Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – IF184802

DETEKSI KENDARAAN BERBASIS VIDEO MENGGUNAKAN METODE DEEP LEARNING

PANDITO HUDIARSO ABDUL RAHIM SETIA NEGARA
NRP 05111540000060

Dosen Pembimbing I
Dini Adni Navastara, S.Kom., M.Sc.

Dosen Pembimbing II
Dr. Eng. Nanik Suciati, S.Kom, M.Kom.

Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

(Halaman ini sengaja dikosongkan)

TUGAS AKHIR – IF184802

DETEKSI KENDARAAN BERBASIS VIDEO MENGGUNAKAN METODE DEEP LEARNING

PANDITO HUDIARSO ABDUL RAHIM SETIA NEGARA
NRP 05111540000060

Dosen Pembimbing I
Dini Adni Navastara, S.Kom., M.Sc.

Dosen Pembimbing II
Dr. Eng. Nanik Suciati, S.Kom, M.Kom.

Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESIS – IF184802

VIDEO-BASED VEHICLE DETECTION USING DEEP LEARNING METHOD

PANDITO HUDIARSO ABDUL RAHIM SETIA NEGARA
NRP 05111540000060

First Advisor
Dini Adni Navastara, S.Kom., M.Sc.

Second Advisor
Dr. Eng. Nanik Suciati, S.Kom, M.Kom.

Department of Informatics Engineering
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2020

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

DETEKSI KENDARAAN BERBASIS VIDEO MENGGUNAKAN METODE *DEEP LEARNING*

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas dan Visi
Program Studi S-1 Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh:
PANDITO HUDIARSO ABDUL RAHIM SETIA NEGARA
NRP: 05111540000060

Disetujui oleh Pembimbing Tugas Akhir:

1. Dini Adni Navastara, S.Kom, M.Sc. .....
   (NIP. 19851017 201504 2 001) (Pembimbing 1)

2. Dr. Eng. Nanik Suciati, S.Kom, M.Kom. .....
   (NIP. 19710428 199412 2 001) (Pembimbing 2)

SURABAYA
Januari, 2020

(Halaman ini sengaja dikosongkan)

DETEKSI KENDARAAN BERBASIS VIDEO MENGGUNAKAN METODE *DEEP LEARNING*

Nama Mahasiswa	**:**	**Pandito Hudiarso Abdul Rahim Setia Negara**
NRP	**:**	**05111540000060**
Jurusan	**:**	**Informatika, ELECTICS-ITS**
Dosen Pembimbing 1	**:**	**Dini Adni Navastara, S.Kom., M.Sc.**
Dosen Pembimbing 2	**:**	**Dr. Eng. Nanik Suciati, S.Kom, M.Kom.**

ABSTRAK

Pendeteksian kendaraan sangat identic dengan analisis sebuah rekaman. Banyak perusahaan, badan riset dan universitas yang terus mengembangkan machine learning agar mendapat hasil yang lebih akurat dan cepat. Dari situlah lahir algoritma deep learning.

Mask Region-Convolutional Neural Network (CNN) adalah salah satu deep neural network yang cocok digunakan untuk mengolah data yang berbentuk 2 dimensi, seperti rekaman. Data gambar pada rekaman diproses menjadi features map. Selanjutnya, features map diproses sehingga menghasilkan label dan bounding box. Terakhir, masking dilakukan terhadap daerah objek.

Pada tugas akhir ini, dilakukan implementasi Mask R-CNN untuk melakukan pendeteksian kendaraan pada data video. Data uji merupakan data rekaman yang diambil manual. Data latih diambil dari model pre-trained MS-COCO yang memiliki gambar lebih dari 80.000 gambar. Hasil uji coba optimal didapatkan dari arsitektur Mask R-CNN nya dengan nilai akurasi 79,79%.

Kata kunci: deteksi kendaraan, machine learning, deep learning, mask region-convolutional neural network.

(Halaman ini sengaja dikosongkan)

VIDEO-BASED VEHICLE DETECTION USING DEEP LEARNING METHOD

Student's Name : Pandito Hudiarso Abdul Rahim Setia Negara
Student's ID : 05111540000060
Department : Informatics Engineering, ELECTICS-ITS
First Advisor : Dini Adni Navastara, S.Kom., M.Sc.
Second Advisor : Dr. Eng. Nanik Suciati, S.Kom, M.Kom.

ABSTRACT

Vehicle detection is very identic with video analysis. Many companies, researchers, and universities keep developing this method to get fastest and most accurate result. From that demand, deep learning was created.

Mask Region-Convolutional Neural Network (Mask R-CNN) is one of deep neural network that satisfy 2 dimentional data type processing, such as video. Frames from video will be processed into features map. After that, features map is processed to produce labels and bounding boxes. Last step, masking is done on the object area

In this final project, Mask R-CNN implementation performed vehicle detection on video data. Testing Data video is taken manually. Training Data is taken from MS-COCO pre-trained model, having more than 80.000 pictures in it. Optimal testing Result is taken from modified parameter of the method with accuracy 79,79%.

Keywords: vehicle detection, machine learning, deep learning, mask region-convolutional neural network.

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Puji syukur saya sampaikan kepada Allah yang Maha Kuasa karena berkat rahmat-Nya saya dapat melaksanakan Tugas Akhir yang berjudul:

"**DETEKSI KENDARAAN BERBASIS VIDEO MENGGUNAKAN METODE DEEP LEARNING**"

Terselesaikannya Tugas Akhir ini tidak terlepas dari bantuan dan dukungan banyak pihak, oleh karena itu melalui lembar ini penulis ingin mengucapkan terima kasih dan penghormatan kepada:

1. Allah SWT, karena limpahan rahmat dan karunia-Nya penulis dapat menyelesaikan Tugas Akhir dan juga perkuliahan di Informatika ITS.
2. Kedua orangtua penulis, dan anggota keluarga lainnya yang telah memberikan dukungan doa, moral, dan material kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir ini.
3. Dini Adni Navastara, S.Kom., M.Sc. dan Dr. Eng. Nanik Suciati, S.Kom., M.Kom. selaku pembimbing I dan II yang telah membimbing dan memberikan motivasi, nasihat dan bimbingan dalam menyelesaikan Tugas Akhir ini.
4. Dr. Eng. Darlis Herumurti, S.Kom., M.Kom. selaku Ketua Departemen Informatika ITS dan seluruh dosen dan karyawan Departemen Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa kuliah di Informatika ITS.
5. Admin-admin Laboratorium Komputasi Cerdas & Visi (KCV) yang memberikan kesempatan penulis untuk fokus mengerjakan Tugas Akhir ini dan menyediakan tempat di laboratorium tersebut.

6. Teman-Teman dari kontrakan Kampar yang selalu memberikan saya motivasi.
7. Teman-teman dari wardug yang selalu memberikan dukungan untuk penyelesaian tugas ini.
8. Seluruh mahasiswa Informatika ITS angkatan 2015 yang telah menjadi teman penulis selama menjalani masa kuliah di Informatika ITS.
9. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa laporan Tugas Akhir ini masih memiliki banyak kekurangan. Oleh karena itu dengan segala kerendahan hati penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan penulis kedepannya. Selain itu, penulis berharap laporan Tugas Akhir ini dapat berguna bagi pembaca secara umum.

Surabaya, Januari 2020

DAFTAR ISI

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Deteksi kendaraan memainkan peran penting pada sistem transportasi cerdas berbasis teknologi [1]. Dalam pengembangannya, deteksi kendaraan memilik tujuan yang berbeda-beda. Deteksi kendaraan dilakukan pada rekaman video baik secara real time maupun rekaman yang sudah ada.

Pengembangan deteksi kendaraan memilik beberapa tujuan. Salah satu tujuan pengembangan yang paling mencolok adalah penerapan mobil *self-driving*. Perkembangan mobil *self-driving* juga membuat pihak otoritas jalan raya memberikan perhatian lebih terhadap keamanan jalan raya melalui *monitoring* dan *controlling* berbasis teknologi [2]. Oleh karena itu, dibutuhkan deteksi kendaraan yang efektif dan akurat.

Banyak dari algoritma pendeteksi objek mengajukan cara yang berhubungan dengan aliran optik seperti mosi objek, permukaan pada gambar, serta sudut pandang pengambilan gambar objek [3]. Pada pengembangannya, deteksi objek dengan *machine learning* digunakan seperti menggunakan metode Viola-Jones dan *Histogram of Oriented Gradients* [4]. Namun penggunaan metode-metode ini menemui kegagalan ketika latar belakang dari gambar memiliki gambar kendaraan lebih dari satu [5]. Lalu pada pengembangannya, metode *deep learning* mulai digunakan dalam deteksi objek.

Metode *deep learning* berkembang mulai dari *Region-Convolutional Neural Network* (R-CNN) yang dapat melakukan deteksi objek pada sebuah gambar berdasarkan area objek. Lalu, optimasi dilakukan sehingga dilahirkan metode seperti *Fast Region-Convolutional Neural Network* (Fast R-CNN) dan *Faster Region-Convolutional Neural Network* (Faster R-CNN). Pengembangan dari metode ini sampai pada metode *Mask Region-Convolutional Neural Network* (Mask R-CNN) yang dapat melakukan deteksi hingga ke level piksel [6].

Pada tugas akhir ini, dilakukan implementasi salah satu metode *deep learning* yaitu Mask R-CNN untuk deteksi kendaraan berbasis video. Hasil dari tugas akhir ini diharapkan dapat digunakan di berbagai pengenalan kendaraan berbasis video seperti mobil *self-driving*, *monitoring* dan *controlling* jalan raya, serta teknologi *traffic counting.*

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana cara mengimplementasikan Mask R-CNN untuk deteksi kendaraan?
2. Bagaimana tingkat performa sistem metode Mask R-CNN diimplementasikan pada deteksi berbasis video?
3. Bagaimana tingkat performas sitem dibandingkan dengan metode lainnya?

1.3 Batasan Permasalahan

Permasalahan yang dibahas pada Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Data latih yang digunakan adalah model *pre-trained* dari dataset MS-COCO
2. Implementasi program menggunakan bahasa pemrograman *Python 3*.
3. Sudut pandang video yang dijadikan testing adalah sudut pandang depan, samping, dan belakang kendaraan.
4. Ada 4 jenis kendaraan yang dikenali yaitu adalah motor, mobil, bus, dan truk.

1.4 Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah untuk membangun sebuah sistem deteksi video menggunakan metode *deep learning* (Mask-RCNN) agar dapat mengenali kendaraan yang ada dalam video.

1.5 Manfaat

Tugas akhir ini diharapkan dapat diterapkan pada sistem yang membutuhkan deteksi kendaraan seperti sistem analisis kecelakaan berbasis video, sistem *traffic counting*¸ dan juga sistem otonom pada mobil *self-driving.*

1.6 Metodologi

Pembuatan Tugas Akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Proposal Tugas Akhir

Tahapan awal dari Tugas Akhir ini adalah penyusunan Proposal Tugas Akhir yang berisi pendahuluan, deskripsi dan gagasan metode-metode yang dibuat dalam Tugas Akhir ini. Pendahuluan ini terdiri dari latar belakang diajukannya Tugas Akhir, rumusan masalah dan batasan masalah yang ditetapkan, serta manfaat dari hasil pembuatan Tugas Akhir ini. Selain itu, dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan Tugas Akhir. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan Tugas Akhir.

1.6.2 Studi Literatur

Pada tahap ini dilakukan pencarian literatur berupa jurnal yang digunakan sebagai referensi untuk pengerjaan tugas akhir ini. Literatur yang dipelajari pada pengerjaan tugas akhir ini berasal dari jurnal ilmiah yang diambil dari berbagai sumber di internet, beserta berbagai literatur online tambahan terkait penggunaan *library* pada python 3, *Mask Region-Convolutional Neural Network*, *TensorFlow* dan *Keras*.

1.6.3 Implementasi Perangkat Lunak

Pada tahap ini akan dilaksanakan implementasi metode dan algoritma yang telah direncanakan. Implementasi sistem menggunakan *Python 3* sebagai bahasa pemrograman, *TensorFlow* dan *Keras* sebagai *framework*, serta *library* pendukung lainya.

1.6.4 Pengujian dan Evaluasi

Tahap pengujian dan evaluasi dilakukan menggunakan dataset MS-COCO sebagai *training* dan video rekaman kendaraan lalu lintas yang diambil sebagai *testing* untuk mengetahui hasil dan performa arsitektur yang telah dibangun. Evaluasi dilakukan dengan metode pengukuran akurasi, *precision,* dan *recall*.

1.6.5 Penyusunan Buku

Pada tahap ini dilakukan penyusunan buku yang menjelaskan seluruh konsep, teori dasar dari metode yang digunakan, implementasi, serta hasil yang telah dikerjakan sebagai dokumentasi dari pelaksanaan Tugas Akhir.

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan Tugas Akhir adalah sebagai berikut:

Bab I Pendahuluan

Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan Tugas Akhir.

Bab II Tinjauan Pustaka

Bab ini berisi kajian teori dari metode dan algoritma yang digunakan dalam penyusunan Tugas Akhir ini. Secara garis besar, bab ini berisi tentang *Mask Region-Convolutional Neural Network* dan *library* yang digunakan.

Bab III Perancangan Sistem

Bab ini berisi pembahasan mengenai perancangan dari metode *Mask Region-Convolutional Neural Network* yang digunakan untuk pengenalan ekspresi wajah manusia pada data gambar.

Bab IV Implementasi

Bab ini membahas implementasi dari perancangan yang telah dibuat pada bab sebelumnya. Penjelasan berupa kode yang digunakan untuk proses implementasi.

Bab V Uji Coba Dan Evaluasi

Bab ini membahas tahapan uji coba, kemudian hasil uji coba dievaluasi terhadap kinerja dari sistem yang dibangun.

Bab VI Kesimpulan dan Saran

Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses dan tertulis saat pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.

(Halaman ini sengaja dikosongkan)

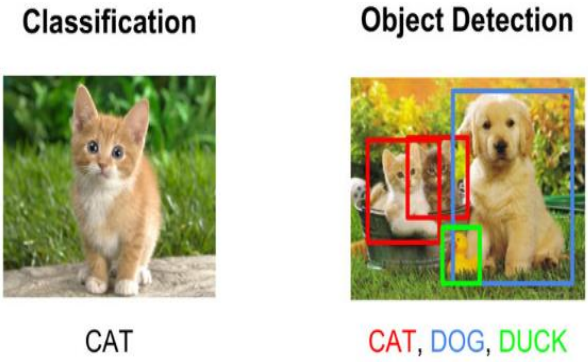
BAB II
TINJAUAN PUSTAKA

Bab ini membahas mengenai teori-teori dasar yang digunakan dalam Tugas Akhir. Teori-teori tersebut diantaranya adalah dasar pendeteksian objek dan *Mask Region-Convolutional Neural Network*, dan beberapa teori lain yang mendukung pembuatan Tugas Akhir. Penjelasan ini bertujuan untuk memberikan gambaran umum dan diharapkan dapat mendukung sistem yang dibangun.

2.1 Deteksi Objek

Pada dasarnya, deteksi objek adalah proses yang diawali dengan pembuatan *classifier* yang dapat mengklasifikasi gambar yang telah dipotong pada sebuah objek [7]. *Classifier* ini dibangun oleh model yang didapat dari dataset yang dipilih dan dijadikan model. Untuk mendapatkan evaluasi dari sebuah deteksi objek, dilakukan *testing* terhadap model yang telah dibangun agar dapat menentukan apakah deteksi objek tersebut berjalan baik atau tidak.

Perbedaan mendasar antara deteksi objek dengan klasifikasi gambar terletak pada hasil akhirnya. Klasifikasi gambar hanya dapat mengklasifikasi sebuah gambar tanpa melakukan klasifikasi terhadap semua objek pada gambar objek tersebut. Sedangkan deteksi objek dapat mengsegmentasi gambar sehingga semua objek pada gambar dapat di deteksi [8]. Perbedaan dapat dilihat pada Gambar 2.1.



Gambar 2.1 Perbedaan klasifikasi gambar dan deteksi objek [8]

2.2 Mask Region-Convolutional Neural Network

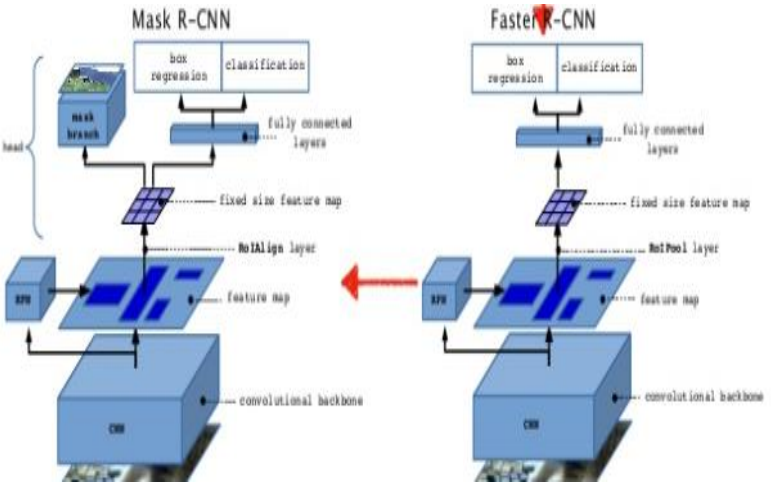
Penggunaan metode *deep learning* di ranah klasifikasi gambar dan deteksi objek dimulai dari penggunaan *Convolutional Neural Network* (CNN). Namun penggunaan CNN hanya terbatas pada klasifikasi gambar dan tidak dapat melakukan segmentasi objek-objek pada sebuah gambar. Untuk menjawab kebutuhan ini maka dimulai riset kearah deteksi objek dengan segmentasi gambar [6].

Region-Convolutional Neural Network (R-CNN) adalah terobosan pertama dari jenis CNN yang dapat melakukan segmentasi gambar. Dengan memanfaatkan *selective search*¸ metode ini dapat menghasilkan *proposal region* dari sebuah gambar sebagai kandidat objek sehingga didapat hasil deteksi seluruh objek pada sebuah gambar [9]. Kebutuhan ini juga berkembang sesuai dengan keinginan pengguna. Kemunculan *Fast Region-Convolutional Neural Network* (*Fast* R-CNN) dan *Faster Region-Convolutional Neural Network* (Faster R-CNN) pada ranah deteksi objek juga menandai perkembangan yang sangat pesat terhadap metode CNN [7].

Mask Region-Convolutional Neural Network (Mask-RCNN) merupakan metode hasil dari pengembangan dari *Faster*

Region-Convolutional Neural Network (Faster R-CNN) yang diberikan beberapa penambahan. Metode ini mendukung *instance segmentation* yang dapat mendeteksi objek-objek pada sebuah gambar ataupun video hingga tingkat piksel [6].

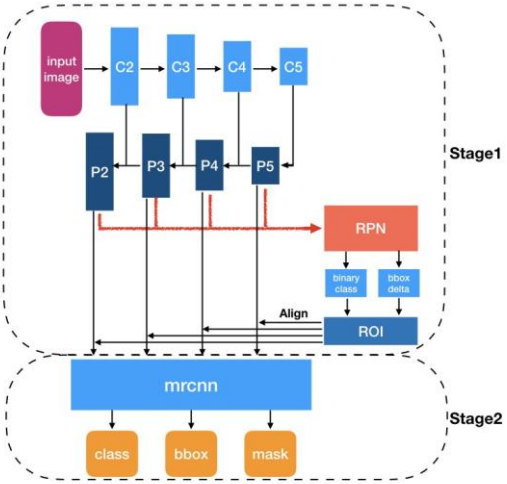
Perbedaan mendasar antara Mask R-CNN dengan pendahulunya yaitu Faster R-CNN terletak pada penambahan sebuah *segmentation mask branch* dan perubahan pada lapisan *Region of Interest* (RoI) [6]. Pada Faster R-CNN digunakan RoIPool untuk menghasilkan *fixed featured map*. Namun pada Mask R-CNN digunakan RoIAlign untuk menghasilkan *fixed featured map*. Perbedaan arsitektur dapat dilihat pada Gambar 2.2.



Gambar 2.2 Perbandingan Mask R-CNN dan Faster R-CNN [7]

Ada 2 langkah utama pada *Mask R-CNN* seperti pada Gambar 2.3. Pertama, metode ini akan memroses gambar masukan untuk diekstraksi fiturnya sehingga didapatkan bagian objek yang

diusulkan menjadi calon objek yang dideteksi dan nilai *features map* masukan. Sedangkan langkah utama kedua proses prediksi kelas dari objek. Proses ini juga menghasilkan *bounding box* dan menghasilkan *mask* berupa pewarnaan objek dalam tingkat piksel [6].



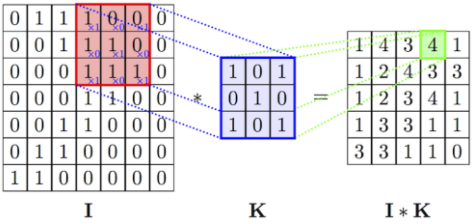
Gambar 2.3 Dua Tahap utama Mask R-CNN

Untuk langkah-langkah utama yang terdapat pada 2 tahap utama ini terdapat 4 langkah yang akan dibahas pada subbab dibawah yaitu *Backbone Model*, *Pooling Layer*, *Region Proposal Network*, dan *Segmentation Mask*.

2.2.1 Backbone Model

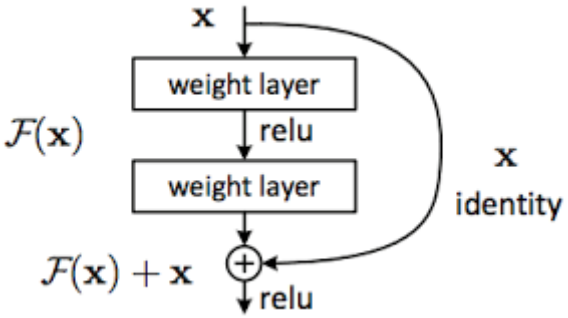
Convolution Layer melakukan operasi konvolusi pada output dari lapisan sebelumnya. Konvolusi adalah istilah matematis yang artinya mengaplikasikan sebuah fungsi pada *output* fungsi lain secara berulang. Tujuan dilakukannya konvolusi pada data citra adalah untuk mengekstrak fitur dari citra masukan.

Ilustrasi cara kerja konvolusi bisa dilihat pada Gambar 2.4, dimana I adalah citra, K adalah *filter* atau *kernel* yang digunakan, I * K adalah hasil operasi konvolusi [10].



Gambar 2.4 Ilustrasi cara kerja konvolusi

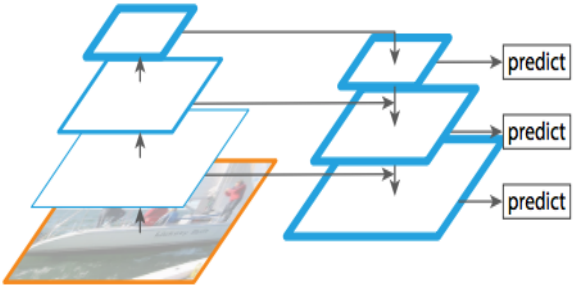
Pada *Backbone Model* yang terdapat di metode ini, arsitektur ResNet-101 dipilih untuk memenuhi lapisan konvolusinya sebagai ekstraktor fitur. Lapisan awalnya akan mendeteksi fitur *low-level* yaitu bagian tepi dan pojokan masukan gambar. Setelah itu, baru lapisan ini akan mendeteksi fitur *high-level* yaitu objek-objek yang terdapat pada gambar [11]. Bentuk umum dari arsitektur ResNet-101 ini dapat dilihat seperti pada Gambar 2.5.



Gambar 2.5 Bentuk Umum arsitektur ResNet-101

Selanjutnya pada *Backbone Model* terdapat *Feature Pyramid Network* (FPN) yang digunakan untuk meningkatkan kinerja model. FPN dapat merepresentasikan objek lebih baik dengan berbagai skala. FPN meningkatkan piramida ekstraksi fitur

standar dengan menambahkan piramida kedua yang mengambil fitur tingkat tinggi dan meneruskannya ke lapisan piramida ekstraktor fitur yang lebih rendah. [12] Hal ini dapat memungkinkan fitur di setiap level memiliki akses ke fitur level yang lebih rendah dan lebih tinggi. Ilustrasi dari FPN dapat dilihat pada Gambar 2.6.



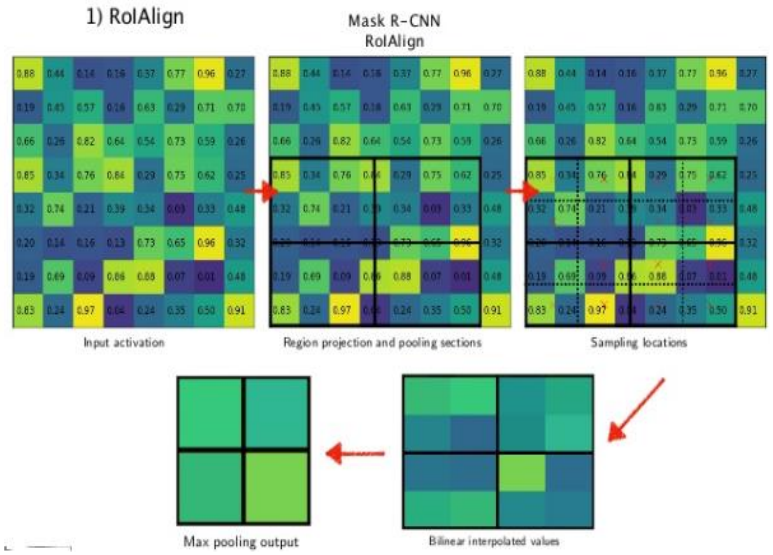
Gambar 2.6 Bentuk Umum Lapisan *Feature Pyramid Network* [13]

2.2.2 Pooling Layer

Pooling Layer pada Mask R-CNN digunakan untuk mereduksi data citra yang telah dijadikan *features map* menjadi sebuah *features map* dengan besaran yang tetap. *Pooling Layer* yang digunakan untuk mendapatkan *Region of Interest* (RoI) dari *features map* adalah RoIAlign.

Pada penggunaan RoI, batas sel dari *features map* hasil dari *Pooling Layer* disetel kembali sehingga besaran batas selnya sama besar dengan *features map* masukan citra. RoIAlign bertugas untuk membuat sel didalam hasil reduksi *features map* memiliki ukuran yang sama dengan besaran batas sel *features map* awal. Interpolasi dilakukan untuk menghasilkan nilai baru untuk setiap sel *features map* akhir sehingga ketika ukurannya berubah, nilai yang didalamnya tetap mewakili nilai-nilai dari *features map*

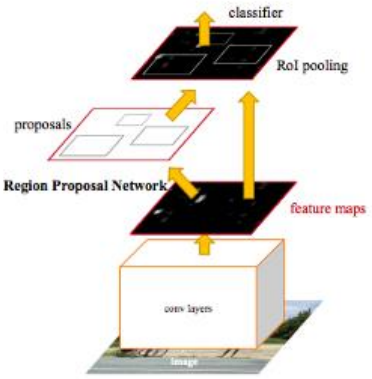
sebelumnya [6]. Contoh dari penggunaan RoIAlign pada reduksi fitur dapat dilihat pada gambar 2.7.



Gambar 2.7 Ilustrasi cara kerja *RoiAlign pada Pooling Layer* [14]

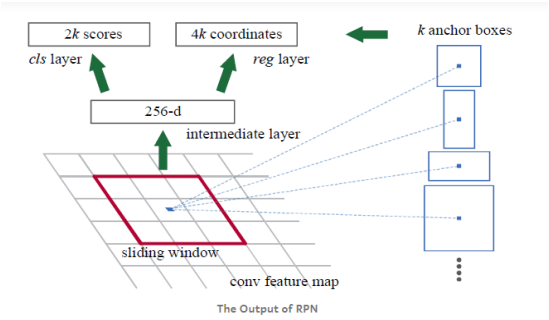
2.2.3 Region Proposed Network

Region Proposed Network (RPN) adalah kumpulan bagian dari gambar yang diusulkan menjadi wilayah deteksi objek. Jaringan ini hanya dapat melihat peta fitur konvolusional terakhir dan menghasilkan proposal wilayah darinya (Gambar 2.8)



Gambar 2.8 Letak *Region Proposal Network* pada arsitektur [15]

RPN memiliki sebuah *classifier* dan regresor. *Classifier* akan menghasilkan probabilitas dari sebuah usulan menjadi objek target. Regresor pada RPN melakukan regresi pada koordinat setiap usulan [16] (Gambar 2.9).



Gambar 2.9 Ilustrasi *Region Proposed Network* pada arsitektur

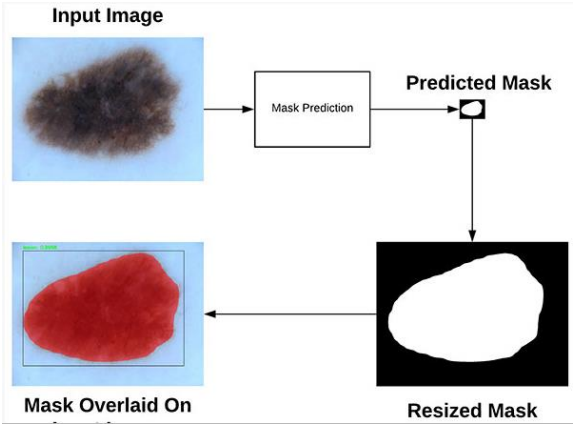
Dalam pencarian usulan yang diajukan, pergeseran dalam gambar untuk menjadi besaran bagian objek dipengaruhi oleh

anchor box. Bagian ini menyimpan informasi besaran pergesran kotak pendeteksian objek. Adapun properti yang menjadi penting didalamnya adalah besaran kotak *anchor*, panjang dan tinggi pergeseran (stride), dan skala rasio besaran kotak *anchor* [17].

2.2.4 Segmentation Mask

Segmentation mask adalah jaringan konvolusi yang mengambil nilai bagian positif dari *RoI Classifier* dan menghasilkan *mask* dari bagian tersebut. *Mask* yang dihasilkan memiliki resolusi rendah yaitu 28x28 [6]. Namun pada representasi *soft mask* memiliki nilai desimal sehingga memiliki nilai detil dari pada *mask* bernilai biner. Jadi nilai-nilai desimal tersebut dapat merepresentasikan nilai cahaya dengan tepat.

Pada saat proses RoI, *ground-truth* diperkecil hingga memiliki besaran 28x28 dan diambil nilai *loss* objek sehingga didapat bagian *mask* dari objek. Lalu ketika meletakkan hasil *mask*, besaran 28x28 akan diperbesar sehingga memiliki bagian sebesar gambar objek aslinya [6]. Ilustrasi dari proses *masking* ini dapat dilihat pada Gambar 2.10.



Gambar 2.10 Proses pada *segmentation mask* [6]

2.3 Akurasi, Precision & Recall

Ketika membangun sebuah model klasifikasi, pertanyaan yang muncul adalah bagaimana mengetahui seberapa baik model tersebut. Oleha karena itu, diperlukan evaluasi terhadap uji coba model. Ada 3 poin yang dapat dievaluasi yaitu akurasi, *precision*, dan *recall*.

Akurasi merupakan rasio prediksi Benar (positif dan negatif) dengan keseluruhan data. *Precision* merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positif. Sedangkan *recall* merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif.

Mengevaluasi model klasifikasi dilakukan dengan mencari tahu seberapa baik hasil prediksi dari model tersebut. *Recall* di Persamaan (2.1), *precision* di Persamaan (2.2), dan akurasi di Persamaan (2.3) adalah metode pengukuran yang biasa digunakan dalam mengevaluasi model, penjelasan variabel ada pada *confusion matrix* pada Tabel 2.1. Namun pada evaluasi penulisan ini tidak dikenali nilai *true negative* disebabkan nilai *true negative* di penulisan ini adalah dimana pada gambar tidak ada jenis kendaraan yang ingin dikenali dan tidak terdeteksi.

Tabel 2.1 *Confusion matrix*

		Kelas Prediksi	
		Benar	Salah
Kelas sebenarnya	Benar	*True Positive* (TP)	*False Negative* (FN)
	Salah	*False Positive* (FP)	-

Keterangan :

1. *True Positive* (TP) pada gambar dikenali jenis kendaraan, pada keluaran program jenis kendaraan tersebut.
2. *False Positive* (FP) pada gambar dikenali jenis kendaraan, pada keluaran program tidak mengenali jenis kendaraan.

3. *False Negative* (FN) pada gambar tidak dikenali jenis kendaraan, pada keluaran program mengenali jenis kendaraan.

$$Recall = TP / (TP + FN) \quad (2.1)$$

$$Precision = TP / (TP + FP) \quad (2.2)$$

$$Akurasi = (TP) / (TP + FP + FN) \quad (2.3)$$

2.4 Python

Python adalah bahasa pemograman yang populer. *Python* sering dimanfaatkan dalam pengembangan web, perangkat lunak, penelitian, dan *system scripting*. Python dapat digunakan untuk menangani data besar dan melakukan operasi matematika yang kompleks. Python bekerja di berbagai *platform* seperti Windows, Mac, Linux, Raspberry Pi, dan lain-lain. Python dirancang untuk mudah dibaca, yaitu memiliki sintaks yang sederhana dan menggunakan bahasa inggris [18].

2.4.1 Keras

Keras adalah *high-level neural networks* API, yang ditulis dalam bahasa pemograman Python dan mampu berjalan di atas TensorFlow dan Theano. Keras dikembangkan dalam rangka memungkinkan eksperiman dilakukan dengan cepat. Keras dapat berjalan baik di CPU dan GPU. Keras berisi banyak implementasi *neural network* yang umum digunakan, fungsi aktivasi, *optimizer*, dan *tool* lain yang memudahkan dalam pengolahan citra dan data teks [19].

2.4.2 TensorFlow

TensorFlow adalah *library open source* untuk pembuatan program yang membutuhkan komputasi numerik berkinerja tinggi. TensorFlow dikembangakan oleh tim Google Brain. TensorFlow menyediakan fungsi-fungsi *machine learning* dan *deep learning*, dan dapat dijalankan dalam CPU atau GPU [20].

2.4.3 OpenCV

OpenCV (*Open Source Computer Vision*) adalah *library* yang dimanfaatkan dalam pengolahan citra dinamis secara *real-time*. OpenCV dapat digunakan dalam berbagai bahasa pemograman seperti Python, C++, Java, atau MATLAB. OpenCV memiliki fitur seperti *Feature & Object Detection, Motion Analysis and Object Tracking, Image Filtering, Image Processing*, dan lain-lain [21].

2.4.4 Numpy

Numpy adalah *library Python* yang mendukung pengolahan data pada *array* dan matriks multidimensi yang besar. Numpy menyediakan kumpulan fungsi matematika, seperti aljabar linear, transformasi Fourier, pembuatan angka acak, dan lain-lain. *Numpy* bersifat *open source* sehingga banyak dimanfaatkan dalam pengolahan data peneilitian [22].

2.4.5 Scikit-learn

Scikit-learn adalah *open source machine learning library* untuk bahasa pemograman Python. Scikit-learn menyediakan fitur seperti *classification*, *regression*, *clustering*, termasuk juga didalamnya algoritma *support vector machines*, *random forest*, *gradient boosting*, dan lain-lain [23].

2.4.6 MoviePy

MoviePy adalah *library* yang disediakan Python yang digunakan untuk menyunting, memotong, dan menggabungkan video yang dijadikan masukan. Library ini sangat berfungsi untuk proses deteksi video sehingga didapatkan setiap *frame* pada video [24].

BAB III
PERANCANGAN SISTEM

Bab ini menjelaskan mengenai perancangan data dan sistem deteksi kendaraan berbasi video menggunakan *Mask* R-CNN. Bab ini juga akan menjelaskan gambaran umum sistem dalam bentuk diagram alir.

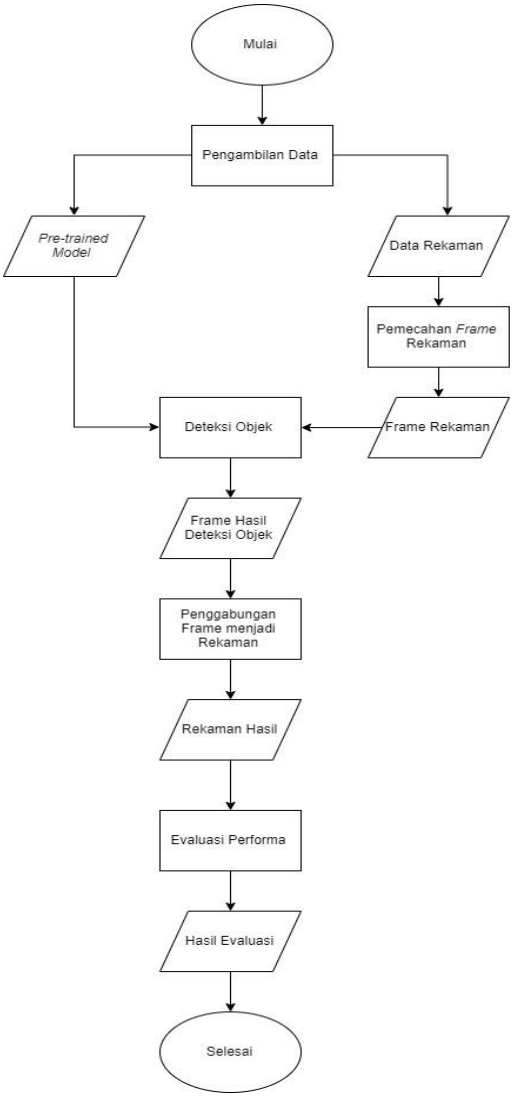
3.1 Perancangan Data

Data yang digunakan pada sistem ini ada 2 yaitu data *pre-trained model* sebagai data model dan data rekaman sebagai *testing*. Untuk data model dipilih dari model *pre-trained model* dari dataset MS-COCO yang memiliki 80 kelas. Model ini telah memiliki bobot dan telah diklasifikasi menggunakan Mask R-CNN.

Untuk data *testing*, digunakan data rekaman yang diambil secara manual dengan durasi 40 detik dan tingkat *frame per second* sebesar 30 FPS. Data rekaman akan direduksi sesuai kebutuhan pengujian.

3.2 Desain Umum Sistem

Sistem pendeteksian kendaraan yang dibangun memiliki proses utama diantaranya pelatihan dan pengujian *Mask* R-CNN. Diagram alir dari sistem ditunjukan pada Gambar 3.3.



Gambar 3.1 Diagram alir sistem yang dibangun

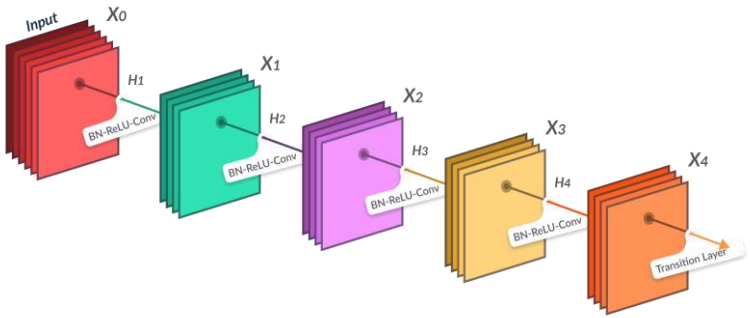
Pada arsitektur sistem ini, pertama kali yang dilakukan adalah proses masukan model *pre-trained* dari MS-COCO sebagai model deteksi objek. Lalu dilakukan pengambilan data rekaman yang selanjutnya *frame* pada rekaman dijadikan data *testing*. Kedua data ini menjadi masukan terhadap deteksi objek pada sistem.

Selanjutnya, hasil dari deteksi ini akan digabungkan kembali menjadi video keluaran dengan tingkat *frame* per detik yang sama. Terakhir, evaluasi dilakukan terhadap video keluaran untuk mengetahui tingkat keberhasilan dari sistem.

3.2.1 Tahap Pembangunan Arsitektur

Pembangunan arsitektur pada sistem ini dibagi menjadi beberapa tahap. Tahap pada arsitektur ini mencakup model *backbone*, *region proposal network*, ROI *Align Layer*.

Pada tahap model *backbone*, model CNN dari ResNet-101 digunakan untuk ekstraksi fitur. *Backbone Model* yang digunakan adalah model dari ResNet yang konfigurasinya dapat dilihat seperti pada Gambar 3.2.



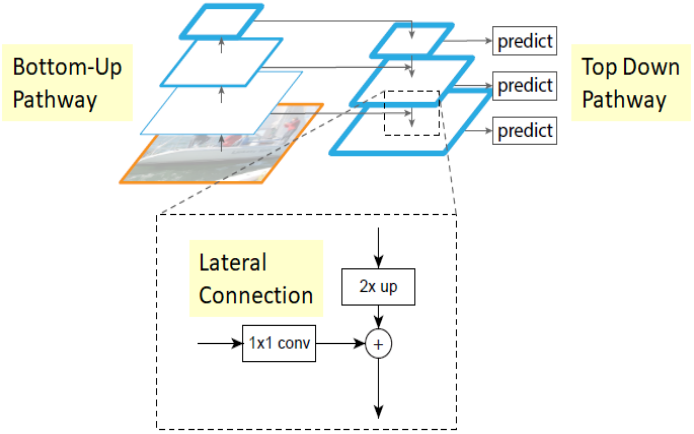
Gambar 3.2 Arsitektur CNN dari ResNet [25]

Pada metode ini, penguji mencoba 2 jenis ResNet untuk ujicoba, yaitu ResNet-50 dan ResNet-101. Untuk konfigurasi kedua *backbone model* ini dapat dilihat pada Tabel 3.1.

Tabel 3.1 Konfigurasi model ResNet

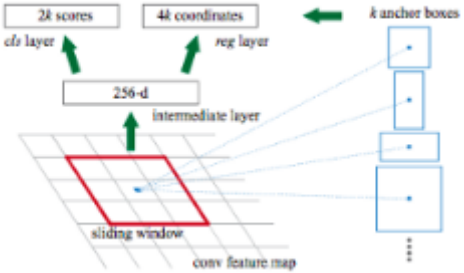
Model	Layer	Ukuran Input	Ukuran Output	Spesifikasi
RN-101 RN-50	Conv1	600x400	112x112	Window = 7x7 Filter = 64 Stride = 2
RN-101 RN-50	Conv2_x	112x112	56x56	3 layer berisi: 1x1, 64 3x3, 64 1x1, 256
RN-101 RN-50	Conv3_x	56x56	28x28	4 layer berisi: 1x1, 128 3x3, 128 1x1, 512
RN-101	Conv4_x	28x28	14x14	23 layer berisi: 1x1, 256 3x3, 256 1x1, 1026
RN-50				6 layer berisi: 1x1, 256 3x3, 256 1x1, 1026
RN-101 RN-50	Conv5_x	14x14	7x7	3 layer berisi: 1x1, 512 3x3, 512 1x1, 2048
RN-101 RN-50	x	7x7	1x1	Align Pooling

Adapun untuk gambar arsitektur untuk *feature pyramid network* seperti pada gambar 3.3



Gambar 3.3 Arsitektur *feature pyramid network* [15]

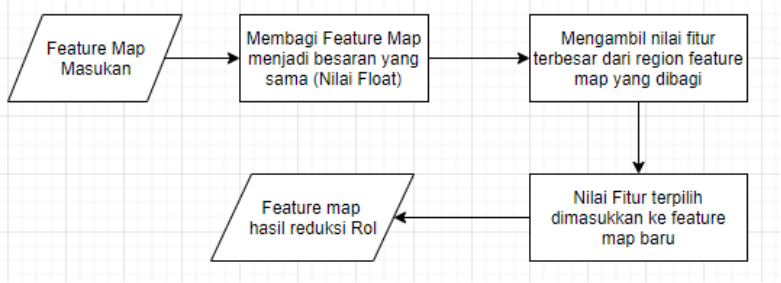
Pada langkah *bottom-up*, dilakukan *feedforward* terhadap hasil dari setiap komputasi lapisan dengan dimensi 1x1 lapisan konvolusi. Output dari lapisan terakhir dari setiap tahap akan digunakan sebagai set referensi peta fitur untuk memperkaya jalur *top-down* dengan koneksi lateral. Untuk langkah *top-down*, akan menghasilkan set referensi untuk dimensi yang lebih besar.



Gambar 3.4 Arsitektur *Region Proposal Network* [15]

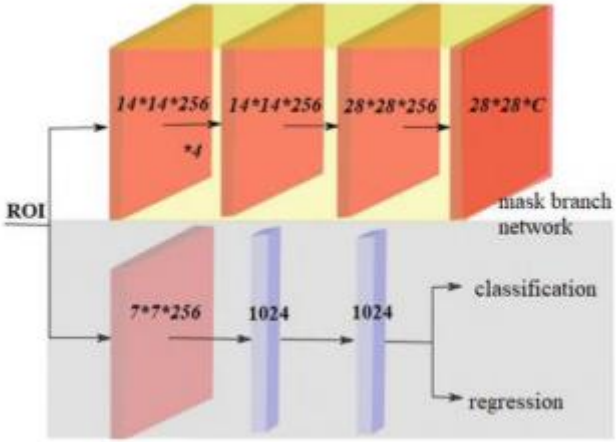
Selanjutnya, untuk *region proposal network* seperti pada Gambar 3.4, setiap pergeseran *window* sesuai dengan besaran kotak *anchor* akan dihasilkan keluaran berupa *score* dari lapisan *classification* yang merepresentasikan *Region of Interest* atau *background*. Lalu dari lapisan *regression*, dihasilkan 4 koordinat untuk menyimpan letak dari fitur

Setelah langkah diatas, arsitektur selanjutnya yaitu *ROI Align Layer* dapat dilihat pada Gambar 3.5. Peta fitur dari hasil *backbone* akan dibagi menjadi besaran yang sama. Nilai fitur yang telah dibagi akan direduksi dengan dilakukannya interpolasi sehingga hasil yang didapat masih dalam bentuk nilai *float*.



Gambar 3.5 Diagram Alur RoI dalam reduksi fitur

Lapisan selanjutnya ada lapisan *segmentation mask* seperti pada Gambar 3.6. Dari setiap peta fitur yang berisi *region of interest* dan *background*, peta fitur yang mengandung nilai *region of interest* akan diambil untuk dijadikan piksel terpilih dalam *segmentation mask*. Fitur ini dilakukan konvolusi sehingga didapatkan besaran mask berupa 28*28.



Gambar 3.5 Arsitektur *Segmentation Mask* [6]

3.2.2 Tahap Pengujian

Dalam melakukan tahap pengujian, digunakan rekaman yang telah diambil *frame-per-frame* pada rekaman tersebut untuk dijadikan data uji. Data uji dilakukan terhadap model yang telah diambil dari model *pre-trained* MS-COCO. Pengujian disesuaikan dengan scenario uji coba untuk penentuan besaran *batch* ujicoba.

Evaluasi terhadap hasil uji coba dilakukan manual dikarenakan data rekaman yang diambil tidak memiliki label per objek yang berada didalamnya. Evaluasi dilakukan dengan cara menghitung tingkat kebenaran deteksi dari suatu frame dan dibandingkan dengan label objek sesungguhnya. Dari hasil evaluasi ini akan didapatkan akurasi, *precision*, dan *recall* per scenario uji coba.

(Halaman ini sengaja dikosongkan)

BAB IV
IMPLEMENTASI

Bab ini menjelaskan mengenai implementasi perangkat lunak dari rancangan sistem yang telah dibahas pada Bab 3 meliputi kode program dalam perangkat lunak. Selain itu, implementasi dari tiap proses, parameter masukan, keluaran, dan beberapa keterangan yang berhubungan dengan program juga dijelaskan.

4.1 Lingkungan Implementasi

Dalam mengimplementasikan aplikasi pengenalan ekspresi manusia diperlukan beberapa perangkat pendukung sebagai berikut.

4.1.1 Perangkat Keras

Implementasi tugas akhir ini menggunakan desktop *personal computer* (PC) MS-7886. Sistem operasi yang digunakan adalah Windows 10 64-bit. PC yang digunakan memiliki spesifikasi AMD Ryzen 7 2700 dengan kecepatan 3,2 GHz, *Random Access Memory* (RAM) sebesar 16 GB, dan mempunyai *Graphics Processing Unit* (GPU) yaitu NVIDIA GeForce RTX 2080 sebesar 8 GB.

4.1.2 Perangkat Lunak

PC dari sisi perangkat lunak memiliki spesifikasi antara lain menggunakan bahasa pemograman Python 3.6, dilengkapi dengan *library* antara lain OpenCV, Tensorflow-GPU, Keras-GPU, Numpy, MoviePy dan Scikit-learn.

4.2 Implementasi Pembangunan Arsitektur

Pada subbab ini akan dijabarkan implementasi fungsi-fungsi pada tahap pembangunan model. Arsitektur Mask R-CNN dimulai dengan tahap implementasi graf arsitektur CNN yang

menggunakan ResNet-101. Adapun implementasi nya dapat dilihat pada Kode Sumber 4.1.

```
1.  Def resnet_graph(input_image, architecture,
    stage5=False, train_bn=True):
2.  #Tahap 1 Konvolusi
3.      x = KL.ZeroPadding2D((3, 3))(input_image)
4.      x = KL.Conv2D(64, (7, 7), strides=(2, 2),
            name='conv1', use_bias=True)(x)
5.      x = BatchNorm(name='bn_conv1')(x,
            training=train_bn)
6.      x = KL.Activation('relu')(x)
7.      C1 = x = KL.MaxPooling2D((3, 3), strides=(2,
             2), padding="same")(x)
8.  #Tahap 2 Konvolusi
9.      x = conv_block(x, 3, [64, 64, 256], stage=2,
            block='a', strides=(1, 1), train_bn =
            train_bn)
10.     x = identity_block(x, 3, [64, 64, 256],
            stage=2, block='b', train_bn=train_bn)
11.     C2 = x = identity_block(x, 3, [64, 64, 256],
             stage=2, block='c', train_bn=train_bn)
12. #Tahap 3 Konvolusi
13.     x = conv_block(x, 3, [128, 128, 512], stage=3,
            block='a', train_bn=train_bn)
14.     x = identity_block(x, 3, [128, 128, 512],
            stage=3, block='b', train_bn=train_bn)
15.     x = identity_block(x, 3, [128, 128, 512],
            stage=3, block='c', train_bn=train_bn)
16.     C3 = x = identity_block(x, 3, [128, 128, 512],
             stage=3, block='d', train_bn=train_bn)
17. #Tahap 4 Konvolusi
18.     x = conv_block(x, 3, [256, 256, 1024], stage=4,
            block='a', train_bn=train_bn)
19.     block_count = { "resnet101":22}[architecture]
20.     for i in range(block_count):
21.         x = identity_block(x, 3, [256, 256, 1024],
                stage=4, block=chr(98 + i), train_bn =
                train_bn)
22.     C4 = x
```

```
#Tahap 5 Konvolusi
    if stage5:
        x = conv_block(x, 3, [512, 512, 2048],
            stage=5, block='a', train_bn=train_bn)
        x = identity_block(x, 3, [512, 512, 2048],
            stage=5, block='b', train_bn=train_bn)
        C5 = x = identity_block(x, 3, [512, 512,
             2048], stage=5, block='c', train_bn =
             train_bn)
    else:
        C5 = None
#Keluaran Hasil Konvolusi
    return [C1, C2, C3, C4, C5]
```

Kode Sumber 4.1 Fungsi pembangunan lapisan konvolusi ResNet-101

Selanjutnya keluaran dari implementasi akan dijadikan sebagai masukan terhadap implementasi selanjutnya yaitu lapisan *region proposal network*. Adapun implementasi pembuatan model graf lapisan ini dapat dilihat pada Kode Sumber 4.2

```
    def rpn_graph(feature_map, anchors_per_location,
    anchor_stride):wave_convolution_layer(inputs1,
    127)
    #shared convolutional menjadi dasar RPN
        shared = KL.Conv2D(512, (3, 3), padding =
                 'same', activation='relu', strides
                 = anchor_stride,
        x = KL.Conv2D(2 * anchors_per_location,
            (1, 1), padding='valid', activation =
            'linear', name='rpn_class_raw')(shared)
        rpn_class_logits = KL.Lambda(lambda t:
        tf.reshape(t, [tf.shape(t)[0], -1, 2]))(x)
    #softmax untuk penentuan fore/back ground
        rpn_probs = KL.Activation("softmax", name =
        "rpn_class_xxx")(rpn_class_logits)
```

Kode Sumber 4.2 Fungsi pembangunan graf RPN

Lapisan selanjutnya *feature pyramid network* dapat dilihat pada implementasi dibawah ini pada Kode Sumber 4.3

```
1.  Def build_fpn_mask_graph(rois, feature_maps,
    image_meta, pool_size, num_classes, train_bn=True):
2.      x = PyramidROIAlign([pool_size, pool_size],
            name="roi_align_mask")([rois, image_meta] +
            feature_maps)
3.  #Lapisan Konvolusi FPN
4.      x = KL.TimeDistributed(KL.Conv2D(256, (3, 3),
            padding="same"), name="mrcnn_mask_conv1")(x)
5.      x = KL.TimeDistributed(BatchNorm(), name=
            'mrcnn_mask_bn1') (x, training=train_bn)
6.      x = KL.Activation('relu')(x)
7.      x = KL.TimeDistributed(KL.Conv2D(256, (3, 3),
            padding="same"),name="mrcnn_mask_conv2" )
            (x)
8.      x = KL.TimeDistributed(BatchNorm(),
            name='mrcnn_mask_bn2')(x, training=
            train_bn)
9.      x = KL.Activation('relu')(x)
```

Kode Sumber 4.3 Implementasi Pembangunan FPN

Setelah membahas implementasi pembangunan FPN, terdapat implementasi RoIAlign Pada implementasi selanjutnya ada implementasi pembangunan RoIAlign yang terdapat pada Kode Sumber 4.4

```
1.  roi_level = log2_graph(tf.sqrt(h * w) / (224.0 /
    tf.sqrt(image_area))), image_meta, pool_size,
    num_classes, train_bn=True):
2.  pooled.append(tf.image.crop_and_resize
    feature_maps[i], level_boxes, box_indices,
    self.pool_shape, method="bilinear”))
3.  pooled = tf.concat(pooled, axis=0)
            'mrcnn_mask_bn1') (x, training=train_bn)
4.  box_to_level = tf.concat(box_to_level, axis=0)
5.  box_range = tf.expand_dims(tf.range
    (tf.shape(box_to_level) [0]), 1)
```

```
6.      box_to_level = tf.concat ([tf.cast(
        box_to_level, tf.int32), box_range])
```

Kode Sumber 4.4 Implementasi Region of Interest Align

Setelah implementasi diatas, diperlukan implementasi penerapan *segmentation mask* seperti pada Kode 4.5

```
1.      boxes = positive_rois
2.      y1, x1, y2, x2 = tf.split(positive_rois, 4,
        axis=1)
3.      gt_y1, gt_x1, gt_y2, gt_x2 =
        tf.split(roi_gt_boxes, 4, axis=1)
4.      gt_h = gt_y2 - gt_y1
5.      gt_w = gt_x2 - gt_x1
6.      y1 = (y1 - gt_y1) / gt_h
7.      x1 = (x1 - gt_x1) / gt_w
8.      y2 = (y2 - gt_y1) / gt_h
9.      x2 = (x2 - gt_x1) / gt_w
10.     boxes = tf.concat([y1, x1, y2, x2], 1)
11.     box_ids = tf.range(0, tf.shape(roi_masks)
                  [0])
12.     masks = tf.image.crop_and_resize (tf.cast
                (roi_masks, tf.float32), boxes,
                box_ids, config.MASK_SHAPE)
13.     masks = tf.squeeze(masks, axis=3)
14.     masks = tf.round(masks)
```

Kode 4.5 Implementasi *Segmentation Mask*

Keseluruhan implementasi diatas memiliki beberapa parameter global penting. Berikut penjelasan parameter-parameter tersebut:

1. Parameter *filters* adalah jumlah *filter* atau *kernel*.
2. Parameter *kernel_size* adalah ukuran *filter* atau *kernel*.
3. Parameter *strides* adalah ukuran *stride* yang digunakan.
4. Parameter *acitivation* adalah fungsi aktivasi apa yang digunakan, terdapat banyak pilihan fungsi aktivasi yang

disediakan oleh Keras, salah satunya adalah ‘relu’ yakni fungsi ReLU.
5. Parameter *anchor* untuk besaran *stride*.
6. Parameter *feature_map* untuk peta fitus dari input hingga melewati beberapa tingkatan lapisan

4.3 Implementasi Pengujian dan Evaluasi

Setelah pembangunan arsitektur *Mask Region-Convolutional Neural Network*, dilakukan proses pengujian. Adapun langkah ini pada implementasi mencakup pemasukan model yang telah dilatih sebagai network, pemecahan video sebagai *frame* yang terpisah, pengujian terhadap *frame* rekaman, dan penyatuan *frame*.

Pemasukan model yang telah dilatih dapat dilihat implementasinya pada Kode Sumber 4.6

```
1.  COCO_MODEL_PATH = os.path.join('',
    "mask_rcnn_coco.h5")
2.  model = modellib.MaskRCNN(mode="inference",
    model_dir='mask_rcnn_coco.hy', config=config)
3.  model.load_weights('mask_rcnn_coco.h5',
    by_name=True)
```

Kode Sumber 4.6 Input Pre-Trained Model

Lalu video yang telah diambil dilakukan pengujian seperti yang diimplemtasikan pada Kode Sumber 4.7.

```
1.  img = cv2.resize(input_img, (1024, 1024))
2.  img = image.img_to_array(img)
3.  results = model.detect([img], verbose=0)
4.  r = results[0]
5.  final_img  =
    visualize_car_detection.display_instances2(img,
```

```
    r['rois'], r['masks'], r['class_ids'], class_names,
    r['scores'])
6.  inp_shape = image.img_to_array(input_img).shape
7.  final_img = cv2.resize(final_img, (inp_shape[1],
    inp_shape[0]))
8.  return final_img
```

Kode Sumber 4.7 Implementasi pengujian

Seleksi kelas selanjutnya dilakukan dengan fungsi display_instances2 seperti pada Kode Sumber 4.8

```
1.  def display_instances2(image, boxes, masks,
    class_ids, class_names, scores=None, title="",
    figsize=(16, 16), ax=None):
2.  if class_ids[i] == “Car” or class_ids[i] ==
    “Motorcycle” or class_ids[i] == “Bus” or
    class_ids[i] == “Truck”
3.  label = class_names[class_id]
4.  mask = class_masks[class_id]
5.  color = random(rgb)
6.  masked_image = apply_mask(masked_image, mask,
    color)
7.  return masked_image.astype(np.uint8)
```

Kode Sumber 4.8 Implementasi seleksi kelas

Pada tahap evaluasi terhadap pengujian, dilakukan evaluasi manual terhadap setiap *frame* pada rekaman sehingga didapatkan nilai kebenaran dari setiap *frame*. Nilai kebenaran setiap objek yang dicari akan menjadi masukan terhadap evaluasi. Dalam penghitungan nilai kebenaran deteksi objek, dilakukan pengujian terhadap 100 gambar untuk mendapatkan parameter terbaik.

Setelah didapatkan parameter optimal untuk pengujian, pengujian dilakukan dengan membandingkan metode lain pada data rekaman. Data rekaman yang diujikan disesuaikan dengan tingkat FPS yang membuat metode Mask-RCNN menjadi *realtime.*

Evaluasi pada implementasi ini didapat nilai dari akurasi, presisi, dan *recall*.

BAB V
UJI COBA DAN EVALUASI

Bab ini akan membahas mengenai hasil uji coba sistem yang telah dirancang dan dibuat. Uji coba dilakukan untuk mengetahui kinerja sistem dengan lingkungan uji coba yang telah ditentukan.

5.1 Lingkungan Uji Coba

Lingkungan uji coba pada tugas akhir ini adalah sebuah desktop *personal computer* (PC) MS-7886. Sistem operasi yang digunakan adalah Windows 10 64-bit. PC yang digunakan memiliki spesifikasi perangkat keras AMD Ryzen 7 2700 dengan kecepatan 3,2 GHz, *Random Access Memory* (RAM) sebesar 16 GB, dan mempunyai *Graphics Processing Unit* (GPU) yaitu NVIDIA GeForce RTX 2080 sebesar 8 GB. Pada sisi perangkat lunak, uji coba pada tugas akhir ini dilakukan dengan menggunakan bahasa pemograman Python 3.6 dilengkapi dengan *library* antara lain Keras, Tensorflow, OpenCV, Numpy, Matplotlib dan Scikit-learn.

5.2 Dataset

Pada tugas akhir ini, sistem menggunakan dataset MS-COCO 2017 yang telah dibentuk menjadi sebuah model *pre-trained*. Dataset MS-COCO 2017 terdiri dari 118.000 gambar dan dibagi menjadi 81 kelas. Data yang diambil adalah data *weight* dari model *pre-trained*.

Data pengujian untuk menemukan parameter terbaik untuk model adalah 100 gambar yang mengandung 4 kelas objek pada tugas akhir ini. Data gambar diambil dari detik ke 12 sampai dengan detik ke 16 dari rekaman yang diambil manual. Data pengujian ini digunakan pada skenario uji coba 1 dan 2. Parameter terbaik selanjutnya menjadi parameter metode Mask R-CNN untuk diujikan terhadap metode lain.

Selanjutnya, data pengujian untuk perbandingan metode adalah rekaman lalu lintas yang diambil manual. Data pengujian

ini digunakan pada skenario uji coba 3 dan 4. Rekaman memiliki durasi 40 detik dan memiliki tingkat *frame per second* (FPS) 30 FPS. Pada uji coba perbandingan metode, diujikan rekaman yang memiliki tingkat FPS yang membuat metode Mask-RCNN menjadi *realtime*.

5.3 Skenario Uji Coba

Proses uji coba berguna untuk menemukan parameter yang menghasilkan performa model yang paling optimal. Parameter yang tepat akan memberikan hasil yang lebih baik pada saat proses uji coba. Hasil terbaik dari suatu skenario uji coba akan digunakan untuk skenario uji coba berikutnya. Ada 4 macam skenario uji coba dan semuanya akan dicoba pada arsitektur Mask-RCNN yang telah dirancang. Skenario uji coba yang akan dilakukan yaitu:

1. Uji coba berdasarkan *backbone model*.
2. Uji coba berdasarkan *detection minimum confidence.*
3. Uji coba perbandingan metode *deep learning*.
4. Uji coba perbandingan metode *machine learning*.

Tabel 5.1 berisi parameter-parameter awal arsitektur CNN yang digunakan dan dapat berubah di setiap uji coba yang dilakukan. Pada setiap skenario uji coba akan ditetapkan nilai parameter yang dapat meningkatkan kinerja arsitektur.

Tabel 5.1 Parameter awal yang digunakan dalam arsitektur

Keterangan	**Parameter**
Steps Per Epoch	1000
Ukuran *batch*	2
Feature Reduction	*RoIAlign*
Learning rate	0,001

Adapun untuk hasil deteksi Mask R-CNN, warna kotak menandai nama kelas tersebut. Warna hijau untuk kelas motor, warna merah untuk kelas mobil, warna putih untuk kelas truk dan warna biru untuk kelas bus. Untuk pewarnaan *color splash* pada

piksel objek akan diberikan warna secara acak untuk menandai perbedaan antara objek yang dideteksi.

5.3.1 Uji Coba Berdasarkan Backbone Model

Pada uji coba ini, dilakukan percobaan membandingkan penggunaan arsitektur Backbone Model yang berbeda pada metode Mask R-CNN. Adapun Backbone Model yang dibandingkan adalah:

1. ResNet-50
2. ResNet-101

Adapun hasil dari ujicoba kedua model dapat dilihat dalam Tabel 5.2

Tabel 5.2 Hasil Evaluasi *Backbone Model*

Model	Frame	Durasi	Akurasi	Kelas	Precision	Recall
ResNet-50	100	61,7 s	50,67%	Mobil	79,82%	84,67%
				Motor	100,00%	9,38%
				Bus	61,11%	38,60%
				Truk	59,13%	70,10%
ResNet-101	100	61,7 s	**62,04%**	Mobil	82,82%	91,48%
				Motor	100,00%	71,88%
				Bus	100,00%	69,01%
				Truk	74,26%	77,32%

Untuk pengujian parameter berikutnya, *backbone model* ResNet-101 dipilih karena memiliki hasil yang lebih optimal.

5.3.2 Uji Coba Berdasarkan Detection Minimum Confidence

Pada uji coba ini, dilakukan perbandingan terhadap nilai *Detection Minimum Confidence* (DMC). Nilai ini merupakan *threshold* terhadap nilai *confidence* untuk menampilkan hasil deteksi. Adapun nilai DMC yang dibandingkan seperti berikut:

1. *Detection Minimum Confidence* = 0,9
2. *Detection Minimum Confidence* = 0,925
3. *Detection Minimum Confidence* = 0,95

Uji coba ini menghasilkan nilai-nilai evaluasi seperti pada Tabel 5.3.

Tabel 5.2 Hasil Evaluasi *Backbone Model*

Nilai DMC	Frame	Durasi	Akurasi	Kelas	Precision	Recall
0,9	100	62 s	**62,04%**	Mobil	82,82%	91,48%
				Motor	100,00%	71,88%
				Bus	100,00%	69,01%
				Truk	74,26%	77,32%
0,925	100	61 s	57,49%	Mobil	88,22%	82,00%
				Motor	100,00%	65,63%
				Bus	100,00%	67,61%
				Truk	75,27%	72,16%
0,95	100	57 s	52,61%	Mobil	91,99%	75,43%
				Motor	100,00%	59,38%
				Bus	100,00%	60,61%
				Truk	82,05%	65,98%

Sesuai dengan hasil uji coba ini, maka nilai DMC = 0,9 dipilih untuk menjadi parameter untuk pengujian berikutnya. Terlihat bahwa durasi yang dibutuhkan untuk uji coba 100 gambar dengan menggunakan *backbone model* ResNet-101 dan nilai *Detection Minimum Confidence* = 0,9 adalah 61,7 detik.

Dari data diatas, dapat diambil nilai bahwa model ini dapat memroses 100 gambar dalam waktu 61,7 detik. Dari hasil ini, untuk pengujian perbandingan metode, akan digunakan rekaman dengan tingkat *frame per second* (FPS) sebesar 1,6 FPS

5.3.3 Uji Coba Perbandingan Metode Deep Learning

Pada uji coba ini, dilakukan uji coba terhadap 2 metode Deep Learning lainnya yaitu Faster Region-Convolutional Neural Network (Faster R-CNN) dan Single Shot Detector (SSD). Data rekaman di reduksi menjadi 1.6 *frame per second* (FPS) untuk mencapai hasil *realtime* pada metode Mask-RCNN. Data rekaman yang direduksi inilah yang dijadikan pengujian. Hasil dari uji coba ini dievaluasi untuk didapatkan nilai accuracy, precision, recall. Perbandingan setiap metode dapat dilhat pada Tabel 5.4

Tabel 5.3 Hasil Evaluasi Perbandingan Metode *Deep Learning*

Metode	Frame	Akurasi	Kelas	Precision	Recall
Mask R-CNN	64	**79,79%**	Mobil	98,82%	94,92%
			Motor	90,94%	95,08%
			Bus	81,82%	52,94%
			Truk	93,75%	73,17%
Faster R-CNN	64	70,20%	Mobil	98,72%	87,01%
			Motor	92,46%	88,26%
			Bus	81,82%	52,94%
			Truk	87,50%	68,29%
SSD	64	56,86%	Mobil	94,12%	72,32%
			Motor	78,71%	74,24%
			Bus	66,67%	35,29%
			Truk	66,67%	34,15%

5.3.4 Uji Coba Perbandingan Metode Machine Learning

Pada uji coba ini, dilakukan uji coba hasil dengan metode *machine learning*. Metode yang dipilih untuk dilakukan perbandingan terhadap *Mask* R-CNN adalah *Hierarchical Multi-Support Vector Machine* (Hierarchical Multi-SVM). Metode tersebut telah diimplementasikan sebelumnya, dioptimasi dengan *Histogram of Oriented Gradients* dan *Local Binary Patterns* [26].

Pengujian dilakukan pada rekaman yang digunakan terhadap metode *Hierarchical Multi*-SVM dengan durasi 56 detik dengan tingkat *frame per second* 1.6 FPS. Pengujian ini hanya mengujikan terhadap 3 kelas yaitu motor, mobil, truk. Adapun evaluasi terhadap pengujian ini seperti pada Tabel 5.5

Tabel 5.4 Hasil Evaluasi Perbandingan Metode *Machine Learning*

Metode	Frame	Durasi	Akurasi
Mask R-CNN	90	61 s	86,78%
Hierarchical Multi-SVM	90	454,8 s	82,60%

5.4 Hasil dan Evaluasi

Uji coba pertama membandingkan penggunaan *backbone model* ResNet-101 dan ResNet-50. Dalam nilai akurasi dan setiap *precision* dan *recall* semua kelas, ResNet-101 memiliki nilai akurasi yang lebih bagus yaitu 62.04%. Oleh karena itu, *backbone model* yang optimal untuk metode ini menggunakan ResNet-101.

Gambar 5.1 Hasil Deteksi ResNet-50

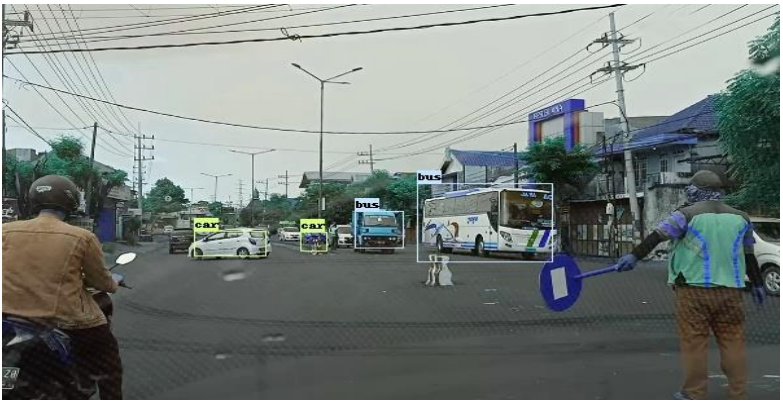
ResNet-50 memiliki nilai *recall* yang rendah terhadap kelas motor dan bus. Seperti pada Gambar 5.1, ResNet-50 tidak mampu mendeteksi kendaraan motor. Hal ini bisa terjadi dikarenakan bentuk motor yang belum lengkap pada gambar. Motor hanya terlihat tampak belakang. Beberapa misklasifikasi juga terjadi di model ResNet-50 seperti bus yang dianggap truk (Gambar 5.1). dikarenakan bentuknya yang sedikit mirip, maka model menjadi misklasifikasi.

Pada uji coba nilai *detection minimum confidence* (DMC), didapat hasil bahwa semakin tinggi nilai DMC, maka nilai *precision* setiap kelas cenderung meningkat. Hal ini dikarenakan nilai *confidence* yang kecil memiliki kemungkinan salah deteksi. Namun semakin tinggi nilai DMC, maka *recall* dari setiap kelas

cenderung menurun. Hal ini karena nilai DMC sangat memengaruhi sensitifitas pendeteksian. Pada poin akurasi, nilai DMC = 0,9 memiliki nilai yang paling bagus. Pada nilai DMC=0,9 dan DMC=0,95 ada beberapa objek yang tidak terdeteksi. Maka untuk uji parameter, didapat nilai optimal *Detection Minimum Confidence* = 0,9.

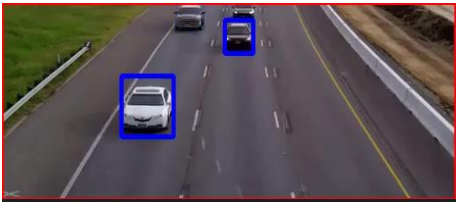
Dengan konfigurasi *backbone model* ResNet-101 dan nilai *detection minimum confidence* = 0,9, waktu yang dibutuhkan untuk menjalankan 100 gambar adalah 1 menit 2 detik. Dengan kata lain, apabila dijalankan dalam basis video akan menghasilkan video dengan tingkat *frame per second* (FPS) sekitar 1.6 FPS. Maka untuk pengujian selanjutnya yaitu membandingkan dengan metode lain dalam basis video, digunakan rekaman dengan tingkat 1.6 FPS.

Pada uji coba dengan metode lain, dilakukan perbandingan terhadap 2 metode *deep learning* yaitu *Faster Region-Convolutional Neural Network* (Faster R-CNN) dan *Single Shot Detector* (SSD). Pada pengujian dengan metode SSD, terdapat misklasifikasi truk menjadi bus pada beberapa frame (Gambar 5.2). Kesalahan ini dapat terjadi dikarenakan tampak depan truk yang berbentuk kotak.



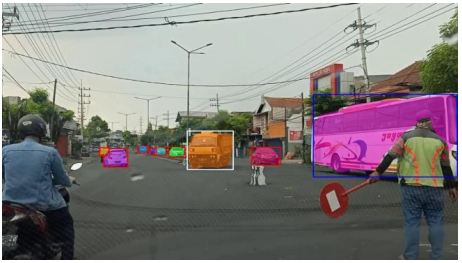
Gambar 5.2 Hasil Deteksi SSD

Selanjutnya juga dilakukan perbandingan dengan metode *machine learning* yaitu *Hierarchical Multi-Support Vector Machine* (*Hierarcichal* Multi-SVM) dengan optimasinya [26]. Beberapa *frame* yang dideteksi oleh metode *Hierarcichal* Multi-SVM tidak mampu menampilkan segmentasi yang lengkap apabila terlalu banyak objek pada *frame* (Gambar 5.3). Terlihat juga secara waktu uji coba, Mask R-CNN membutuhkan waktu sekitar 61 detik untuk 90 frame. Sedangkan Hierarcichal Multi-SVM membutuhkan waktu yang lebih lama yaitu untuk proses 1650 *frame* sebesar 8338,5 detik.



Gambar 5.3 Hasil Deteksi *Hierarchical* Multi-SVM

Setelah dilakukan beberapa pengujian, didapat hasil terbaik yaitu Mask R-CNN menggunakan *backbone* ResNet-101 dan nilai *detection minimum confidence* = 0,9 yang hasil pendeteksian nya dapat dilihat pada Gambar 5.4



Gambar 5.4 Hasil Deteksi Mask R-CNN

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas tentang kesimpulan yang didasari oleh hasil uji coba yang telah dilakukan pada bab sebelumnya. Kesimpulan nantinya sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut di masa depan.

6.1 Kesimpulan

Dalam pengerjaan Tugas Akhir ini setelah melalui tahap perancangan aplikasi, implementasi metode, serta uji coba, diperoleh kesimpulan sebagai berikut:

1. Sistem deteksi objek berbasis video dapat diimplementasikan dengan menggunakan metode Mask-RCNN.
2. Dengan menggunakan uji coba 100 *frame* pada rekaman, *backbone model* yang optimal untuk deteksi objek kendaraan adalah ResNet-101 yaitu dengan nilai akurasi 62,04%.
3. Dengan menggunakan uji coba 100 *frame* pada rekaman, nilai *detection minimum confidence* yang dapat menghasilkan hasil deteksi paling optimal adalah sebesar 0,9 dengan nilai akurasi sebesar 62,04%.
4. Untuk menghasilkan video dengan deteksi *realtime*, video harus diatur ulang dengan tingkat *frame per second* sebesar 1.6 FPS.
5. Dengan menggunakan rekaman 40 detik dan jumlah *frame* sebanyak 64 *frame*, dibandingkan dengan 2 metode *deep learning* lainnya Mask R-CNN menghasilkan nilai akurasi yang lebih baik yaitu 79,79%.
6. Dengan menggunakan rekaman 1 menit 20 detik, dibandingkan dengan salah satu metode *machine learning* yaitu *hierarchical multi-svm* metode Mask R-CNN menghasilkan hasil evaluasi yang lebih bagus yaitu dengan nilai akurasi 86,78%.

6.2 Saran

Saran yang diberikan untuk pengembangan sistem deteksi kendaraan berbasis video menggunakan metode *Mask Region-Convolutional Neural Network*, yaitu:

1. Pengembangan sistem yang dapat mengenali kendaraan lebih cepat secara waktu
2. Pengembangan sistem yang dapat mengenali lebih banyak jenis kendaraan secara mendetil.
3. Pengembangan sistem dengan arsitektur yang menghasilkan performa yang lebih baik.
4. Melakukan eksplorasi parameter selain arsitektur *backbone model* dan *detection minimum confidence* agar model dapat menghasilkan yang lebih optimal.
5. Menggunakan perangkat keras dengan spesifikasi lebih baik secara pengolahan citra agar dapat meningkatkan performa sistem secara waktu.

DAFTAR PUSTAKA

[1] M. Djalalov, H. Nisar, Y. Salih dan A. S. Malik, "An Algorithm For Vehicle Detection and Tracking," dalam *IEEE*, 2010.

[2] K. Arya, S. Tiwari dan S. Behwal, "Real-Time Vehicle Detection and Tracking," dalam *IEEE*, Chiang Mai, 2016.

[3] H. Zhu, J. Wang, K. Xie dan J. Ye, "Detection of Vehicle Flow in Video Surveillance," dalam *Institute of Electrical and Electronics Engineers*, Chongqing, 2018.

[4] Z. Yang, J. Li dan H. Li, "Real-time Pedestrian and Vehicle Detection for Autonomous Driving," dalam *IEEE*, Changshu, 2018.

[5] B. F. Momin dan T. M. Mujawar, "Vehicle detection and attribute based search of vehicles in video surveillance system," dalam *IEEE*, Nagercoil, 2015.

[6] K. He, G. Gkioxari, P. Dollar dan R. Girshick, "Mask R-CNN," Facebook Artificial Intelligence Research, 2018.

[7] Z.-Q. Zhao, P. Zheng, S.-t. Xu dan . X. Wu, "Object Detection With Deep Learning: A Review," dalam *IEEE*, 2019.

[8] M. Štancel dan M. Hulič, "An Introduction to Image Classification and Object," Technical University of Košice, 2018.

[9] R. Girshick, J. Donahue, T. Darrell dan J. Malik, "Region-based Convolutional Networks for Accurate Object Detection and Segmentation," dalam *IEEE*, 2015.

[10] "Deep learning for complete beginners: convolutional neural networks with keras," Cambridgespark, 20 March 2017. [Online]. Available: https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html. [Diakses 29 November 2018].

[11] K. He, . X. Zhang, S. Ren dan J. Sun, “Deep Residual Learning for Image Recognition,” Microsoft Research, 2015.

[12] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan dan S. Belongie, “Feature Pyramid Networks for Object Detection,” Facebook AI Research (FAIR) & Cornell University and Cornell Tech, 2017.

[13] “Review: FPN — Feature Pyramid Network (Object Detection),” Towards DataScience, 18 January 2019. [Online]. Available: https://towardsdatascience.com/review-fpn-feature-pyramid-network-object-detection-262fc7482610. [Diakses 10 January 2020].

[14] “Region of Interest Pooling Explained,” deepsense.ai, 28 February 2017. [Online]. Available: https://deepsense.ai/region-of-interest-pooling-explained/. [Diakses 10 January 2020].

[15] “How Faster R-CNN works on object detection?,” BecomingHuman.ai, 18 November 2019. [Online]. Available: https://becominghuman.ai/how-faster-r-cnn-works-on-object-detection-3d92432ce321. [Diakses 10 Januari 2020].

[16] R. Girshick, “Fast R-CNN,” Microsoft Research, 2015.

[17] S. Ren, . K. He, R. Girshick dan . J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” 2016.

[18] “About Python,” Python, [Online]. Available: https://www.python.org/about/. [Diakses 30 November 2018].

[19] “Keras: The Python Deep Learning library,” Keras, [Online]. Available: https://keras.io/. [Diakses 30 November 2018].

[20] “TensorFlow,” TensorFlow, [Online]. Available: https://www.tensorflow.org/. [Diakses 30 November 2018].

[21] “OpenCV,” [Online]. Available: https://opencv.org/. [Diakses 30 November 2018].

[22] “NumPy,” NumPy, [Online]. Available: http://www.numpy.org/. [Diakses 30 November 2018].

[23] “Scikit-learn,” Scikit-learn, [Online]. Available: http://scikit-learn.org/stable/index.html. [Diakses 30 November 2018].

[24] Zulko, “Video editing with Python”.

[25] “ResNet (34, 50, 101): Residual CNNs for Image Classification Tasks,” NeuroHive, 23 January 2019. [Online]. Available: https://neurohive.io/en/popular-networks/resnet/. [Diakses 10 Januari 2020].

[26] M. F. Maulana, “Klasifikasi Kendaraan dari Video Rekaman CCTV menggunakan Histogram of Oriented Gradients, Local Binary Patterns dan Hierarchical Multi-SVM,” Institut Teknologi Sepuluh Nopember, 2018.

(Halaman ini sengaja dikosongkan)

LAMPIRAN

L.1 Contoh Hasil Uji Coba ResNet-50

L.2 Contoh Hasil Uji Coba Resnet-101

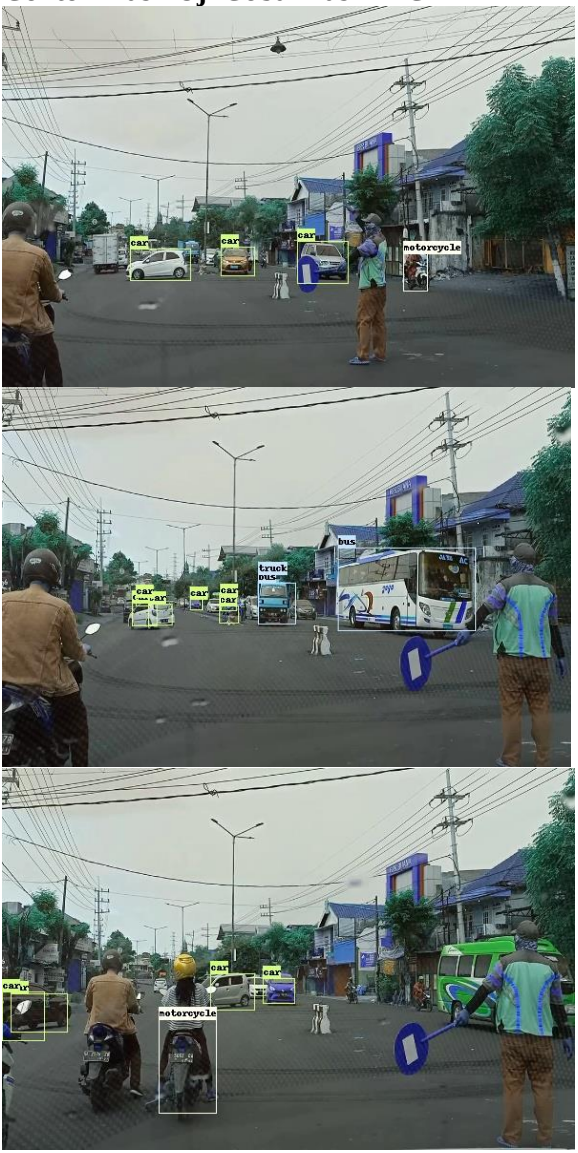
L.3 Contoh Hasil Uji Coba Faster-RCNN


car
motorcycle
motorcycle
motorcycle
motorcycle
car
truck
truck
car
car
motorcycle
motorcycle
car
car
car
car
motorcycle
motorcycle
car
car
car
motorcycle
motorcycle
car

L.4 Contoh Hasil Uji Coba Single Shot DetectoR


car
motorcycle
motorcycle
motorcycle
motorcycle
car
truck
motorcycle
motorcycle
car
truck
car
car
car
car
car
car
car
car
motorcycle
motorcycle
car

L.5 Contoh Hasil Uji Coba Mask R-CNN




truck
car
motorcycle
car
car
motorcycle
motorcycle
motorcycle
car
car
car
car
car
motorcycle
motorcycle
car

BIODATA PENULIS

Pandito Hudiarso Abdul Rahim Setia Negara, lahir di Dumai pada tanggal 27 Juni 1997. Penulis menempuh pendidikan mulai dari TK Cendana Dumai (2001-2003), SD Binaan Khusus Kota Dumai (2003-2009), SMP Negeri 2 Dumai (2009-2012), SMA Negeri Plus Provinsi Riau (2012-2015), dan sekarang sedang menjalani pendidikan S1 Informatika di ITS. Penulis aktif dalam organisasi dan kepanitiaan Himpunan Mahasiswa Teknik Computer (HMTC) dan Schematics. Diantaranya adalah menjadi staff Departemen Kesejahteraan Mahasiswa HMTC ITS 2016-2017, staff Departemen National Logic Competition (NLC) Schematics ITS 2016 dan Ketua Panitia Pelaksana Orientasi Keilmiahan dan Keprofesian Berbasis Kompetensi (OKKBK) tahun 2016. Penulis juga merupakan salah satu penerima beasiswa dari Pemerintah Provinsi Riau. Komunikasi dengan penulis dapat melalui telepon: +6282338588305 dan *email*: **harsdito27@gmail.com**.