harus dilaksanakan dengan beberapa perbedaan strategi, query, atau jalur navigasi yang mewakili penggunaan sistem yang *tipical,* kritis atau abnormal. Isu kunci pada pengembangan sistem adalah pemilihan sekelompok *test-case* yang cocok, sekecil dan secepat mungkin, untuk meyakinkan perilaku sistem secara detail. Pengujian harus mencakup *unit testing*, yang mengecek validasi dari prosedur dan fungsi-fungsi secara independen dari komponen sistem yang lain. Kemudian modul testing harus menyusul dilakukan untuk mengetahui apakah penggabungan beberapa unit dalam satu modul sudah berjalan dengan baik, termasuk eksekusi dari beberapa modul yang saling berelasi, apakah sudah berjalan sesuai karakteristik sistem yang diinginkan.

Jika struktur kendali antar modul sudah terbukti bagus, maka pengujian yang tak kalah pentingnya adalah pengujian unit. Pengujian unit digunakan untuk menguji setiap modul untuk menjamin setiap modul menjalankan fungsinya dengan baik. Ada 2 metode untuk melakukan unit testing, yaitu :

1. White Box Testing

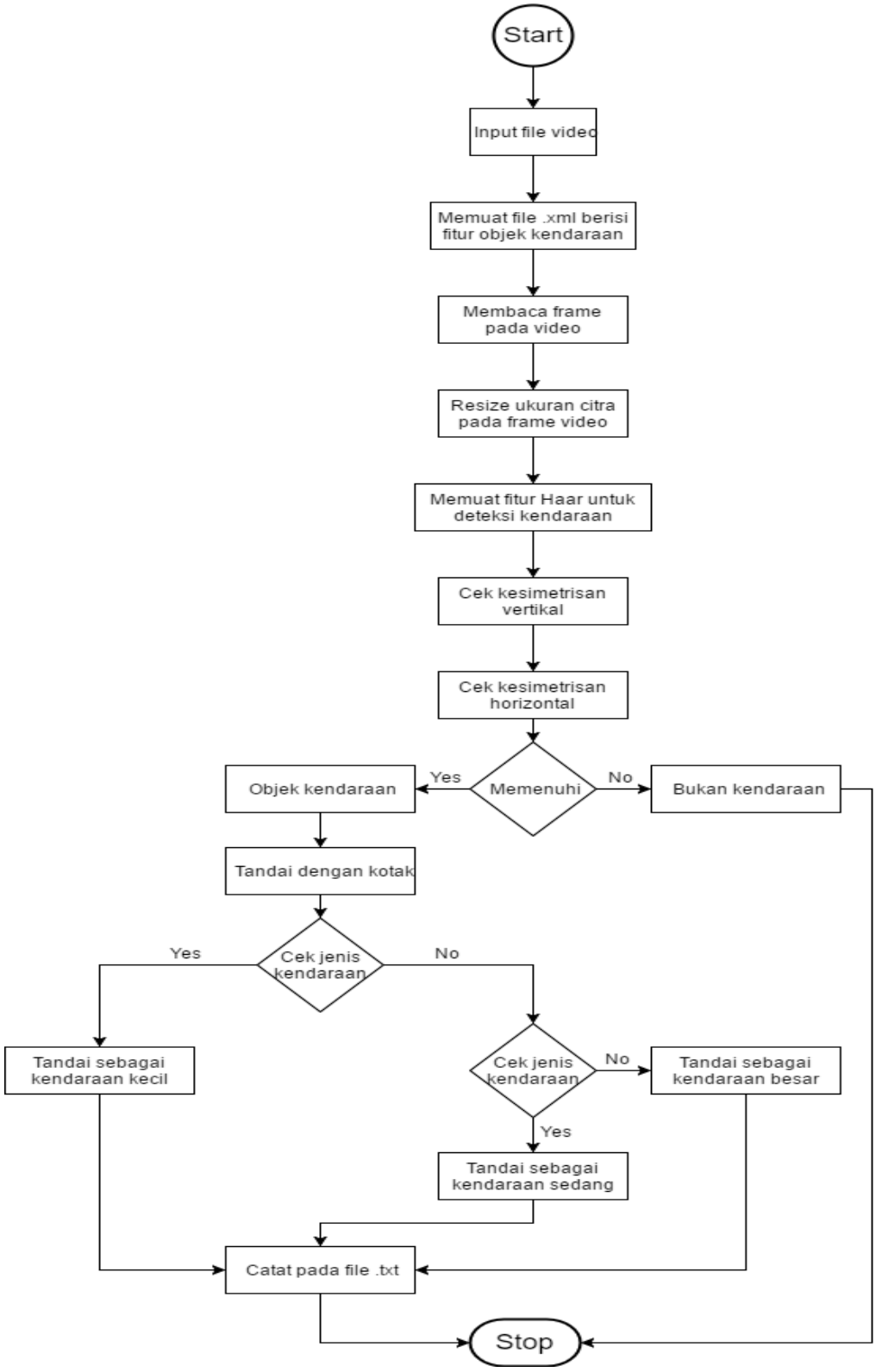
   Uji coba *white box testing* merupakan metode perancangan *testcase* yang mengunakan struktural untuk mendapatkan *testcase,* test ini digunakan untuk meramal cara kerja perangkat lunak secara rinci kepada *logic path* (jalur logika), perangkat lunak di tes dengan kondisi dan perulangan secara fisik.

Contoh pengujian white box testing ini merupakan peringatan ketika user menginputkan password user yang salah, untuk kesalahan semacam ini akan memberikan suatu informasi kepada user mengenai kesalahan yang di lakukan.

2. Black Box

Berdasarkan hasil uji coba yang dilakukan, seluruh navigasi dan tombol fasilitas program lainnya serta proses yang di jalankan tidak terjadi kesalahan, tetapi aplikasi mempunyai aturan-aturan yang sudah di tetapkan dan harus di ikuti karena apabila di hiraukan maka sistem akan menolak perintah yang tidak sesuai seperti kesalahan ketika user belum menginput data yang harusnya di input sesuai ketentuan sistem yang di jalankan dan sistem memberikan informasi kepada user karena data yang ingin diproses belum lengkap atau tidak memenuhi ketentuan untuk proses selanjutnya.

G. Desain Sistem

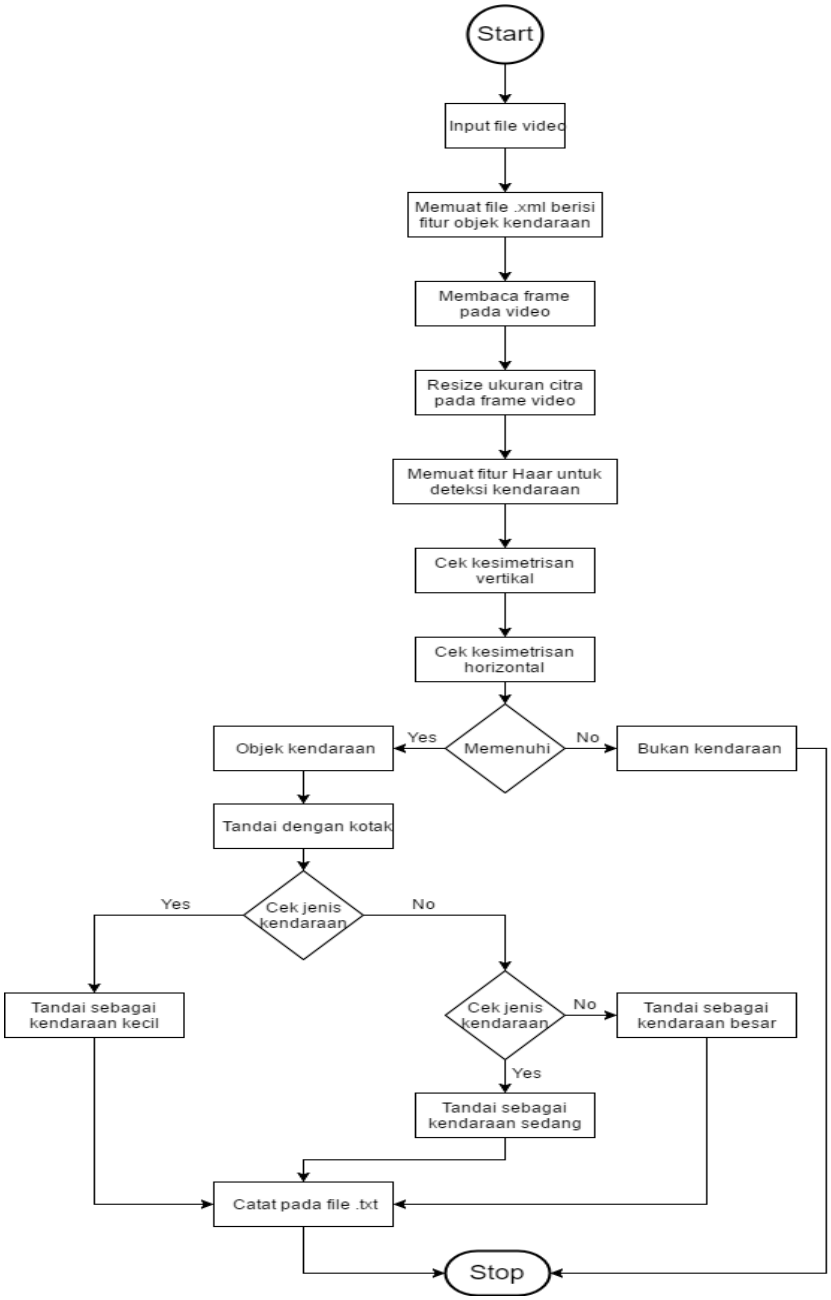


Untuk menggunakan program CarDetection, langkah pertama adalah memasukkan file video ke program. Proses ini dilakukan dengan cara menuliskan nama file video beserta ekstensinya (misal center.mp4). Kemudian program akan memuat file XML yang berisi fitur-fitur objek kendaraan. File XML ini merupakan *template* yang digunakan untuk menetapkan apakah sebuah objek merupakan kendaraan atau bukan. Lalu, program akan membaca video secara *frame-byframe*. Pembacaan ini dilakukan pada wilayah deteksi yang telah ditetapkan yaitu di antara dua garis biru.

BAB IV

HASIL DAN PEMBAHASAN

A. Desain Sistem yang di Usulkan

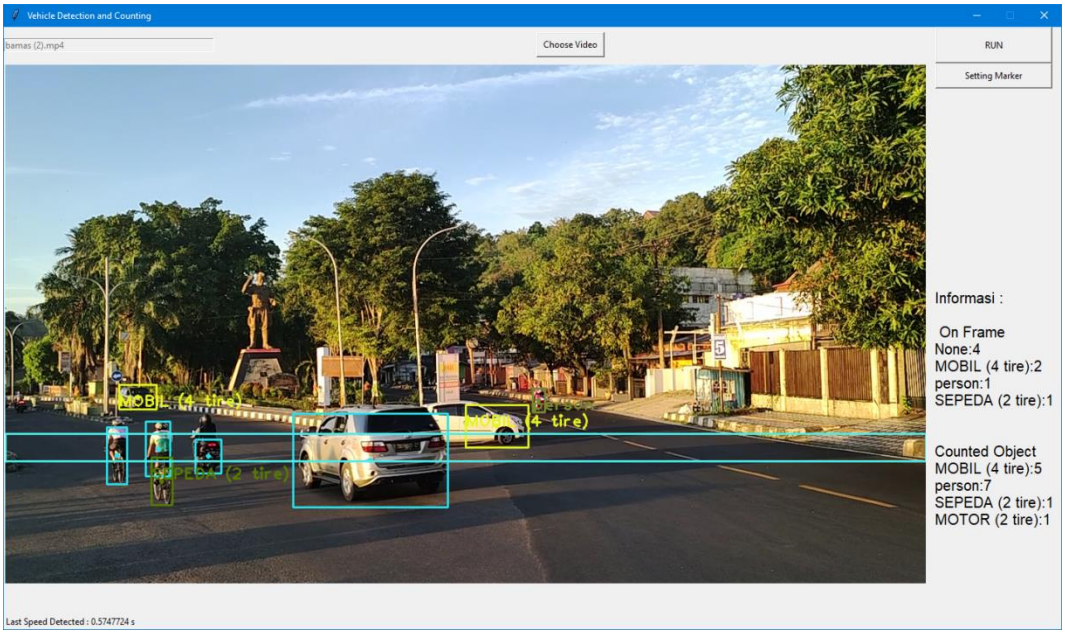


Gambar 4.1. Gambaran umum system yang di usulkan

Untuk menggunakan program CarDetection, langkah pertama adalah memasukkan file video ke program. Proses ini dilakukan dengan cara memilih video (misal center.mp4). Kemudian program akan memuat file VIDEO yang berisi fitur-fitur objek kendaraan. File VIDEO ini merupakan *template* yang digunakan untuk menetapkan apakah sebuah objek merupakan kendaraan atau bukan. Lalu, program akan membaca video secara *frame-byframe*. Pembacaan ini dilakukan pada wilayah deteksi yang telah ditetapkan yaitu di antara dua garis yang sudah di buat tadi.

B. Detail Aplikasi

Proses anotasi citra menggunakan software LabelImg. Pelabelan ini menggunakan anotasi dalam format anotasi YOLO. Hasil dari anotasi tersebut adalah data yang terdapat informasi letak kotak pembatas dan labelnya dalam bentuk .txt. Pada .txt file terdapat baris file yang memiliki format <*object-class*> <*x_center*> <*y_center*> <*width*> <*height*>, dimana pada <*object-class*> merupakan bilangan bulat yang menyatakan kelas objek, <*x_center*> dan <*y_center*> adalah koordinat pusat persergi kotak pembatas, <*width*> dan <*height*> adalah nilai *float* relatif terhadap dimensi gambar. Hasil dari .txt file dapat dilihat pada Gambar 4.2.



Gambar 4.2 tampilan utama

C. *Training Data*

Setelah anotasi selesai dilakukan, langkah selanjutnya melakukan proses *training*. Proses ini bertujuan untuk melatih komputer dengan cara mengolah gambar dan anotasi yang sudah dibuat sehingga terbentuk pola atau karakteristik dari setiap kelas yang akan menjadi bahan pertimbangan komputer dalam mencapai sebuah keputusan atau prediksi. Pada bagian ini menggunakan *pre-trained weights* YOLOv3. Dengan menggunakan teknik *transfer learning*. *Transfer learning* adalah suatu teknik yang menggunakan model yang telah di-*training* sebelumnya (*pre-trained model*) yang dapat digunakan untuk klasifikasi dataset baru tanpa harus melakukan *training* data dari awal. Proses *transfer learning* pada Darknet menggunakan data file, cfg file, dan *pre-trained weigths*. Data file berisi lokasi gambar yang digunakan untuk *train* dan *test*. Cfg file berisi bentuk jaringan yang

digunakan untuk *training*, dan *pre-trained weights* berisi model *weight* yang telah dilatih sebelumnya pada jaringan YOLO.

Karena proses komputasi yang berat saat melakukan proses *training*, maka penulis menggunakan *Google Colaboratory* yang merupakan sebuah platform yang telah disediakan oleh *Google*. Ketika proses *training* sedang berjalan, dibutuhkan koneksi internet yang stabil agar runtime saat proses *training* tidak terputus. Proses *training* ini dilakukan menggunakan *framework neural network Darknet* dengan konfigurasi seperti pada Tabel 4.1.

Tabel 4.1 Konfigurasi pada Darknet

Jenis Konfigurasi	**Keterangan**
Load model	Darknet
Load weight	Yolov3
OPENCN	1
GPU	1
CUDNN	1

Proses training menggunakan Darknet-53 sebagai load model dengan susunan layer. dan YOLOv3 sebagai load weight dengan konfigurasi seperti pada Tabel 4.2. Jumlah batch menentukan jumlah gambar yang akan diproses sebelum *network weight* mengalami pembaharuan. *Subdivision* bertujuan untuk memproses sebagian kecil batch size sekaligus pada GPU. Max_batch merupakah batas iterasi dalam melakukan *training* yang didapatkan dengan persamaan. Ketika iterasi sudah mencapai 10000, maka proses *training* akan otomatis berhenti. Nilai *max_batches* ditentukan pada persamaan 4.1.

Max_*Batches* = *Jumlahclass*.2000 (4.1)

Nilai step didapatkan pada persamaan 4.2.

Steps = (80%max_*batches*), (90%max_*batches*) (4.2)

Height dan Weight merupakan dimensi gambar masukan yang akan dilatih. Classes merupakan jumlah kelas yang akan dideteksi. Nilai filter didapatkan pada persamaan 4.3.

Filter = (*jumlahclass* + 5) x 3 (4.3)

Tabel 4.2 Konfigurasi pada *weights* YOLOv3

Jenis Konfigurasi	**Keterangan**
Batch	64
Subdivisions	16
Width	416
Height	416
max_batches	10000
Steps	8000, 9000
Classes	5
Filters	30

D. Instalasi Software

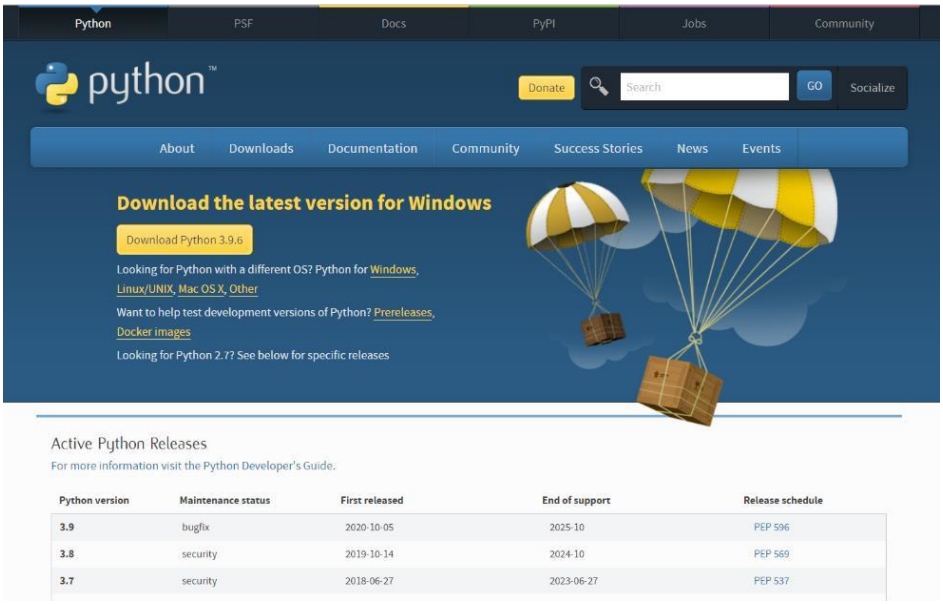
1. Download file weight dan file configuration YOLO

Langkah selanjutnya adalah melakukan download *file weight* dan *file configuration* YOLOv3 yang sudah selesai di-*training* seperti pada Gambar 4.5. File ini adalah inti dari algoritma YOLO untuk mendeteksi objek dan *file configuration* adalah pengaturan dari algoritma YOLO.

Gambar 4.5 File weight dan cfg YOLO

2. Instalasi Python

Pada penelitian ini, penulis menggunakan bahasa Python untuk melakukan pendeteksian.



Gambar 4.6 Tampilan *website* python

3. Instalasi OpenCV

Selanjutnya melakukan instalasi OpenCV. Disini penulis menggunakan Open CV 3 dengan code seperti pada Gambar 4.7.

```
>pip install opencv-python
```

Gambar 4.7 Kode program instalasi OpenCV

E. Deteksi Kendaraan dengan Algoritma You Only Look Once (YOLO)

Pada proses deteksi, setelah semua *software* yang dibutuhkan telah ter-instalasi dengan baik. Langkah selanjutnya adalah menggabungkan file *weight*, *configuration*, dan dataset pada satu folder.

1. Import library

Menggunakan library cv2, numpy, dan time. Library cv2 berfungsi untuk melakukan computer vision task. Library numpy berfungsi untuk pengolahan data numerical.

```
import cv2
import numpy as np
import time

print(cv2.__version__)
```

Gambar 4.8 Kode program import library

```
C:\Users\HP\anaconda3\envs\OpenCV\python.exe C:/Users/HP/PycharmProjects/pythonProject/detect.py
4.5.1
```

Gambar 4.9 Hasil dari keluaran versi OpenCV

2. Membaca Masukan Video

Pada bagian ini dilakukan membaca masukan video dengan mendefinisikan cv2.VideoCapture() untuk mendapatkan objek dari video capture pada kamera sesuai dengan lokasi penyimpanan. Lalu menggunakan variabel *'writer'* yang digunakan untuk menulis *frame* yang diproses. Kemudian menggunakan variabel 'h' dan 'w' untuk dimensi *frame*.

```
video = cv2.VideoCapture('kendaraan.mp4')
writer = None
h, w = None, Non
```

Gambar 4.10 Kode program membaca masukan video

3. Memuat *network* YOLOv3

Menggunakan file *weight*, cfg, dan *name files*. *Weight file* adalah model yang sudah di- training, inti dari algoritma YOLO untuk mendeteksi objek. Cfg file adalah file konfigurasi, dimana semua pengaturan algoritma YOLO terdapat pada file tersebut. Dan Name files adalah file yang berisi nama-nama objek yang dapat dideteksi menggunakan algoritma YOLO. Disini penulis menggunakan name files yang berisikan lima kelas yang akan dideteksi. Setelah itu mengatur minimum probabilitas untuk mengeliminasi prediksi rendah dengan nilai 0,5. Serta melakukan pengaturan untuk menyaring kotak pembatas yang rendah dengan nilai threshold

0,5 jika lebih besar maka objek akan terdeteksi dengan benar, jika tidak maka akan dilewatkan. Nilai threshold berubah dari 0 ke 1. Semakin dekat ke 1 maka semakin besar akurasi deteksi, jika semakin dekat ke 0 maka semakin sedikit akurasi tetapi juga semakin besar jumlah objek yang terdeteksi. Untuk menghasilkan warna pada setiap objek yang terdeteksi, menggunakan fungsi *randint* dengan seperti pada Gambar 4.11.

```
with open('kendaraan.names') as f:
    labels = [line.strip() for line in f]

print('List with labels names:')
print(labels)

network = cv2.dnn.readNetFromDarknet('yolov3_custom2.cfg',
                                     'yolov3_custom2_last.weights')
layers_names_all = network.getLayerNames()
layers_names_output = \
    [layers_names_all[i[0] - 1] for i in
network.getUnconnectedOutLayers()]
probability_minimum = 0.5
threshold = 0.5
colours = np.random.randint(0, 255, size=(len(labels), 3),
dtype='uint8')
```

Gambar 4.11 Kode program memuat *network* YOLOv3

```
List with labels names:
['mobil', 'sepeda motor', 'becak', 'truk', 'bus']
```

Gambar 4.12 Hasil keluaran *labels*

4. Membaca *frame* untuk perulangan

Kemudian melakukan perulangan pada *frame*. Disini akan dilakukan fungsi perulangan untuk menangkap *frame-by-frame* dengan fungsi *read*, yaitu membaca *frame* dari video masukan. Jika *frame* tidak diambil, misalnya pada akhir video, maka program akan menghentikan perulangannya.

```
f = 0
t = 0

while True:
   ret, frame = video.read()
    if not ret:
     break
    if w is None or h is None:
        h, w = frame.shape[:2]
```

Gambar 4.13 Kode program membaca *frame* untuk perulangan

5. Mengambil fungsi *blob* dari *frame*

Kemudian, data akan diolah menggunakan library OpenCV untuk diubah menjadi bentuk blob (Binary Large Object) dari frame. Menggunakan fungsi 'cv2.dnn.blobFromImage' akan mengembalikan 4-dimensional blob dari frame saat ini setelah pengurangan rata-rata, normalisasi, dan pertukaran saluran RB (Irwan, Putrada, & Prabowo, 2019). Pada fungsi

'cv2.dnn.blobFromImage' berisi parameter frame yang merupakan masukan gambar yang akan diproses melalui jaringan saraf dalam bentuk klasifikasi. 1/255.0 sebagai scale factor untuk melakukan pengurangan rata-rata. Ukuran (416,416) adalah ukuran pada YOLO. swapRb memberikan asumsi pada gambar berada dalam saluran BGR, namun nilai mean mengasumsikan kita menggunakan urutan RGB. Untuk mengatasi perbedaan ini, kita dapat menetapkan nilai menjadi ‘True’ untuk menukan saluran R dan B. Bentuk yang dihasilkan memiliki jumlah bingkai, jumlah saluran, lebar dan tinggi. Blob digunakan untuk mengekstrak fitur gambar dan mengubah ukurannya. Pada YOLO terdapat 3 ukuran frame, yaitu 320x320,

416x416, dan 609x609. Disini penulis menggunakan ukuran frame 416x416 agar proses deteksi memiliki kecepatan dan akurasi yang seimbang.

```
blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416),
                             swapRB=True,  crop=False)
```

Gambar 4.14 Kode program mengambil fungsi *blob* dari *frame*

6. Menerapkan *Forward Pass*

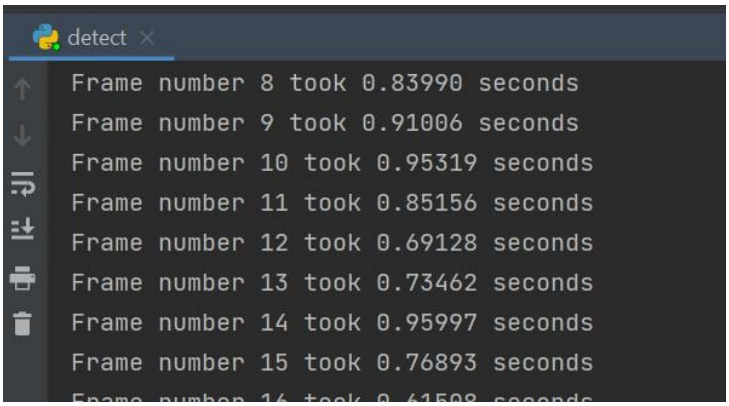
Mengimplementasi *forward pass* dengan '*blob*' melalui lapisan keluaran. Lalu, menghitung waktu yang dibutuhkan untuk *forward pass*.

```
network.setInput(blob)
    start = time.time()
    output_from_network = network.forward(layers_names_output)
    end = time.time()

    f += 1
    t += end - start

print('Frame number {0} took {1:.5f} seconds'.format(f, end - start))
```

Gambar 4.15 Kode program menerapkan *Forward Pass*



Gambar 4.16 Contoh hasil dari proses deteksi per *frame*

7. Mendapatkan *bounding boxes*

Pada langkah ini dilakukan ekstraksi seluruh informasi objek yang dideteksi dan menampilkannya pada layar. Melakukan inisialisasi list *bounding_boxes* untuk kotak pembatas yang mengelilingi objek. *Confidences* memberikan nilai keyakinan pada YOLO untuk suatu objek, nilai keyakinan (*confidence*) dari objek yang

terdeteksi bernilai 0 sampai 1. ClassIDs untuk memberikan label kelas objek yang terdeteksi.

Melakukan perulangan untuk setiap *output_from_network* dan perulangan untuk setiap *detected_object* pada *result* dengan mengekstrak *class_current* dan *confidence_current*. Menggunakan perintah *confidence_current* untuk menyaring prediksi yang lemah dengan memastikan objek yang terdeteksi dengan probabilitas lebih besar dari probabilitas minimum. Kemudian melakukan skala koordinat kotak pembatas untuk menampilkan dengan benar pada gambar asli. Lalu, mengekstraksi koordinat dan dimensi dari kotak pembatas dengan mengembalikan koordinat kotak pembatas dalam bentuk: *x_center, y_center, box_width, box_height*. Menggunakan informasi untuk mendapatkan *top-left (x, y)-coordinates* dari kotak pembatas. Lalu memperbarui list *bounding_boxes*, *confidences*, dan *classIDs*.

```
bounding_boxes = []
confidences = []
classIDs = []

    for result in output_from_network:

        for detected_objects in result:
            scores = detected_objects[5:]
            class_current = np.argmax(scores)
            confidence_current = scores[class_current]

            if confidence_current > probability_minimum:
                box_current = detected_objects[0:4] *
np.array([w, h, w, h])
                x_center, y_center, box_width, box_height =
box_current
                x_min = int(x_center - (box_width / 2))
                y_min = int(y_center - (box_height / 2))
                bounding_boxes.append([x_min, y_min,
int(box_width),
         int(box_height)])
          confidences.append(float(confidence_current))
          classIDs.append(class_current)
```

8. *Non-maximum Suppresion*

Menerapkan *non-maximum suppresion* atau yang disebut juga *non-maxima suppression* (NMS). Objek yang terdapat pada video atau gambar dapat memiliki ukuran dan bentuk yang berbeda, dan untuk menangkap setiap objek maka algoritma akan memuat kotak pembatas (*bounding boxes*). Jadi, untuk setiap objek yang terdapat pada video atau gambar harus memiliki satu kotak pembatas (*bounding boxes*). Untuk mendapatkan kotak pembatas (*bounding boxes*) terbaik dari beberapa kotak pembatas (*bounding boxes*) yang diprediksi, algoritma akan menggunakan fungsi *non-max*. teknik ini digunakan untuk "menekan" kotak pembatas yang kemungkinannya kecil dan hanya menyimpan yang terbaik. Tujuan dari *non- maximum suppression* adalah untuk memilih kotak pembatas (*bounding boxes*) terbaik pada suatu objek. NMS akan mempertimbangkan dua hal, yang pertama skor objektifitas yang diberikan oleh model, dan yang kedua IoU dari kotak pembatas (*bounding boxes*) (Hosang & May, n.d.).

Pada Gambar 4.18 memanfaatkan implementasi modul DNN pada OpenCV, kemudian melakukan *non-maximum suppression* dengan parameter *bounding_boxes, confidences, probability_minimun*, dan *threshold*.

```
results = cv2.dnn.NMSBoxes(bounding_boxes, confidences,
                           probability_minimum, threshold)
```

Gambar 4.18 Kode program *non-maximum suppresion*

9. Mendapatkan *bounding boxes* dengan *labels*

Menggunakan perintah if untuk memeriksa jika setidaknya ada satu objek yang terdeteksi setelah NMS. Dengan mendapatkan *bounding box* saat ini, lebar, dan tingginya. Kemudian mempersiapkan warna untuk *bounding box* saat ini dan mengkonversi dari *array* numpy. Setelah mendapatkan warna, selanjutnya menggambar *bounding box* pada *frame*. Setelah itu, mempersiapkan teks dengan label dan nilai *confidence* untuk *bounding box* saat ini dan menampilkan hasilnya.

```
if len(results) > 0:
   for i in results.flatten():
       x_min, y_min = bounding_boxes[i][0], bounding_boxes[i][1]
  box_width, box_height = bounding_boxes[i][2], bounding_boxes[i][3]
       colour_box_current = colours[classIDs[i]].tolist()

       cv2.rectangle(frame, (x_min, y_min),(x_min + box_width, y_min
+
      box_height), colour_box_current, 2)
  text_box_current = '{}: {:.4f}'.format(labels[int(classIDs[i])],
             confidences[i])
  cv2.putText(frame, text_box_current, (x_min, y_min - 5),
                  cv2.FONT_HERSHEY_SIMPLEX, 0.5, colour_box_current,
2)
```

Gambar 4.19 Kode program mendapatkan *bounding boxes* dengan *labels*

10. Menyimpan video proses deteksi

Pada bagian ini, akan menyimpan hasil prediksi masukan video dengan mengecek '*writer'*, kemudian '*writer'* akan diinisialisasi pada iterasi pertama dari perulangan. Mengeluarkan perkiraan waktu yang dibutuhkan untuk proses video. Dan mengeluarkan hasil akhir dari proses deteksi pada masukan video.

```
if writer is None:
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    writer = cv2.VideoWriter('resultvideo8juli.mp4', fourcc, 30,
                             (frame.shape[1], frame.shape[0]),
                             True)
writer.write(frame)


# Printing final results print()
print('Total number of frames', f)
```

Gambar 4.20 Kode program menyimpan video deteksi

```
Total number of frames 237
Total amount of time 139.25257 seconds
```

Gambar 4.21 Contoh hasil keluaran akhir deteksi

F. Pengujian Sistem

Pada langkah ini, dipaparkan hasil dari pengujian dari implementasi algoritma YOLO.

1. Pengujian pada performansi model YOLOv3

Pengujian dilakukan bertujuan untuk mengetahui tingkat keakurasian dari model *weight* YOLOv3 yang sudah di-*training* menjadi YOLOv3_custom. Pengujian performa *training* data yolov3_custom menggunakan data *training* menggunakan lima kelas, yaitu mobil, sepeda motor, becak, truk, dan bus. Data validasi yang sudah diberi anotasi diolah sebagai ground-truth box dibandingkan dan predicted box yang kemudian menghasilkan *confusion matrix*, kemudian

dikalkulasi untuk mendapatkan nilai *precision*, *recall*, *average precision* (AP), *F1-score*, *intersection over union* (IoU), dan *mean average precision* yang sudah dijelaskan pada BAB II.

Hasil dari *training* data dapat dilihat pada Tabel 4.3 hasil *training* pada kelas mobil menunjukan jumlah *True Positive* sebesar 315 jauh lebih besar dibandingkan jumlah *False Positive* sebesar 12 dengan nilai AP 99,88%. Pada kelas sepeda motor menunjukan jumlah *True Positive* sebesar 166 jauh lebih besar dibandingkan jumlah *False Positive* sebesar 4 dengan nilai AP 97,79%. Kelas becak menunjukan jumlah *True Positive* sebesar 6 jauh lebih besar dibandingkan jumlah *False Positive* sebesar 0 dengan nilai AP 100%. Kelas truk dan bus menunjukan jumlah *True Positive* sebesar 8 jauh lebih besar dibandingkan jumlah *False Positive* sebesar 0 dengan nilai AP pada kelas truk 100% dan kelas bus 99,09%. Hal ini mencerminkan sistem telah dapat mendeteksi objek dengan benar dari dataset yang telah dilatih. YOLOv3_custom membutuhkan 4 detik waktu proses untuk mendeteksi 5 kelas. Kemudian, diperoleh presisi sistem mengklasifikasikan objek dengan tepat mencapai 97%. Sementara, recall mengukur kemampuan model untuk menemukan seluruh objek positif mencapai nilai 98%. Pada pengujian dilakukan pula pengukuran parameter nilai precision dan recall untuk menentukan keseimbangan nilai diperoleh nilai F1-score sebesar 97%.

Tabel 4.3 Pengujian pada *training data*

Load Model		Yolov3_custom
Mobil	AP	99,88%
	TP	315
	FP	12
Sepeda Motor	AP	97,79%
	TP	166
	FP	4
Becak	AP	100%
	TP	6
	FP	0
Truk	AP	100%
	TP	8
	FP	0
Bus	AP	99,09%
	TP	8
	FP	0
FN		10
Waktu pemrosesan		4 detik
Recall		0.98
Precision		0,97
F1-Score		0,97
IoU		79,12%
mAP@0,5		99,35%

Pada Tabel 4.4 menunjukkan menunjukkan perbandingan akurasi waktu beberapa metode untuk pendeteksian objek. Metode YOLOv3 dengan masukan 608 x 608 menghasilkan nilai mAP dua persen lebih rendah dari metode FPN FRCN, yaitu 57,9% untuk YOLOv3-608, dan 59,1% untuk FPN FRCN, tetapi metode YOLOv3-608 dapat memproses masukan lebih cepat selama 51 ms dibandingkan FPN FRCN selama 172 ms. mAP adalah suatu metrik untuk mengukur akurasi objek detektor dan semakin tinggi mAP berarti pendeteksi objek semakin akurat. Beberapa metode yang lain menghasilkan nilai mAP yang baik, namun waktu pemrosesan lebih lama dibandingkan metode YOLOv3 dengan masukan 608x608, 416x416, dan 320x320 (Redmon & Farhadi, 2018). Sedangkan pada penelitian ini

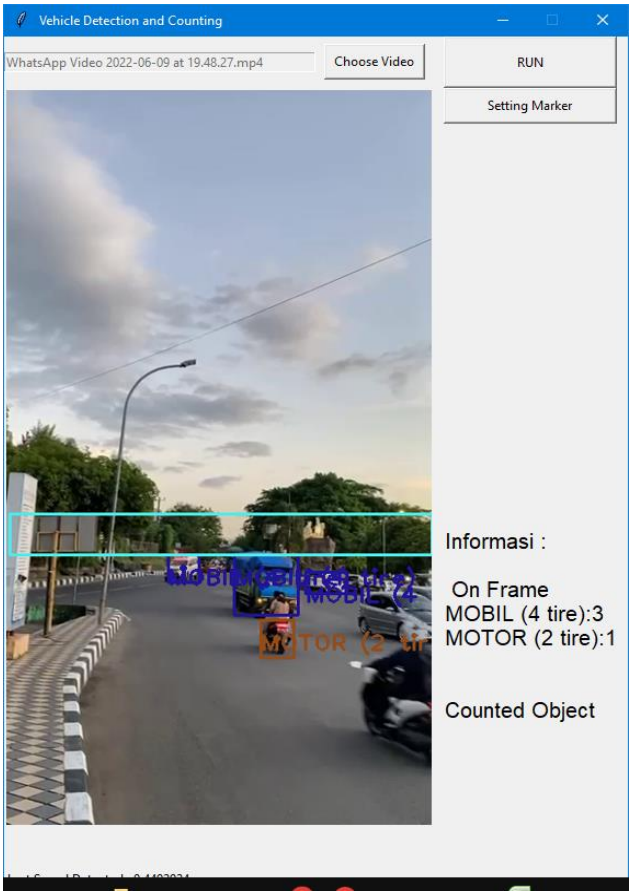
menggunakan *pre- trained weights* model YOLOv3 dengan teknik *transfer learning* menghasilkan nilai mAP sebesar 99,35% dengan waktu pemrosesan selama 4 detik. Hal ini menunjukkan metode YOLOv3 bekerja dengan baik untuk melakukan deteksi objek.

Tabel 4.4 Perbandingan kecepatan/akurasi dari beberapa model deteksi objek

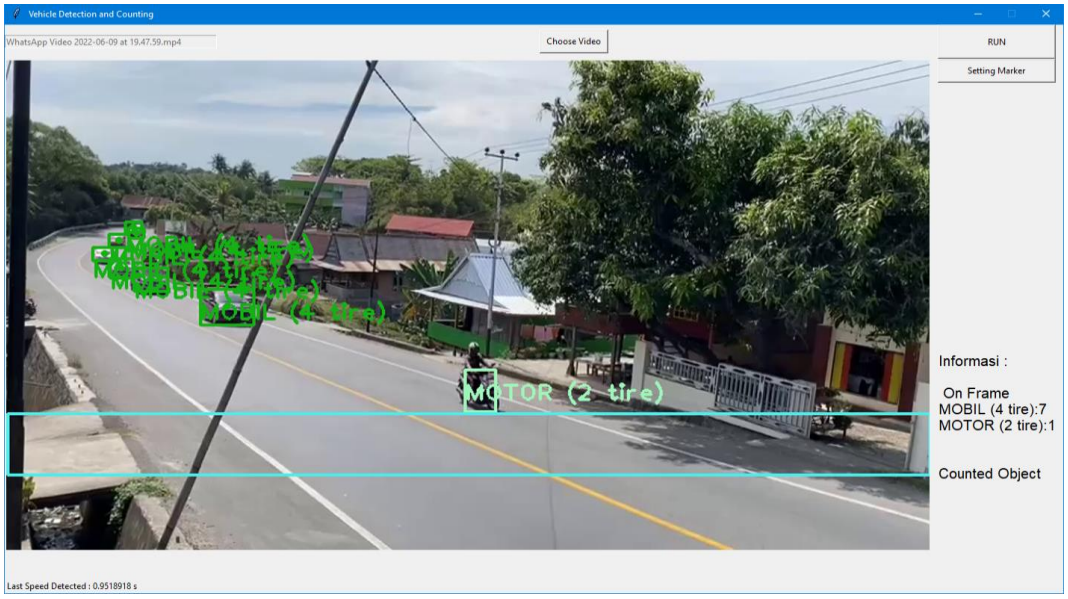
Metode	mAP@50	Waktu (ms)
SSD321	45,4%	61
DSSD321	46,1%	85
R-FCN	51,9%	85
SSD513	50,4%	125
DSSD513	53,3%	156
FPN FRCN	59,1%	172
RetinaNet-50-500	50,9%	73
RetinaNet-101-500	53,1%	90
RetinaNet-101-800	57,5%	198
YOLOv3-320	51,5%	22
YOLOv3-416	55,3%	29
YOLOv3-608	57,9%	51

2. Pengujian pada hasil deteksi

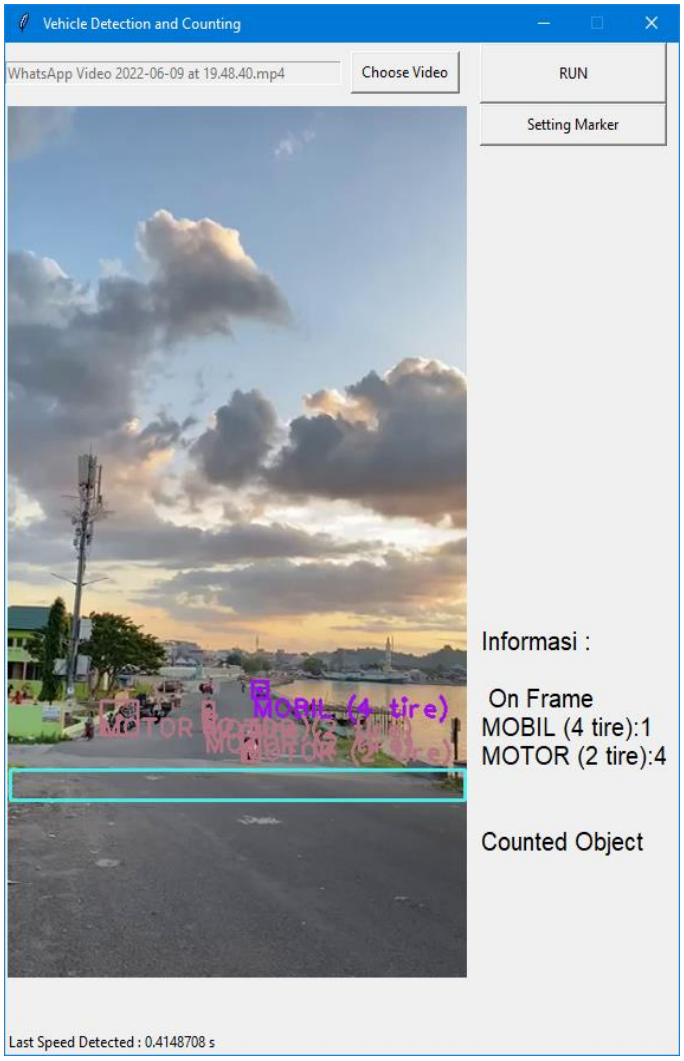
Pada pengujian deteksi, menggunakan *pre-trained weights* dengan lima kelas dilakukan untuk mengetahui keakuratan model dalam mendeteksi kelas yang sudah dilatih. Pengujian dapat dilakukan menggunakan program yang telah dibangun dan menggunkan model *weights* yang telah dilatih. Program kemudian di-*run* dengan masukan berupa video rekaman yang berbeda-beda untuk mendapatkan seberapa akurat model *weights* dapat mendeteksi kendaraan. Keluaran dari hasil deteksi adalah bounding box pada objek yang terdeteksi dan nilai confidence seperti.



Gambar 4.22 Hasil deteksi jalan satu



Gambar 4.23 Hasil deteksi jalan Dua



Gambar 4.24 Hasil deteksi jalan tiga

Bab ini membahas pengujian dan evaluasi pada aplikasi yang dikembangkan. Pengujian yang dilakukan adalah pengujian terhadap daftar fungsi aplikasi yang telah dijabarkan pada Bab III dan terhadap tujuan dibuatnya aplikasi ini, yakni melakukan deteksi jenis kendaraan dan menghitung objek kendaraan yang terdeteksi berdasarkan jenisnya.

3. Skenario Pengujian

Pada bagian ini akan dijelaskan tentang skenario pengujian yang dilakukan. Uji coba dilakukan untuk mengenali jenis kendaraan melalui input video. Terdapat beberapa skenario saat ujicoba, dikarenakan hasil *output* dipengaruhi skenario yang digunakan. Pada setiap skenario, penulis menggunakan 4 nilai batas waktu maksimal objek kendaraan untuk melewati wilayah deteksi yaitu 25, 30, 35, dan 40. Berikut ini adalah skenario pengujian yang diterapkan:

a. Skenario 1: Melakukan uji coba dengan kondisi jalan sepi.
   - ✓ Memasukkan video berformat .mp4 ke dalam aplikasi.
   - ✓ Aplikasi memuat file video yang berisi fitur-fitur objek kendaraan.
   - ✓ Aplikasi membaca video secara *frame by frame*.
   - ✓ Aplikasi mendeteksi objek kendaraan menggunakan *classifie*
   - ✓ Hasil akan keluar sesuai dengan fitur objek kendaraan yang dikenali oleh aplikasi.
   - ✓ Aplikasi memberikan tanda segi empat pada objek yang dianggap sebagai kendaraan.
   - ✓ Aplikasi melakukan pengenalan jenis ken daraan.
   - ✓ Hasil akan keluar sesuai dengan luas segi empat.
   - ✓ Hasil bisa berubah-ubah (tergantung jarak antar objek kendaraan dan nilai batas waktu maksimal).

b. Skenario 2: Melakukan uji coba dengan kondisi jalan normal.
   - ✓ Memasukkan video berformat .mp4 ke dalam aplikasi.
   - ✓ Aplikasi memuat file yang berisi fitur-fitur objek kendaraan.

- ✓ Aplikasi membaca video secara frame by frame.
- ✓ Aplikasi mendeteksi objek kendaraan menggunakan classifier.
- ✓ Hasil akan keluar sesuai dengan fitur objek kendaraan yang dikenali oleh aplikasi.
- ✓ Aplikasi memberikan tanda segi empat pada objek yang dianggap sebagai kendaraan.
- ✓ Aplikasi melakukan pengenalan jenis kendaraan.
- ✓ Hasil bisa berubah-ubah (tergantung jarak antar objek kendaraan dan nilai batas waktu maksimal).

c. Skenario 3: Melakukan uji coba dengan kondisi jalan padat.

- ✓ Memasukkan video berformat .mp4 ke dalam aplikasi.
- ✓ Aplikasi memuat file yang berisi fitur-fitur objek kendaraan.
- ✓ Aplikasi membaca video secara *frame by frame*.
- ✓ Aplikasi mendeteksi objek kendaraan menggunakan *classifier*
- ✓ Hasil akan keluar sesuai dengan fitur objek kendaraan yang dikenali oleh aplikasi.
- ✓ Aplikasi memberikan tanda segi empat pada objek yang dianggap sebagai kendaraan.
- ✓ Aplikasi melakukan pengenalan jenis kendaraan.
- ✓ Hasil akan keluar sesuai dengan luas segi empat.
- ✓ Hasil bisa berubah-ubah (tergantung jarak antar objek kendaraan dan nilai batas waktu maksimal).

Tabel 4.6 Skenario Pengujian

Kode Pengujian	Skenario Pengujian
SP-1	Pengujian dengan kondisi jalan sepi
SP-2	Pengujian dengan kondisi jalan normal
SP-3	Pengujian dengan kondisi jalan padat

4. Pengujian dengan Kondisi Jalan Sepi

Pengujian ini dimulai dengan pengguna melakukan input video ke aplikasi. Video yang digunakan adalah video rekaman lalu lintas yang direkam dengan video rekaman. Rincian skenario pengujian pada kasus penggunaan.

Tabel 4.7 Tabel Pengujian dengan Kondisi Jalan Sepi

ID	**SP-1**
Nama	Mendeteksi jenis kendaraan dalam kondisi jalan sepi.
Tujuan Pengujian	Pengguna mengetahui jenis kendaraan pada video dan jumlah kendaraan yang terdeteksi.

Skenario	Mendeteksi jenis kendaraan pada video
Kondisi Awal	Pengguna membuka aplikasi.
Data Uji	Video dengan durasi 20 detik.

Langkah Pengujian	1. Pengguna memasukkan 2. video ke dalam Pengguna menjalankan aplikasi. 3. Aplikasi melakukan pengenalan jenis kendaraan pada video sesuai dengan durasi video. 4. Aplikasi mencatat hasil deteksi jenis kendaraan. 5. Aplikasi perangkat bergerak akan melakukan proses. 6. Aplikasi akan menampilkan hasil berupa *pop-up*.
Hasil Yang Diharapkan	Tertampil detail jenis kendaraan serta jumlah kendaraan
Hasil Yang Didapatkan	Detail jenis kendaraan dan jumlah kendaraan.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Aktor mendapatkan yang berisi ID serta jenis kendaraan yang terdeteksi dari video.

5. Pengujian dengan Kondisi Jalan Normal

Pengujian ini dimulai dengan pengguna melakukan input video ke aplikasi. Video yang digunakan adalah video rekaman lalu lintas yang direkam dengan video rekaman. Rincian skenario pengujian pada kasus penggunaan.

Tabel 4.8 Tabel Pengujian dengan Kondisi Jalan Normal

ID	**SP-1**
Nama	Mendeteksi jenis kendaraan dalam kondisi jalan Normal.
Tujuan Pengujian	Pengguna mengetahui jenis kendaraan pada video dan jumlah kendaraan yang terdeteksi.

Skenario	Mendeteksi jenis kendaraan pada video
Kondisi Awal	Pengguna membuka aplikasi.
Data Uji	Video dengan durasi 20 detik.
Langkah Pengujian	1. Pengguna memasukkan 2. video ke dalam Pengguna menjalankan aplikasi. 3. Aplikasi melakukan pengenalan jenis kendaraan pada video sesuai dengan durasi video. 4. Aplikasi mencatat hasil deteksi jenis kendaraan. 5. Aplikasi perangkat bergerak akan melakukan proses. 6. Aplikasi akan menampilkan hasil berupa *pop-up.*
Hasil Yang Diharapkan	Tertampil detail jenis kendaraan serta jumlah kendaraan
Hasil Yang Didapatkan	Detail jenis kendaraan dan jumlah kendaraan.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Aktor mendapatkan yang berisi ID serta jenis kendaraan yang terdeteksi dari video.

6. Pengujian dengan Kondisi Jalan Padat

Pengujian ini dimulai dengan pengguna melakukan input video ke aplikasi. Video yang digunakan adalah video rekaman lalu lintas yang direkam dengan video rekaman. Rincian skenario pengujian pada kasus penggunaan.

Tabel 4.8 Tabel Pengujian dengan Kondisi Jalan Normal

ID	**SP-1**
Nama	Mendeteksi jenis kendaraan dalam kondisi jalan padat.
Tujuan Pengujian	Pengguna mengetahui jenis kendaraan pada video dan jumlah kendaraan yang terdeteksi.

Skenario	Mendeteksi jenis kendaraan pada video
Kondisi Awal	Pengguna membuka aplikasi.
Data Uji	Video dengan durasi 20 detik.
Langkah Pengujian	1. Pengguna memasukkan 2. video ke dalam Pengguna menjalankan aplikasi. 3. Aplikasi melakukan pengenalan jenis kendaraan pada video sesuai dengan durasi video. 4. Aplikasi mencatat hasil deteksi jenis kendaraan. 5. Aplikasi perangkat bergerak akan melakukan proses. 6. Aplikasi akan menampilkan hasil berupa *pop-up*.
Hasil Yang Diharapkan	Tertampil detail jenis kendaraan serta jumlah kendaraan
Hasil Yang Didapatkan	Detail jenis kendaraan dan jumlah kendaraan.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Aktor mendapatkan yang berisi ID serta jenis kendaraan yang terdeteksi dari video.

7. Akurasi Pengujian Fungsionalitas

Pada sub bab ini akan diberikan hasil evaluasi dari pengujian-pengujian yang telah dilakukan. Evaluasi yang diberikan meliputi evaluasi pengujian daftar fungsi sesuai dengan modul yang terdapat aplikasi.

Pada pengujian, terdapat tiga kondisi jalan yaitu sepi, padat, dan sangat padat. Ketiga kondisi ini mewakili kondisi nyata

yang terjadi di jalan raya pada umumnya. Pada Tabel memperlihatkan hasil pengujian dari tiga kondisi jalan raya, jenis kendaraan yang terdeteksi, mana yang sesuai dengan definisi, mana yang tidak sesuai dengan definisi, dan mana yang tidak terdapat pada video inputan.

Tabel 4.10 Hasil Deteksi Jenis Kendaraan

	Jenis Kendaraan		
	Kecil	**Sedang**	**Besar**
Sepi	V	V	V
Normal	V	V	V
Padat	V	V	V

Keterangan simbol:

V = Sesuai dan dipakai.

X = Tidak dipakai karena tidak ditemukan.

Akurasi pengenalan jenis kendaraan dihitung menggunakan Persamaan yang merupakan perhitungan dengan cara sebagai berikut:

$$Acc = \frac{V_{detect}}{V_{real}} * 100\%$$

Di mana V_{detect} adalah banyaknya suatu jenis kendaraan yang terdeteksi dan V_{real} adalah jumlah sebenarnya dari suatu jenis kendaraan. V_{real} diperoleh dengan cara menghitung secara manual setiap jenis kendaraan.

8. Evaluasi Pengujian Fungsionalitas

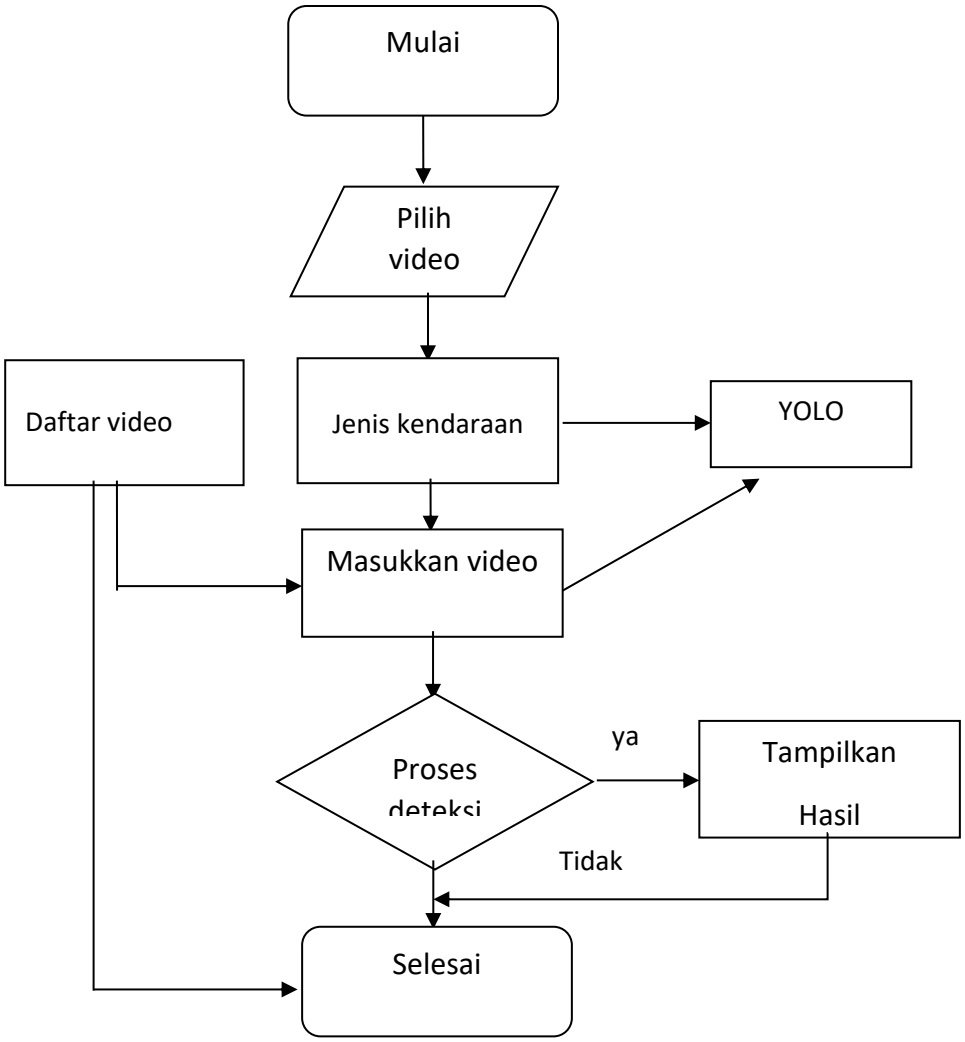
Rangkuman mengenai hasil pengujian fungsionalitas dapat dilihat pada Tabel. Berdasarkan data pada tabel tersebut, semua skenario pengujian berhasil dan program berjalan dengan baik. Sehingga bisa ditarik kesimpulan bahwa fungsionalitas dari aplikasi telah dapat bekerja sesuai dengan yang diharapkan.

Tabel 4.23 Rangkuman Hasil Pengujian

Kode Pengujian	Skenario Pengujian	Hasil
SP-F1	Mendeteksi objek kendaraan pada video	Berhasil
SP-F2	Mendeteksi jenis kendaraan pada video	Berhasil
SP-F3	Mencatat hasil deteksi pada video	Berhasil

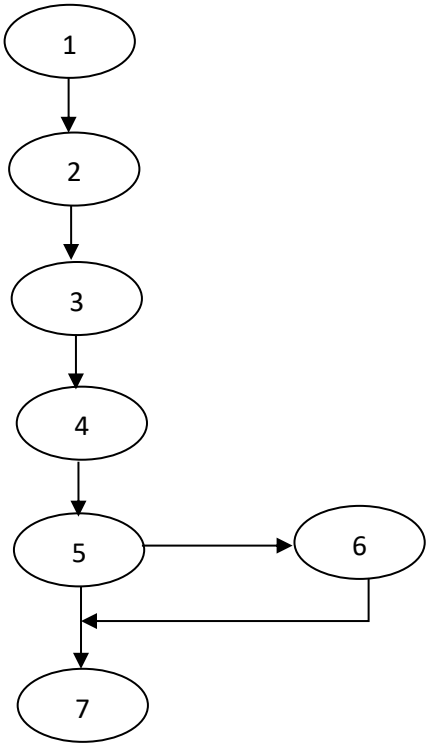
9. Pengujian White box

a. Flowchart Aplikasi



Gambar 12. Flowchart Aplikasi

b. Flowgraph Aplikasi



Gambar 13 . Flowgraph Aplikasi

1) Proses Perhitungan

Dari Gambar FlowGraph di atas dapat di lakukan proses perhitungan sebagai berikut :

a. Menghitung *Cyclomatic Complexcity* V (G) = E- N + 2

N (node) = 7

E (edge) = 7

P (*predikat node)* = 1

Penyelesaian : V(G) = E – N + 2

= 7-7 + 2

= 2

Predikat Node (N) = P + 1

= 1 + 1

= 2

b. Berdasarkan perhitungan *Cyclomatic Complexcity* dari Flowgraph diatas memiliki *Region = 2*

c. *Independent Independent Path* pada *flowgraph* diatas adalah :

Path 1 = 1 – 2 – 3 – 4 – 5 – 6 - 7

Path 2 = 1 – 2 – 3 – 4 – 5 – 7

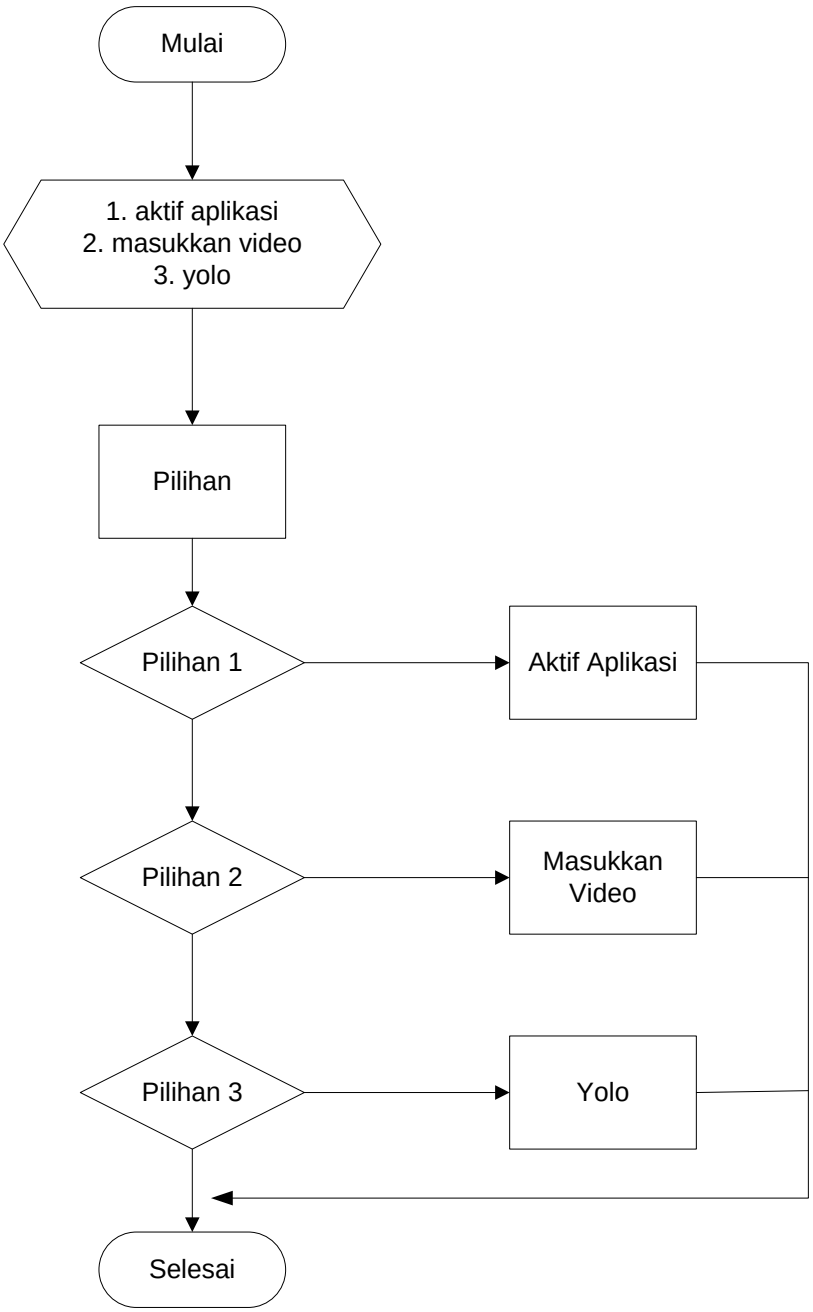
d. Grafik Matriks

Tabel 12. Grafik Matriks

	1	2	3	4	5	6	7	E - 1
1		1						1-1 = 0
2			1					1-1 = 0
3				1				1-1= 0
4					1			1-1= 0
5						1	1	2-1 = 1
6							1	1-1= 1
7								
Zum (E+1)								1+1 = 2

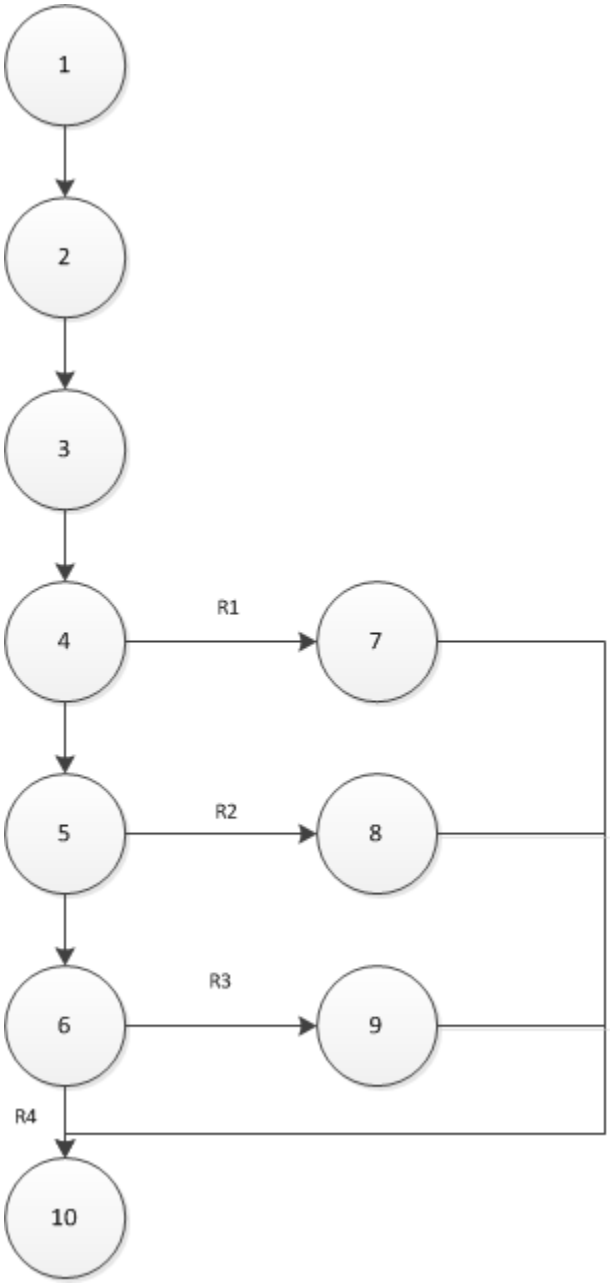
a. *White box* menjalankan aplikasi

1) *Flowchart* menjalankan aplikasi



Gambar 14 *Flowchart* menjalankan aplikasi

2) Flowgraph menjalankan aplikasi



Gambar 15 *Flowgraph* menjalankan aplikasi

3) Proses perhitungan menjalankan aplikasi

Dari gambar *flowgraph* diatas dapat dilakukan proses perhitungan sebagai berikut :

a) Menghitung C*yclomatic Complexcity* V(G) dari *Egde* dan *Node*:

Dengan Rumus : V (G) = E – N + 2

N(*node*) = 10

E(*edge*) = 12

P (*predikat node*) = 3

Penyelesaian : V(G) = E – N + 2

= 12- 10 + 2

= 4

Predikat Node (N) = P + 1

= 3 + 1 = 4

b) Berdasarkan perhitungan C*yclomatic Complexcity* dari *Flowgraph* diatas memiliki *Region* = 4

c) *Independent Path* pada *flowgraph* diatas adalah :

Path 1 = 1 – 2 – 3 – 4 – 7 - 10

Path 2 = 1 – 2 – 3 – 4 – 5 – 8 - 10

Path 3 = 1 – 2 – 3 – 4 – 5 – 6– 9 - 10

Path 4 = 1 – 2 – 3 – 4 – 5 –6 - 10

d) Grafik Matriks Menjalankan Aplikasi

Tabel 13 Grafik matriks menjalankan aplikasi

	1	2	3	4	5	6	7	8	9	10	E-1

1		1									1-1=0
2			1								1-1=0
3				1							1-1=0
4					1		1				2-1=1
5						1		1			2-1=1
6									1	1	2-1=1
7										1	1-1=0
8										1	1-1=0
9										1	1-1=0
10											0
SUM(E+1)											3+1=4

BAB V

PENUTUP

A. Kesimpulan

Dari banyaknya hasil pembahasan di atas dapat disimpulkan beberapa dianataranya adalah

1. Sistem aplikasi penghitung kendaraan yang melintas di jalan raya menggunakan bantuan dari metode YOLO *object detection* yang dimodifikasi sehingga tidak hanya mendeteksi kendaraan tapi juga dapat menghitung kendaraan yang melewatinya.
2. Sistem aplikasi penghitung kendaraan yang melintas tidak hanya menggunakan metode YOLO *object detection* melainkan juga menggunakan beberapa *libray* yang ada pada aplikasi Python Anaconda.
3. Hasil dari pengujian aplikasi dengan metode YOLO object detection sudah bisa membedakan kendaraan dengan roda 4 atau lebih ditandai dengan deteksi kotak berwarna hijau pada kendaraan yang berada di frame video.
4. Hasil pengujian aplikasi penghitung kendaraan berdasarkan metode YOLO object detection telah berhasil menghitung jumlah kendaran yang melewati sensor deteksi walaupun dengan nilai keakuratan yang belum mencapai 60%.

B. Saran

Dalam perancangan dan pengujian yang telah dijalankan oleh penulis, terdapat beberapa hal yang dapat ditambahkan supaya hasil perancangan lebih baik dari penulis, diantaranya adalah:

1. Menambahkan semua jenis kendaraan yang terdeteksi tidak hanya mendeteksi kendaraan dengan roda 4 atau lebih.

2. Dapat menambahkan nilai akurasi dari aplikasi penghitung kendaraan berdasarkan metode YOLO *object detection* yang telah dibuat penulis.

3. Membandingkan nilai akurasi dari aplikasi penghitung kendaraan berdasarkan metode YOLO *object detection* dengan metode-metode lainnya

DAFTAR PUSTAKA

Adi nugroho 2017 dalam buku “Rekayasa Perangkat Lunak Beriorentasi Objek” Penerbit ANDI

G.P.S. Prasanthi, K. Sirisha, G. Ramya3,B. Padma, 2016. Speech to Text Conversion Using

HMM.International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE) Volume 5, Issue 3, March 2016

Hossein Amerkashi,2015, Absolute App Inventor 2: Android Programming for all ages Kindle Edition,Kindle Store

Kamriani,F. & Roy,K.,2015,App Inventor 2 Essentials A Step -By-Step Introductory Guide to mobile app development with app inventor 2,Birmingham:Packt

Shivam Sharma, 2015. Speech Recognition with Hidden Markov Model: A Review.international Journal of Scientific & Engineering Research, Volume 6, Issue 11, November-2015.

Jugianto, 2018. Aplikasi Pada Sebuah Komputer. Penerbit. Andi

Jay A. Kreibich 2018 Pengenalan SQLite Penerbit ANDI

Sharon, Jacob, Israel. 2019. *Speech Processing In Modern Communication*. Germany: Springer. Ebook.

Shalahuddin 2008, *Pengenalan Pemrograman java* Penerbit Andi

Irawan. 2018. *Membuat Aplikasi Android untuk Orang Awam*. Palembang: Maxkom.