Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – IF184802

DETEKSI POSISI DAN PENGENALAN PLAT NOMOR KENDARAAN MENGGUNAKAN SINGLE SHOT DETECTOR DAN RECURRENT NEURAL NETWORK PADA DATA VIDEO

NUZHA MUSYAFIRA
NRP 05111640000014

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

Dosen Pembimbing II
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

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

TUGAS AKHIR – IF184802

DETEKSI POSISI DAN PENGENALAN PLAT NOMOR KENDARAAN MENGGUNAKAN SINGLE SHOT DETECTOR DAN RECURRENT NEURAL NETWORK PADA DATA VIDEO

NUZHA MUSYAFIRA
NRP 05111640000014

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

Dosen Pembimbing II
Dr.Eng. Chastine Fatichah, 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 LICENSE PLATE RECOGNITION USING SINGLE SHOT DETECTOR AND RECURRENT NEURAL NETWORK

NUZHA MUSYAFIRA
NRP 05111640000014

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

Second Advisor
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

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

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

DETEKSI POSISI DAN PENGENALAN PLAT NOMOR KENDARAAN MENGGUNAKAN SINGLE SHOT DETECTOR DAN RECURRENT NEURAL NETWORK PADA DATA VIDEO

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:
NUZHA MUSYAFIRA
NRP: 05111640000014

Disetujui oleh Pembimbing Tugas Akhir:

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

2. Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. .....
   (NIP. 19751220 200112 2 002) (Pembimbing 2)

DEPARTEMEN TEKNIK INFORMATIKA

SURABAYA
Januari, 2020

(Halaman ini sengaja dikosongkan)

DETEKSI POSISI DAN PENGENALAN PLAT NOMOR KENDARAAN MENGGUNAKAN SINGLE SHOT DETECTOR DAN RECURRENT NEURAL NETWORK PADA DATA VIDEO

Nama : Nuzha Musyafira
NRP : 05111640000014
Departemen : Teknik Informatika, FTEIC-ITS
Pembimbing I : Dini Adni Navastara, S.Kom., M.Sc.
Pembimbing II : Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

ABSTRAK

Setiap kendaraan mempunyai identitasnya masing-masing, dengan kata lain, plat nomor kendaraan. Identitas ini sering kali dipergunakan dalam sistem pengolahan parkir, pengamanan, dan sebagainya. Untuk membuat sistem ini andal, diperlukan adanya pengembangan sistem otomatis yang dapat mendeteksi dan mengidentifikasi plat nomor kendaraan. Sistem ini telah banyak dikenal sebagai License Plate Recognition (LPR). LPR menggunakan konsep deteksi posisi (lokalisasi) dan segmentasi untuk identifikasi plat kendaraan pada suatu citra. Hasil plat yang telah terdeteksi nantinya akan dikenali sebagai karakter-karakter yang merepresentasikan identitas kendaraan.

Dalam tugas akhir ini, akan dilakukan pengembangan LPR yang memungkinkan membaca masukan dari data video yang mengacu pada real-time processing. Dengan menggunakan metode lokalisasi Single Shot Detector, segmentasi Binary Image diikuti dengan Connected Component Labelling, dan pengenalan karakter Recurrent Neural Network, hasil dari penelitian ini menunjukkan akurasi sebesar 91.48% untuk lokalisasi plat, 82.69% untuk segmentasi, dan 94.94% untuk pengenalan karakter.

Kata kunci: ***License Plate Recognition, Single Shot Detector, Connected Component Labelling, Recurrent Neural Network***.

VIDEO BASED LICENSE PLATE RECOGNITION USING SINGLE SHOT DETECTOR AND RECURRENT NEURAL NETWORK

Student's Name : Nuzha Musyafira
Student's ID : 05111640000014
Department : Informatics, Faculty of ELECTICS-ITS
First Advisor : Dini Adni Navastara, S.Kom., M.Sc.
Second Advisor : Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

ABSTRACT

Each vehicle has its own identity, in other words, the vehicle number plate. This identity is often used in parking processing, security, and so on. To make this system, it is necessary to develop an automated system that can be used and supported by vehicle number plates. This system is known as License Plate Recognition (LPR). LPR uses the concept of position detection (localization) and segmentation to determine vehicle plates in images. The results of the verified plate will be recognized as characters that represent the vehicle's identity.

In this undergraduate thesis, the development of LPR will be made which allows reading input in the form of video data that refers to real-time processing. By using the Single Shot Detector localization method, Binary Image segmentation followed by Connected Component Labeling, and Recurrent Neural Network character recognition, the results of this study show an accuracy of 91.48% for plate localization, 82.69% for segmentation, and 94.94% for character recognition.

Keywords: ***License Plate Recognition, Single Shot Detector, Connected Component Labelling, Recurrent Neural Network***.

KATA PENGANTAR

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

"**DETEKSI POSISI DAN PENGENALAN PLAT NOMOR KENDARAAN MENGGUNAKAN SINGLE SHOT DETECTOR DAN RECURRENT NEURAL NETWORK PADA DATA VIDEO**"

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. Orang tua dan keluarga penulis, yang telah memberikan dukungan doa, moral, dan material kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir ini.
2. Dini Adni Navastara, S.Kom., M.Sc. dan Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. selaku pembimbing I dan II yang telah membimbing dan memberikan motivasi, nasihat dan bimbingan dalam menyelesaikan Tugas Akhir ini.
3. Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. selaku Ketua Departemen Teknik Informatika ITS dan seluruh dosen dan karyawan Departemen Teknik Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa kuliah di Teknik Informatika ITS.
4. Ghifaroza Rahmadiana yang membantu dan menemani penulis selama perkuliahan semester akhir dan pengerjaan Tugas Akhir ini.
5. Azkiatunnisa Rahma, Fariz Maulana, Fariz Putra, dan Nida Regita sebagai teman-teman *apple* yang telah

memberi dukungan moral, menemani, dan menghibur penulis.

6. Admin-admin Laboratorium *Net-Centric Computing* (NCC) Adin, Akmal, Azkia, Faizal, Siraj, Ubut, Wasil, dan Zayn yang memberikan kesempatan penulis untuk fokus mengerjakan Tugas Akhir ini dan menyediakan tempat di laboratorium tersebut.
7. Admin-admin Laboratorium Komputasi Cerdas & Visi (KCV) yang memberikan kesempatan penulis untuk fokus mengerjakan Tugas Akhir ini dan menyediakan tempat di laboratorium tersebut.
8. Seluruh mahasiswa *user* TA Teknik Informatika ITS angkatan 2016 yang telah menjadi teman penulis selama pengerjaan Tugas Akhir ini.
9. Seluruh mahasiswa Teknik Informatika ITS angkatan 2016 yang telah menjadi teman penulis selama menjalani masa kuliah di Teknik Informatika ITS.
10. 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

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam beberapa tahun terakhir, seiring dengan perkembangan "*Smart City*", transportasi cerdas telah memainkan peranan yang semakin signifikan dan mendapat banyak perhatian dari komunitas akademik dan industri. Seperti yang diketahui, pengenalan plat nomor secara otomatis merupakan bagian tak terpisahkan dari sistem transportasi cerdas.

Setiap kendaraan mempunyai identitasnya masing-masing, dengan kata lain, plat nomor kendaraan. Identitas ini sering kali dipergunakan dalam sistem pemarkiran, pembangunan keamanan, dan tol. Untuk membuat sistem ini andal, diperlukan adanya pengembangan sistem otomatis yang dapat mendeteksi dan mengidentifikasi plat nomor kendaraan. Sistem ini telah banyak dikenal sebagai *License Plate Recognition* (LPR). LPR menggunakan konsep deteksi posisi (lokalisasi) dan segmentasi untuk identifikasi plat kendaraan pada suatu citra. Hasil plat yang telah terdeteksi nantinya akan dikenali sebagai karakter-karakter yang merepresentasikan identitas kendaraan.

Lokalisasi plat kendaraan merupakan proses pertama yang harus dilakukan. Proses ini memegang peranan yang krusial karena proses ini menentukan posisi dari plat. Pendekatan *Single Shot Detector* (SSD) merupakan salah satu metode pembelajaran mesin yang berfokus pada deteksi objek. SSD umumnya digunakan untuk mengenali beberapa objek pada suatu citra. Teknik ini merupakan perluasan dan pengembangan dari metode *Region-based Convolutional Neural Network* (R-CNN), yang dikenal lebih cepat dan hanya membutuhkan satu kali tangkapan [1].

Segmentasi merupakan proses memetakan karakter-karakter hasil lokalisasi menjadi gambar-gambar tersendiri. Untuk proses ini, dapat digunakan suatu pendekatan bernama *Connected Component Labelling* (CCL). CCL merupakan langkah yang umum dilakukan pada aplikasi berbasis visi. Metode ini

menetapkan label-label pada citra sedemikian hingga piksel-piksel yang berdekatan pada fitur-fitur yang sama memiliki label yang sama [2].

Proses terakhir pada keseluruhan sistem LPR adalah pengenalan karakter. Setelah melewati tahap lokalisasi dan segmentasi, keluaran dari proses sebelumnya menjadi masukan untuk proses ini. Proses ini mengklasifikasi dan menghasilkan karakter-karakter yang telah disegmentasi menjadi keluaran akhir dengan metode *Recurrent Neural Network* (RNN) [3]. Akurasi dan kevalidan data diuji di akhir proses ini.

Dalam tugas akhir ini, akan dilakukan pengembangan LPR yang memungkinkan untuk membaca masukan berupa dari data video yang mengacu pada *real-time processing*. Dengan menggunakan metode lokalisasi SSD, segmentasi CCL, dan pengenalan karakter RNN, deteksi dan pengenalan berdasarkan data video diharapkan lebih cepat. Hal ini tentunya diharapkan dapat meningkatkan efisiensi waktu dan kemudahan dalam identifikasi plat nomor kendaraan. Selain itu, penelitian ini juga dapat menjadi referensi untuk pengembangan maupun optimasi bagi studi yang berkaitan.

1.2 Rumusan Masalah

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

1. Bagaimana melakukan lokalisasi plat nomor kendaraan pada data video dengan metode *Single Shot Detector*?
2. Bagaimana proses segmentasi dari hasil deteksi yang didapat dengan metode *Connected Component Labelling*?
3. Bagaimana proses pengenalan karakter dari hasil segmentasi dengan metode *Recurrent Neural Network*?
4. Bagaimana performa sistem yang telah dibangun pada saat proses pengujian?

1.3 Batasan Permasalahan

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

1. Dataset yang dipakai bersumber dari rekam digital kamera area parkir kendaraan motor Departemen Teknik Informatika, ITS dengan hasil citra data uji berisi plat nomor kendaraan motor yang posisinya tepat lurus menghadap kamera.
2. Spesifikasi plat nomor kendaraan yang dapat diatasi adalah standar plat nomor kendaraan motor di Indonesia dengan *background* plat kendaraan berwarna hitam dan tulisan berwarna putih.
3. Menggunakan bahasa Python 3.

1.4 Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah untuk membangun sebuah sistem pengenalan plat nomor kendaraan dengan menggunakan metode klasifikasi *Single Shot Detector* dan *Recurrent Neural Network* yang dapat mengenali plat nomor kendaraan pada data video kendaraan.

1.5 Manfaat

Tugas akhir ini diharapkan dapat membantu menambah kemampuan yang ada pada pengenalan plat nomor kendaraan dan membantu dalam melakukan pengenalan karakter sehingga dapat diimplementasikan pada sistem-sistem yang membutuhkan pengenalan plat nomor kendaraan secara otomatis.

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 *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 data video yang diambil dari kamera CCTV serta untuk mengetahui hasil dan performa arsitektur yang telah dibangun. Evaluasi dilakukan dengan mengevaluasi model *Single Shot Detector* dan *Recurrent Neural Network* dari segi akurasi, *precision, recall,* serta dengan menggunakan data video sebagai data *testing* untuk melihat performa model yang telah dibuat.

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 *Neural Network* dan *library* yang digunakan.

Bab III Perancangan Sistem

Bab ini berisi pembahasan mengenai perancangan dari metode *Single Shot Detector* dan *Recurrent Neural Network* yang digunakan untuk pengenalan plat nomor kendaraan pada data video.

Bab IV Implementasi

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

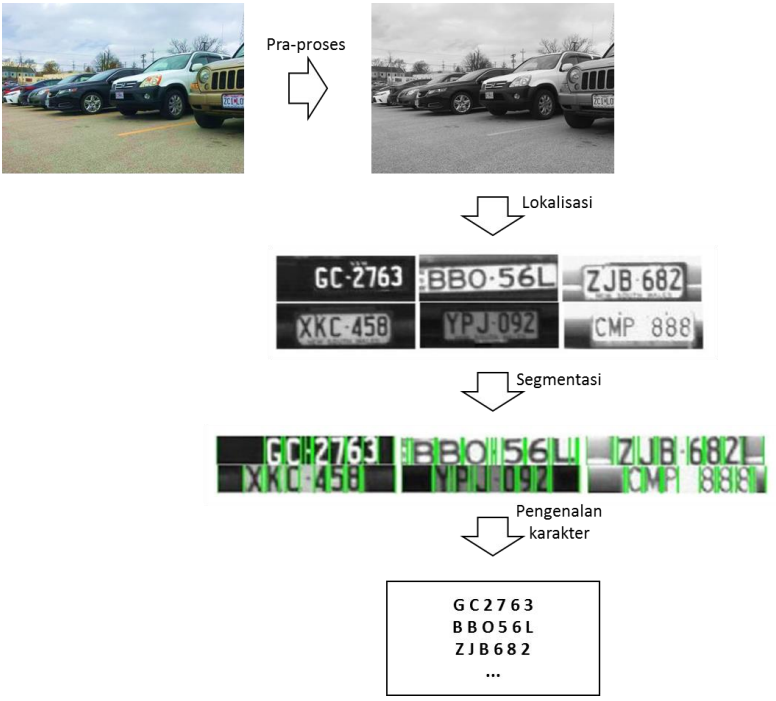
Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses dan tertulis saat pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.

(Halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

Bab ini membahas mengenai teori-teori dasar yang digunakan dalam Tugas Akhir. Teori-teori tersebut adalah *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 License Plate Recognition (LPR)



Gambar 2.1 Ilustrasi Sistem *LPR*

License Plate Recognition (LPR) seperti namanya merupakan sistem untuk mendeteksi dan mengenali plat nomor

pada kendaraan. *LPR* menggunakan konsep pengenalan karakter optik untuk membaca karakter-karakter pada kendaraan. Dengan kata lain, *LPR* mengambil citra dari kendaraan sebagai masukan dan karakter-karakter yang tertulis pada plat kendaraan sebagai keluarannya.

LPR secara garis besar dibagi menjadi tiga tahapan [4]:

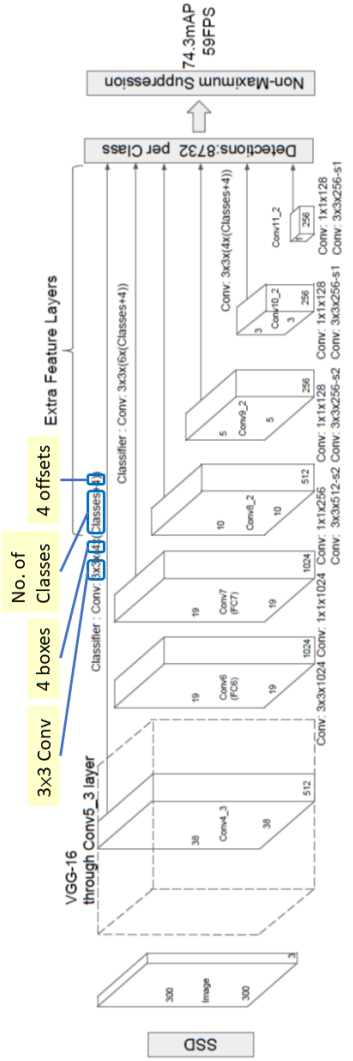
1. Lokalisasi Plat Kendaraan
   Tahap ini merupakan tahap pertama dan dapat dibilang tahap yang paling penting dari sistem. Pada tahap ini posisi plat nomor ditentukan. Masukan pada tahap ini adalah citra kendaraan dan keluarannya adalah plat.
2. Segmentasi
   Pada tahap ini, karakter-karakter pada plat nomor dipetakan dan disegmentasi menjadi gambar-gambar tersendiri.
3. Pengenalan Karakter
   Tahap ini merupakan tahap yang menyelesaikan semuanya. Karakter yang sebelumnya tersegmentasi diidentifikasi di sini.

2.2 Single Shot Detector (SSD)

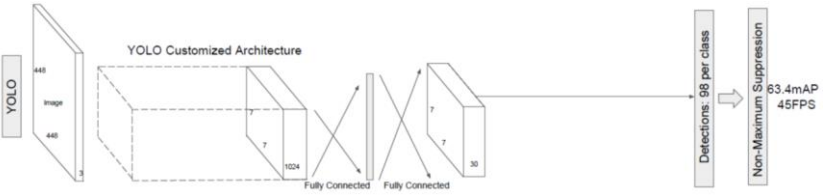
Single Shot Detector (SSD) merupakan salah satu metode pembelajaran mesin yang berfokus pada deteksi objek. SSD umumnya digunakan untuk mengenali beberapa objek pada suatu citra. Teknik ini merupakan perluasan dan pengembangan dari metode *Region-based Convolutional Neural Network* (R-CNN), namun metode ini dikenal lebih cepat dan hanya membutuhkan satu kali tangkapan [5] daripada R-CNN dan *You Only Look Once* (YOLO).

Pada R-CNN, secara garis besar dibutuhkan dua kali tangkapan dalam mendeteksi objek. Tangkapan pertama digunakan untuk menghasilkan kandidat-kandidat yang diasumsikan sebagai objek yang dicari (*region proposals*), kemudian setiap kandidat

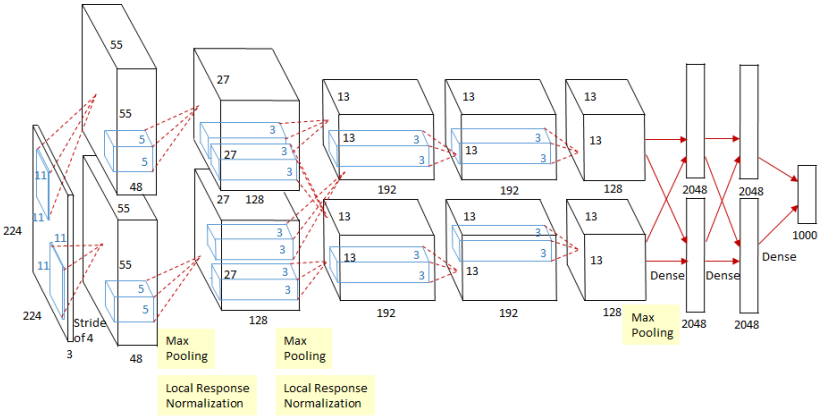
yang dihasilkan nantinya akan diberi label sesuai kelas yang tersedia pada tangkapan yang kedua.



Gambar 2.2 Contoh arsitektur jaringan SSD



Gambar 2.3 Contoh arsitektur jaringan YOLO



Gambar 2.4 Contoh arsitektur jaringan R-CNN

Single Shot Detector menggabungkan kedua tangkapan pada *R-CNN* dalam satu tangkapan. Untuk mencapai hal ini, metode ini menggunakan *ground truth* (*GT*) *boxes* dan *default boxes*. Untuk setiap *default box*, *shape offsets* dan *confidences* untuk seluruh kategori objek diprediksi. Kemudian saat pelatihan data, *default boxes* akan dicocokkan dengan *GT boxes*.

2.3 Otsu Thresholding

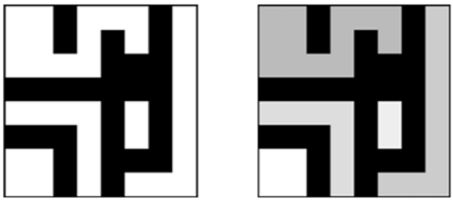
*Otsu Thresholding m*erupakan sebuah metode *thresholding* citra di mana diasumsikan pada sebuah citra bimodal yang mengandung dua kelas piksel yang tinggi, akan dikalkulasi nilai *threshold* optimum yang memisahkan dua kelas agar *intra-class variance* minimal atau setara, agar *inter-class variance* maksimal [6].

2.4 Image Binarization

Image Binarization adalah proses mengubah *grey values* dari citra menjadi nilai-nilai biner dan mereprestansikan ulang citra sebagai citra biner yang sesuai. Proses ini menyoroti *pixels of interest* (*POI*) dan menyupresi *background pixels*.

Langkah sederhana untuk proses *Image Binarization* yaitu memilih nilai *threshold*, kemudian mengklasifikasi semua piksel dengan nilai di atas *threshold* sebagai warna putih (*grey value* 255) dan semua piksel lain sebagai warna hitam (*grey value* 0) [2].

2.5 Connected Component Labelling (CCL)



(a) (b)

Gambar 2.5 Contoh CCL. (a) Gambar orisinil; (b) Setelah CCL.

1	1	0	1	1	1	0	1
1	1	0	1	0	1	0	1
1	1	1	1	0	0	0	1
0	0	0	0	0	0	0	1
1	1	1	1	0	1	0	1
0	0	0	1	0	1	0	1
1	1	0	1	0	0	0	1
1	1	0	1	0	1	1	1

(a)

1	1	0	1	1	1	0	2
1	1	0	1	0	1	0	2
1	1	1	1	0	0	0	2
0	0	0	0	0	0	0	2
3	3	3	3	0	4	0	2
0	0	0	3	0	4	0	2
5	5	0	3	0	0	0	2
5	5	0	3	0	2	2	2

(b)

Gambar 2.6 CCL dalam piksel. (a) Gambar orisinil; (b) Setelah CCL.

Connected Component Labelling (CCL) adalah aplikasi algoritmik dari teori grafik, di mana himpunan bagian dari

komponen yang terhubung dilabeli secara unik berdasarkan heuristik yang diberikan. CCL digunakan dalam visi komputer untuk mendeteksi daerah yang terhubung dalam gambar digital biner misalnya segmentasi. Algoritma CCL yang digunakan pada tugas akhir kali ini dapat dilihat pada Tabel 2.1 [7].

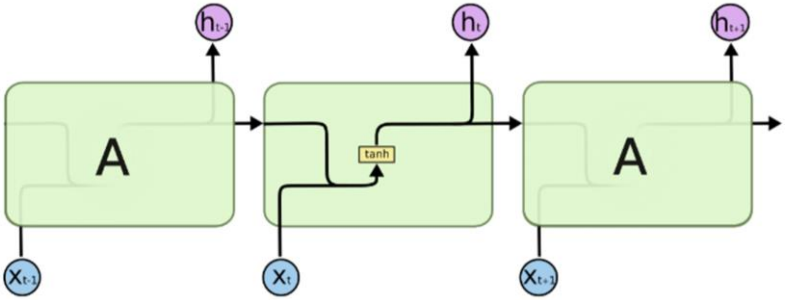
Tabel 2.1 Algoritma CCL

***Algorithm** Connected Component Labelling*	
***Input**: Labelling a particular pixel (x, y)*	
1	***if** the pixel (x, y) has '0' **then***
2	*Do nothing and proceed to next pixel (x + 1, y)*
3	***else if** the pixel (x – 1, y – 1) has a label **then***
4	*Assign the label to the pixel (x, y)*
5	***else if** neither pixels (x – 1, y) or (x, y – 1) is not labelled **then***
6	*Increment label numbering and assign the latest label to pixel (x, y)*
7	***else if** pixels (x – 1, y) XOR (x, y – 1) is labelled **then***
8	*Assign the label to the pixel (x, y)*
9	***else if** both pixels (x – 1, y) and (x, y – 1) are labelled **then***
10	*Assign the label of pixel (x – 1, y) to the pixel (x, y)*
11	*Record the equivalence if labels of pixels (x – 1, y) and (x, y – 1) are not identical*
12	***end if***

2.6 Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN) adalah salah satu metode pembelajaran mesin yang berfokus pada prediksi dari hasil pola. Secara umum, RNN mengingat masa lalu dan membuat keputusan yang dipengaruhi oleh apa yang telah dipelajari dari masa lalu [8]. Selagi RNN belajar dengan cara yang sama saat pelatihan, di samping itu, RNN mengingat hal-hal yang dipelajari dari masukan sebelumnya saat menghasilkan keluaran. RNN dapat mengambil satu atau lebih vektor masukan dan menghasilkan satu atau lebih vektor keluaran. Keluaran tidak hanya dipengaruhi oleh bobot yang diterapkan pada masukan seperti *Neural Network* biasa, tetapi juga oleh vektor status "tersembunyi" yang mewakili konteks

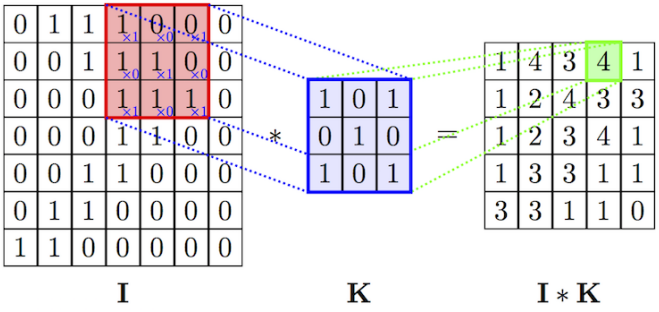
berdasarkan masukan atau keluaran sebelumnya. Jadi, masukan yang sama dapat menghasilkan keluaran yang berbeda tergantung pada masukan sebelumnya dalam suatu seri.



Gambar 2.7 Contoh arsitektur jaringan RNN

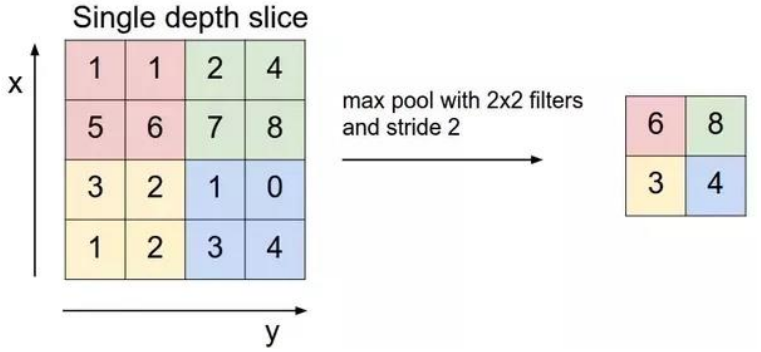
2.6.1 Convolution Layer

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 [9]. Ilustrasi cara kerja konvolusi bisa dilihat pada Gambar 2.8, dimana I adalah citra, K adalah *filter* atau kernel yang digunakan, I * K adalah hasil operasi konvolusi.



Gambar 2.8 Ilustrasi cara kerja konvolusi [10]

2.6.2 Pooling Layer



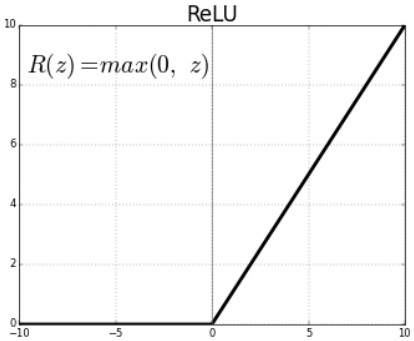
Gambar 2.9 Ilustrasi cara kerja *Max Pooling* [11]

Fungsi dari *Pooling Layer* adalah mereduksi ukuran dari data. Terdapat beberapa tipe *Pooling Layer* diantaranya yaitu *max*, *average*, *sum* dan lainnya. Metode *Pooling* dalam *Convolutional Neural Network* (CNN) yang biasa digunakan adalah *Max Pooling* & *Average Pooling*. *Max Pooling* membagi *output* dari *Convolution Layer* menjadi beberapa matriks kecil lalu mengambil nilai maksimal dari tiap matriks untuk menyusun matriks citra yang telah direduksi, sedangkan *Average Pooling* akan memilih nilai rata-ratanya. Proses tersebut memas-tikan fitur yang didapatkan akan sama meskipun obyek citra mengalami translasi. Ilustrasi cara kerja *Max Pooling* bisa dilihat pada Gambar 2.9.

2.6.3 Fully Connected Layer

Fully Connected Layer dalam penerapannya sama dengan *Multi Layer Perceptron* (MLP) yang bertujuan untuk melakukan transformasi pada dimensi data agar data dapat diklasifikasikan secara linear. *Feature map* dari *Convolution Layer* perlu ditransformasi menjadi data satu dimensi terlebih dahulu yang disebut *feature vector* sebelum dapat dimasukkan ke dalam sebuah *Fully Connected Layer*. Karena hal tersebut menyebabkan data kehilangan informasi spasialnya dan tidak reversibel, *Fully Connected Layer* diimplementasikan di akhir jaringan [12].

2.6.4 ReLU Activation Function



Gambar 2.10 ReLU *Activation Function* [13]

Fungsi aktivasi befungsi untuk menentukan apakah *neuron* tersebut harus aktif atau tidak berdasarkan nilai masukan. Salah satu contoh fungsi aktivasi adalah ReLU (*Rectified Linear Unit*) dimana fungsi ini melakukan *thresholding* dengan nilai nol terhadap nilai masukan, dimana seluruh nilai yang kurang dari nol akan dijadikan nol, seperti pada Gambar 2.10.

2.6.5 Fungsi Softmax

Fungsi *softmax* biasa digunakan dalam klasifikasi banyak kelas. *Softmax* memberikan nilai probabilitas untuk setiap label kelas, dimana jumlah seluruh probabilitas adalah 1. *Softmax* pada dasarnya adalah probabilitas eksponensial yang dinormalisasi dari nilai masukan sejumlah kelas pada model klasifikasi seperti pada Persamaan (2.1).

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \tag{2.1}$$

Dimana *y* adalah nilai masukan. Operasi akan menghasilkan nilai probabilitas. Label dari data masukan akan ditentukan berdasarkan kelas dengan nilai probabilitas tertinggi.

2.6.6 Cross Entropy

Loss function merupakan fungsi yang menggambarkan kerugian yang dihasilkan oleh model. *Loss function* dikatakan baik, ketika menghasilkan *error* yang diharapkan paling rendah. Pada permasalahan klasifikasi banyak kelas, *cross entropy* adalah *loss function* yang biasa digunakan. *Cross entropy* akan menghitung *error* antara nilai prediksi *S* dengan nilai sebenarnya *T*, seperti pada Persamaan (2.2). Selanjutnya, nilai *error* akhir diambil dari rata-rata hasil *cross entropy*, seperti pada Persamaan (2.3).

$$D(S_i, T_i) = -\sum_j T_{ij} \log S_{ij} \tag{2.2}$$

$$J(W, b) = \frac{1}{n} \sum_i D(S_i, T_i) \tag{2.3}$$

2.6.7 Stochastic Gradient Descent

Ketika melatih sebuah model, dibutuhkan sebuah *loss function* yang dapat mengukur kualitas dari setiap bobot atau parameter tertentu. *Stochastic Gradient Descent* (SGD) adalah algoritma pengoptimalan. Tujuan pengoptimalan adalah untuk menemukan parameter yang dapat meminimalkan nilai *error* dari *loss function.* SGD adalah algoritma yang digunakan untuk memperbarui nilai bobot dan bias pada neuron di *neural network*. Pada dasarnya operasi yang dilakukan hanya mengurangi bobot awal dengan sebagian nilai dari nilai gradien yang sudah kita dapat. Nilai sebagian disini diwakili oleh parameter bernama *learning rate*, seperti yang terlihat pada Persamaan (2.4) dan (2.5).

$$w_{j+1} = w_j - \alpha \frac{\partial}{\partial w_j} J(W, b) \tag{2.4}$$

$$b_{j+1} = w_j - \alpha \frac{\partial}{\partial b_j} J(W, b) \tag{2.5}$$

2.6.8 Adagrad

Adagrad adalah algoritma pengoptimalan berbasis gradien yang memperbarui *learning rate* setiap parameternya. Adagrad

adalah algoritma yang digunakan untuk memperbarui nilai bobot dan bias pada neuron di *neural network*. Adagrad menggunakan *adaptive learning rate* untuk setiap parameter. Berbeda dengan *Stochastic Gradient Descent* (SGD) yang selalu menggunakan *learning rate* yang sama, Adagrad memiliki sebuah *learning rate* untuk setiap parameter dan secara terpisah beradaptasi saat proses pelatihan. Pembaruan dilakukan untuk tiap parameter **θ(j)** dengan gradien *loss function* **g(j)**, seperti yang dapat dilihat pada Persamaan (2.6) dan (2.7).

$$g_j = \frac{\partial}{\partial \theta_j} J(\theta_j) \tag{2.6}$$

$$\theta_{j+1,} = \theta_j - \frac{\alpha}{\sqrt{\sum_{i=0}^{j} (g_i)^2}} g_j \tag{2.7}$$

2.6.9 RMSProp

RMSProp (*Root Mean Square Propagation*) adalah metode pengoptimalan berbasis *adaptive learning rate* yang diusulkan oleh Geoffrey Hinton [14]. RMSProp memodifikasi Adagrad dengan mengganti akumulasi gradien menjadi rata-rata bergerak gradien yang diberi bobot secara kuadratik, seperti yang dapat dilihat pada Persamaan (2.8) dan (2.9).

$$s_j = \beta . s_{j-1} + (1 - \beta)(g_j)^2 \tag{2.8}$$

$$\theta_{j+1} = \theta_j - \frac{\alpha}{\sqrt{s_j + \varepsilon}} g_j \tag{2.9}$$

2.6.10 Adam

Adam (*Adaptive Moment Estimation)* juga adalah algoritma pengoptimalan yang dapat digunakan. Hampir sama dengan Adagrad, Adam memiliki sebuah *learning rate* untuk setiap parameter dan secara terpisah beradaptasi saat proses pelatihan. Adam memperbarui nilai setiap parameter seperti RMSProp [15]. Perbedaannya Adam menggunakan gradien yang telah diperhalus dan semakin mengecil seperti yang dapat dilihat pada Persamaan (2.10). Lalu gradien tersebut akan digunakan untuk memperbarui

parameter, seperti yang dapat dilihat pada Persamaan (2.11) dan (2.12).

$$m_j = \beta_1 . m_{j-1} + (1 - \beta_1) . g_j \quad (2.10)$$

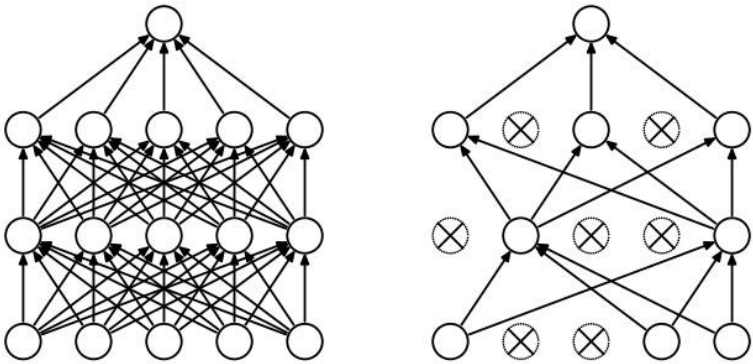
$$s_j = \beta_2 . s_{j-1} + (1 - \beta_2)(g_j)^2 \quad (2.11)$$

$$\theta_{j+1} = \theta_j - \frac{\alpha}{\sqrt{s_j + \varepsilon}} m_j \quad (2.12)$$

2.6.11 Dropout

Dropout merupakan proses mencegah terjadinya *overfitting* dan juga mempercepat proses learning. *Dropout* mengacu kepada menghilangkan neuron yang berupa *hidden layer* mapun *visible layer* di dalam jaringan [16]. Dengan menghilangkan suatu neuron, berarti menghilangkannya sementara dari jaringan yang ada. Neuron yang akan dihilangkan akan dipilih secara acak.

Pada Gambar 2.11, (a) neuron tetap utuh pada *neural network* yang belum memakai *Dropout*, dan (b) *neural network* yang sebagian dari neuronnya tidak digunakan setelah diaplikasikan Dropout.



Gambar 2.11 Ilustrasi *neural network* mengaplikasikan *Dropout* [17]

2.6.12 Batch Normalization

Batch Normalization adalah teknik melakukan normalisasi terhadap *batch* atau kumpulan data masukan, seperti yang terlihat pada Persamaan (2.13). Dimana *x* adalah nilai masukan, μ_b adalah *mean* dari *batch*, σ_b adalah standar deviasi dari *batch*. Normalisasi dilakukan agar data memiliki *mean* mendekati 0 dan standar deviasi mendekati 1.

$$x_{baru} = \frac{x-\mu_b}{\sigma_b} \tag{2.13}$$

2.7 Confusion Matrix

Confusion Matrix adalah pengukuran kinerja untuk masalah klasifikasi pembelajaran mesin di mana keluaran bisa sebanyak dua atau lebih kelas [18]. Berikut adalah tabel dengan 4 kombinasi nilai prediksi dan aktual yang berbeda.

Tabel 2.2 *Confusion matrix*

		Nilai Aktual	
		Positif	Negatif
Nilai Prediksi	Positif	TP	FP
	Negatif	FN	TN

TP (*True Positive*) adalah ketika yang diprediksi positif dan kenyataannya benar. Contohnya, wanita yang diprediksi hamil dan sesungguhnya benar hamil. *TN* (*True Negative*) adalah ketika yang diprediksi negatif dan kenyataannya benar. Contohnya, lelaki yang diprediksi tidak hamil dan sesungguhnya benar tidak hamil. *FP*

(*False Positive*) adalah ketika yang diprediksi positif namun kenyataannya salah. Contohnya, lelaki yang diprediksi hamil namun sesungguhnya tidak hamil. *FN* (*False Negative*) adalah ketika yang diprediksi negatif namun kenyataannya salah. Contohnya, wanita yang diprediksi tidak hamil namun sesungguhnya hamil.

Nilai-nilai di atas dapat digunakan untuk mengukur tingkat validasi data. Beberapa jenis teknik validasi yang umum digunakan antara lain:

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$F - Measure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3}$$

2.8 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 [19].

2.9 Library

Library merupakan sekumpulan program yang dapat digunakan pada program lain tanpa terikat satu dengan yang lainnya. Terdapat beberapa *library* yang digunakan dalam melakukan implementasi tugas akhir ini. *Library* yang digunakan antara lain, Keras, TensorFlow, OpenCV, Numpy, Scikit-learn, Matplotlib, Scikit-image, dan SciPy.

2.9.1 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 [20].

2.9.2 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 [21].

2.9.3 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 [22].

2.9.4 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 [23].

2.9.5 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 [24].

2.9.6 Matplotlib

Matplotlib adalah *library Python* yang mendukung pembuatan grafik dua dimensi dalam berbagai format dan dari berbagai jenis data. Matplotlib bersifat *open source* dan banyak digunakan untuk pengolahan data dalam penelitian. Matplotlib dapat membuat plot, histogram, spektrum daya, diagram batang, diagram kesalahan, plot pencar, dan lain-lain [25].

2.9.7 Scikit-image

Scikit-image (skimage) adalah *library* Python yang berisi kumpulan algoritma untuk pemrosesan gambar dan visi komputer. *Package* utama skimage menyediakan beberapa utilitas untuk mengkonversi antar tipe data gambar [26].

2.9.8 SciPy

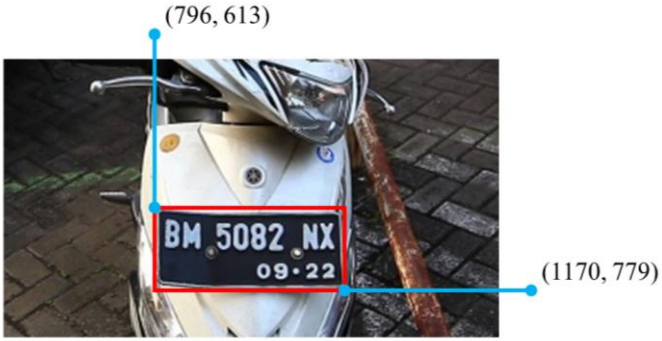
SciPy adalah *library* Python gratis dan *open-source* yang digunakan untuk komputasi ilmiah dan komputasi teknis. SciPy berisi modul untuk optimasi, aljabar linier, integrasi, interpolasi, fungsi khusus, FFT, pemrosesan sinyal dan gambar, pemecah ODE, dan tugas-tugas lain yang umum dalam sains dan teknik [27].

BAB III
PERANCANGAN SISTEM

Bab ini menjelaskan tentang perancangan data dan sistem pengenalan plat kendaraan menggunakan *Single Shot Detector* (SSD) dan *Recurrent Neural Network* (RNN). Bab ini juga akan menjelaskan gambaran umum sistem dalam bentuk diagram alir.

3.1 Perancangan Data

Data yang digunakan sebagai masukan awal dari sistem deteksi posisi dan pengenalan plat nomor kendaraan dibagi menjadi dua, yaitu data untuk deteksi posisi plat nomor kendaraan menggunakan SSD dan data untuk pengenalan karakter pada plat nomor kendaraan menggunakan RNN. Data untuk deteksi posisi plat nomor kendaraan adalah data yang diambil secara mandiri untuk data latih dan data yang diambil dari kamera CCTV untuk data uji. Data yang diambil untuk data latih terdiri dari 1 kelas di mana di masing-masing gambar terdapat minimal 1 plat beserta anotasi posisi plat. Anotasi-anotasi ini merupakan hasil pencocokan yang dilakukan secara mandiri. Contoh gambar untuk data latih bisa dilihat pada Gambar 3.1.



Gambar 3.1 Contoh citra untuk data latih

Sedangkan untuk data uji pada deteksi posisi plat nomor kendaraan, data berasal dari kamera CCTV. Pada setiap citra dari

data uji, gambar dapat meliputi 1 kelas plat nomor kendaraan atau tidak ada plat sama sekali, namun gambar pada data uji tidak dilakukan anotasi seperti pada latih. Untuk proses deteksi posisi plat nomor kendaraan, spesifikasi lengkap data latih awal dapat dilihat pada Tabel 3.1.

Tabel 3.1 Spesifikasi awal dataset lokalisasi

Keterangan	**Spesifikasi**
Ukuran resolusi asli	1280 x 720
Ekstensi	.jpg
Jumlah gambar	280
Jumlah kelas	1 kelas
Ukuran file	200 - 300 kB
Kanal warna	3 (RGB)

Gambar 3.2 Contoh karakter pada dataset

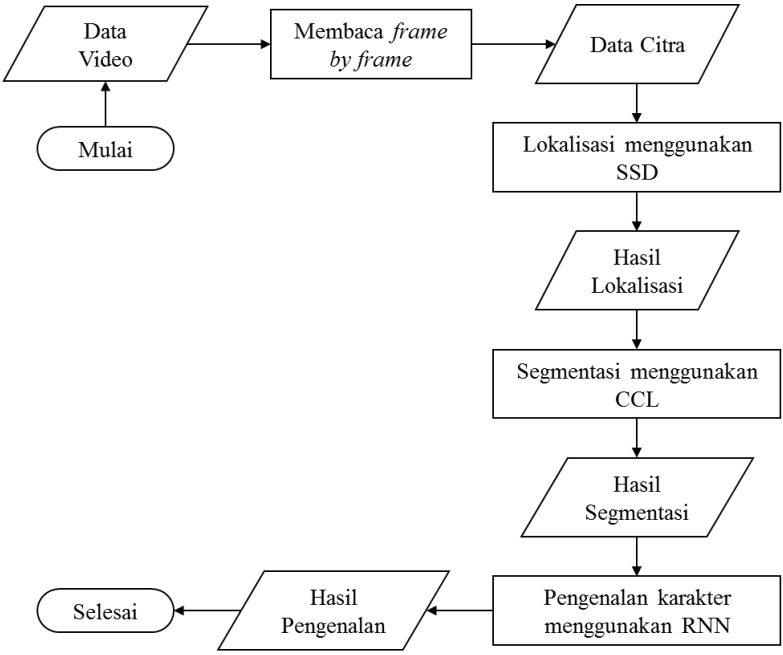
Untuk pengenalan karakter, dataset yang digunakan adalah dataset yang terdiri dari 36 kelas di mana pada masing-masing kelas terdapat sekitar 6000 sampai dengan 8000 citra [28]. Data ini merupakan sekumpulan citra karakter A sampai dengan Z serta angka 0 sampai dengan 9. Contoh karakter bisa dilihat pada Gambar 3.2.

Tabel 3.2 Spesifikasi dataset karakter

Keterangan	**Spesifikasi**
Ukuran resolusi	32 x 32
Ekstensi	.jpg
Jumlah gambar	279,247
Jumlah kelas	36 kelas
Jumlah gambar per kelas	6000 - 8000
Ukuran file	1 kB
Kanal warna	1 (*binary*)

Pembagian data latih dan data validasi menjadi 91,333 data validasi dan 187,914 data latih. Spesifikasi lengkap dataset pengenalan karakter dapat dilihat pada Tabel 3.2.

3.2 Desain Umum Sistem



Gambar 3.3 Diagram alir sistem yang dibangun

Sistem pengenalan nomor polisi kendaraan yang dibangun memiliki proses utama di antaranya lokalisasi, segmentasi, dan pengenalan karakter. Diagram alir dari sistem ditunjukkan pada Gambar 3.3.

Proses lokalisasi merupakan tahap pertama dan dapat dibilang tahap yang paling penting dari sistem. Pada tahap ini posisi plat nomor ditentukan. Masukan pada tahap ini adalah citra kendaraan dan keluarannya adalah plat nomor kendaraan hasil lokalisasi. Tahap ini secara garis besar dibagi menjadi 3 bagian, yaitu praproses dan pembangunan arsitektur SSD, pelatihan, dan pengujian. Model yang telah terbentuk kemudian akan digunakan untuk proses pengujian atau lokalisasi.

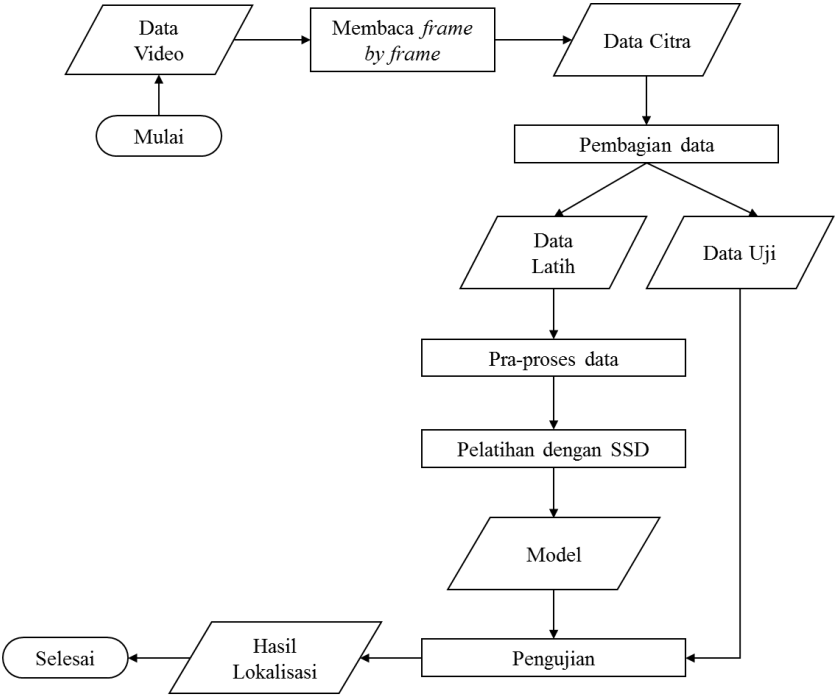
Pada proses segmentasi, karakter-karakter pada plat nomor kendaraan hasil lokalisasi akan dipetakan dan disegmentasi menjadi gambar-gambar tersendiri. Masukan pada tahap ini adalah citra plat nomor kendaraan dan keluarannya adalah kumpulan citra karakter yang berhasil tersegmentasi. Tahap ini secara garis besar dibagi menjadi 2 bagian, yaitu praproses dan segmentasi menggunakan *Connected Component Labelling* (CCL).

Proses pengenalan karakter merupakan tahap yang menyelesaikan semuanya. Karakter yang sebelumnya tersegmentasi diidentifikasi di sini. Masukan pada tahap ini adalah citra karakter hasil segmentasi dan keluarannya adalah kelas dari karakter yang diidentifikasi. Tahap ini secara garis besar dibagi menjadi 3 bagian, yaitu pembangunan arsitektur RNN, pelatihan, dan pengujian. Model yang telah terbentuk kemudian akan digunakan untuk proses pengujian.

3.2.1 Tahap Lokalisasi

Pada tugas akhir ini, akan dilakukan tahap lokalisasi yang berfungsi untuk mendeteksi dan menentukan posisi plat nomor kendaraan pada data masukan. Tahap ini secara garis besar dibagi menjadi 3 bagian, yaitu pembangunan arsitektur SSD, pelatihan,

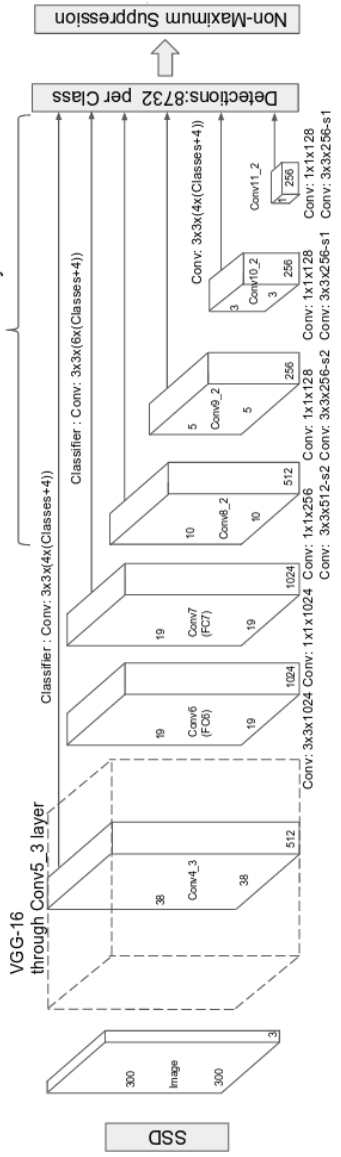
dan pengujian. Diagram alir tahap lokalisasi dapat dilihat pada Gambar 3.4.



Gambar 3.4 Diagram alir lokalisasi plat nomor kendaraan

3.2.1.1 Pembangunan Arsitektur

Pada tahap lokalisasi, akan dilakukan praproses data terlebih dahulu untuk masukan pada proses pelatihan. Praproses data yang dilakukan antara lain *brightness transform*, *contrast transform*, *hue transform*, *saturation transform*, *horizontal_flip*, dan *resize*. Praproses dilakukan agar data bervariasi dan menyamakan ukuran data latih.



Gambar 3.5 Arsitektur SSD yang digunakan [29]

Pembangunan model bertujuan untuk menyiapkan *layer*, fungsi aktivasi, *loss function*, dan parameter apa saja yang dibutuhkan. Arsitektur SSD dapat dilihat pada Gambar 3.5. SSD yang digunakan terdiri dari *convolutional layer*, *max pooling layer* dan *fully connected layer* yang diambil dari *pre-trained* VGG16 model pada data ImageNet.

3.2.1.2 Pelatihan

Untuk pelatihan, data merupakan data yang bersumber dari video. Data video diambil secara mandiri kemudian yang diubah menjadi *frames* atau kumpulan citra sebanyak 280.

Proses pelatihan memanfaatkan data latih untuk membangun SSD. Pelatihan menggunakan *preset* VGG300 dengan *learning rate* yang sudah ditentukan sebelumnya. Proses pelatihan akan dijalankan pada *batch size* 8 dan jumlah *epoch* menyesuaikan skenario uji coba. Dalam setiap akhir *epoch* terdapat proses pengujian model terhadap data validasi untuk mengetahui seberapa baik model dilatih.

3.2.1.3 Pengujian

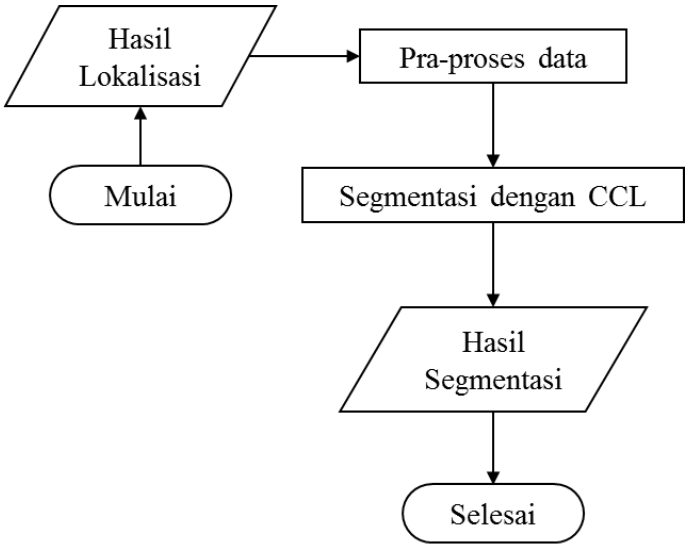
Untuk tahap pengujian, data yang digunakan adalah data video yang berasal dari kamera CCTV. Setiap data dalam bentuk video ini akan dilakukan proses konversi menjadi *frames* terlebih dahulu.

Setelah melalui tahapan konversi, data siap diuji menggunakan model yang telah terbentuk pada tahap pelatihan. Tahap pengujian ini digunakan untuk mengetahui seberapa baik model dilatih dan diaplikasikan pada lingkungan yang belum pernah dikenalinya. Pada akhir pengujian, akan dilakukan perhitungan untuk mendapatkan nilai *akurasi*, *precision* dan *recall*.

3.2.2 Tahap Segmentasi

Pada tugas akhir ini, akan dilakukan tahap segmentasi yang berfungsi untuk memecah citra pada hasil lokalisasi menjadi segmen-segmen karakter. Tahap ini secara garis besar dibagi

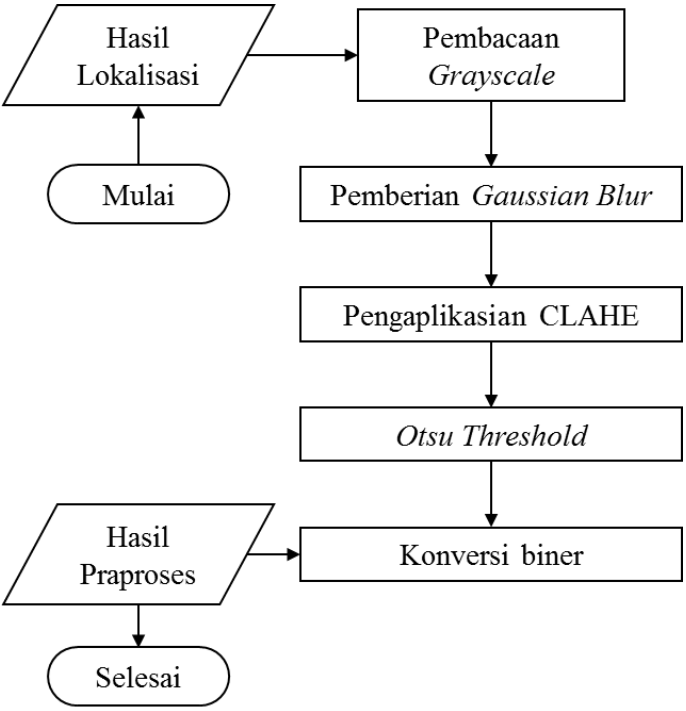
menjadi 2 bagian, yaitu praproses dan segmentasi dengan *Connected Component Labelling* (CCL). Diagram alir tahap segmentasi dapat dilihat pada Gambar 3.6.



Gambar 3.6 Diagram alir tahap segmentasi

3.2.2.1 Praproses

Pada tahap praproses, citra hasil lokalisasi dikonversi terlebih dahulu menjadi citra *grayscale*. Kemudian, gambar diproses kembali menggunakan *Gaussian Blur* untuk mengurangi *noise* pada gambar. Setelah itu, pengaplikasian CLAHE bertujuan untuk menyamaratakan kontras pada gambar sebelum dikonversi menjadi gambar biner. Untuk menentukan *threshold* yang tepat untuk konversi biner, *threshold value* dihitung menggunakan bantuan *Otsu Threshold*. Gambar akhirnya akan diubah ke dalam citra biner untuk digunakan pada tahap selanjutnya. Diagram alir tahap praproses ini dapat dilihat pada Gambar 3.7.



Gambar 3.7 Diagram alir praproses tahap segmentasi

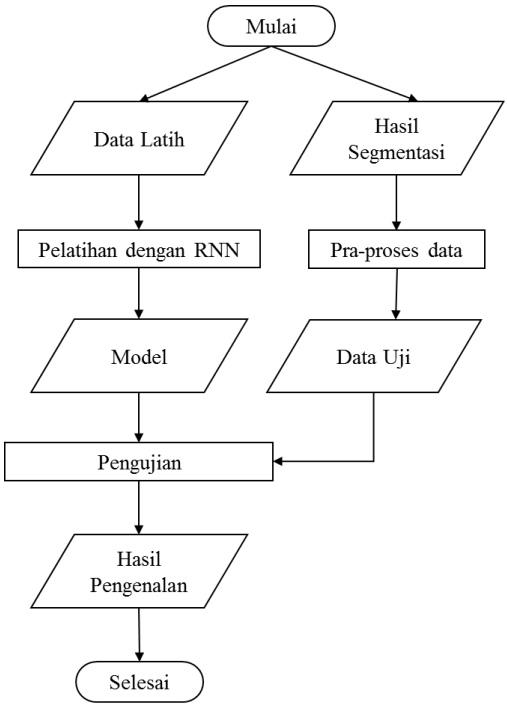
3.2.2.2 Segmentasi dengan Connected Component Labelling (CCL)

Setelah melakukan praproses terhadap citra hasil lokalisasi, segmentasi dilakukan dengan menggunakan bantuan dari *library* Python skimage yaitu modul *measure.label* untuk mendeteksi *blob-blob* pada citra. Seluruh *blob* yang ditemukan nantinya akan difiltrasi berdasarkan *width*, *height*, dan *size ratio* dengan menggunakan bantuan dari *library* yang sama sebelumnya yaitu modul *measure.regionprops*.

Standar *threshold* untuk *width*, *height*, dan *size ratio* ditentukan terlebih dahulu sebelumnya menggunakan analisis perbandingan berdasarkan hasil lokalisasi plat nomor kendaraan.

Apabila *blob* yang diseleksi memenuhi standar *threshold* yang ditentukan, maka *blob* akan ditetapkan sebagai salah satu keluaran atau hasil segmentasi.

3.2.3 Tahap Pengenalan Karakter

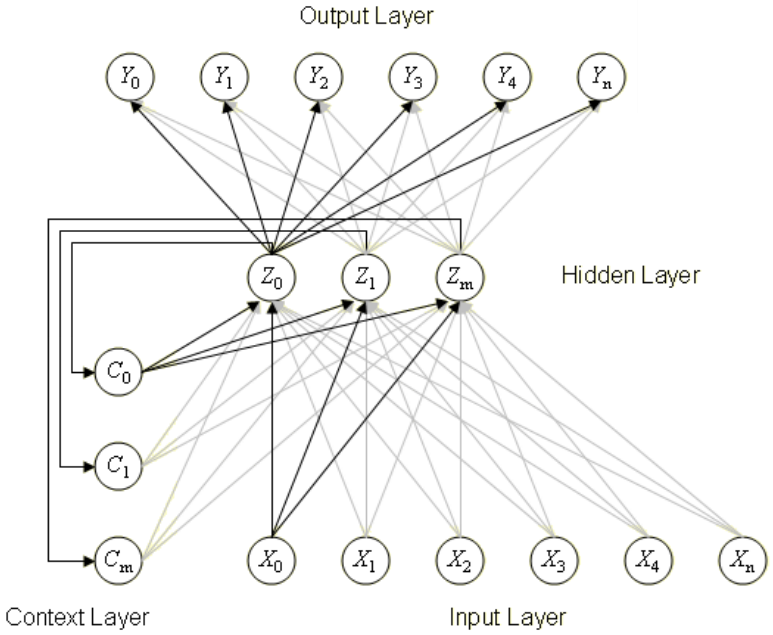


Gambar 3.8 Diagram alir tahap pengenalan karakter

Pada tugas akhir ini, akan dilakukan tahap pengenalan karakter yang berfungsi untuk memberi label kelas pada citra-citra hasil segmentasi. Tahap ini secara garis besar dibagi menjadi 3 bagian, yaitu pembangunan arsitektur RNN, pelatihan, dan pengujian. Diagram alir tahap segmentasi dapat dilihat pada Gambar 3.8.

3.2.3.1 Pembangunan Arsitektur

Pembangunan model bertujuan untuk menyiapkan *layer*, fungsi aktivasi, dan parameter apa saja yang dibutuhkan. Arsitektur RNN dapat dilihat pada Gambar 3.9.



Gambar 3.9 Arsitektur *Elman Network* [30]

Pembangunan arsitektur RNN mengikuti arsitektur *Elman network* yang telah disediakan oleh Keras-TensorFlow dalam tipe *SimpleRNN layer* dan diikuti oleh 1 *Fully Connected layer*.

3.2.3.2 Pelatihan

Untuk pelatihan, data akan dibagi menjadi 2, yaitu data latih dan data validasi dengan jumlah data latih sebanyak 91,333 citra dan jumlah data validasi sebanyak 187,914 citra.

Proses pelatihan memanfaatkan data latih untuk membangun model RNN. Pelatihan menggunakan *activation function* ReLU dan Softmax dengan *learning rate* yang sudah ditentukan sebelumnya. Proses pelatihan akan dijalankan pada *batch size* 128 dan jumlah *epoch* menyesuaikan skenario uji coba. Dalam setiap akhir *epoch* terdapat proses pengujian model terhadap data validasi untuk mengetahui seberapa baik model dilatih. Lalu pada akhir pelatihan akan dilakukan pengujian untuk mendapatkan nilai akurasi, *precision*, dan *recall*.

3.2.3.3 Pengujian

Untuk tahap pengujian, citra hasil dari segmentasi akan diproses kembali terlebih dahulu. Setiap data dalam bentuk citra biner ini akan dilakukan proses *padding*, dengan menambahkan pixel bernilai 0 (warna hitam) hingga membuat rasio citra menjadi 1:1. Citra biner tersebut kemudian akan dilakukan *resize* untuk mengubah ukuran citra menjadi ukuran 32x32.

Setelah melalui tahapan praproses, data siap diuji menggunakan model yang telah terbentuk pada tahap pelatihan. Tahap pengujian ini digunakan untuk mengetahui seberapa baik model dilatih dan diaplikasikan pada lingkungan yang belum pernah dikenalinya. Pada akhir pengujian, akan dilakukan perhitungan untuk mendapatkan nilai akurasi, *precision*, dan *recall*.

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, Matplotlib, dan Scikit-learn, Scikit-image, dan SciPy.

4.2 Implementasi Tahap Lokalisasi

Pada subbab ini akan dijelaskan proses augmentasi data yaitu refleksi dan memperbesar ukuran gambar.

4.2.1 Implementasi Praproses

Sebelum lokalisasi, setiap citra pada data latih akan melalui praproses terlebih dahulu agar lebih bervariasi dan menyamakan ukuran. Praproses ini terdiri dari *brightness transform*, *contrast transform*, *hue transform*, *saturation transform*, *horizontal_flip*, dan *resize*.

Untuk proses *brightness transform*, implementasi dapat dilihat pada Kode Sumber 4.2.1. Pada baris 1, variabel *data* yang merepresentasikan citra yang akan diproses, dikonversi terlebih dahulu menjadi *type float32*. Kemudian, pada baris 2 nilai *delta* untuk penambahan *brightness* ditentukan dengan cara acak pada rentang tertentu menggunakan fungsi *random.randint* dengan parameter -*self.delta* sebagai batas bawah dan *self.delta* sebagai batas atas. Setelah itu pada baris 3, seluruh nilai pada warna pada citra *data* ditambah dengan nilai *delta* yang didapat. Pada baris 4, apabila nilai warna ada yang melebihi 255, maka diubah menjadi 255. Pada baris 5, apabila nilai warna ada yang kurang dari 0, maka diubah menjadi 0. Variabel *data* kemudian diubah menjadi type *int* kembali pada baris 6.

```
1.  data = data.astype(np.float32)
2.  delta = random.randint(-self.delta, self.delta)
3.  data += delta
4.  data[data>255] = 255
5.  data[data<0] = 0
6.  data = data.astype(np.uint8)
```

Kode Sumber 4.2.1 Implementasi *brightness transform*

Untuk proses *contrast transform*, implementasi dapat dilihat pada Kode Sumber 4.2.2. Pada baris 1, variabel *data* yang merepresentasikan citra yang akan diproses, dikonversi terlebih dahulu menjadi *type float32*. Kemudian, pada baris 2 nilai *delta* untuk pengolahan *contrast* ditentukan dengan cara acak pada rentang tertentu menggunakan fungsi *random.uniform* dengan parameter *self.lower* sebagai batas bawah dan *self.upper* sebagai

batas atas. Setelah itu pada baris 3, seluruh nilai pada warna pada citra *data* dikalikan dengan nilai *delta* yang didapat. Pada baris 4, apabila nilai warna ada yang melebihi 255, maka diubah menjadi 255. Pada baris 5, apabila nilai warna ada yang kurang dari 0, maka diubah menjadi 0. Variabel *data* kemudian diubah menjadi type *int* kembali pada baris 6.

```
1.  data = data.astype(np.float32)
2.  delta = random.uniform(self.lower, self.upper)
3.  data *= delta
4.  data[data>255] = 255
5.  data[data<0] = 0
6.  data = data.astype(np.uint8)
```

Kode Sumber 4.2.2 Implementasi *contrast transform*

Untuk proses *hue transform*, implementasi dapat dilihat pada Kode Sumber 4.2.3. Pada baris 1, variabel *data* yang merepresentasikan citra yang akan diproses, dikonversi terlebih dahulu dari format BGR menjadi HSV (*Hue*, *Saturation*, *Value*) menggunakan fungsi *cvtColor* dari OpenCV. Kemudian, pada baris 2 *data* diubah menjadi *type float32*. Pada baris 3, nilai *delta* untuk pemberian *hue* ditentukan dengan cara acak pada rentang tertentu menggunakan fungsi *random.randint* dengan parameter *-self.delta* sebagai batas bawah dan *self.delta* sebagai batas atas. Setelah itu pada baris 4, seluruh nilai pada warna pada citra *data[0]* (*Hue*) ditambah dengan nilai *delta* yang didapat. Karena rentang *hue* berada di antara 0-179, maka pada baris 5, apabila nilai *hue* ada yang melebihi 180, akan dikurangi sebanyak 180. Pada baris 6, apabila nilai *hue* ada yang kurang dari 0, akan ditambah sebanyak 180. Variabel *data* kemudian diubah menjadi type *int* kembali pada baris 7 dan dikonversi kembali menjadi format BGR pada baris 8.

```
1.  data = cv2.cvtColor(data, cv2.COLOR_BGR2HSV)
2.  data = data.astype(np.float32)
3.  delta = random.randint(-self.delta, self.delta)
4.  data[0] += delta
```

```
5.  data[0][data[0]>180] -= 180
6.  data[0][data[0]<0] +=180
7.  data = data.astype(np.uint8)
8.  data = cv2.cvtColor(data, cv2.COLOR_HSV2BGR)
```

Kode Sumber 4.2.3 Implementasi *hue transform*

Untuk proses *saturation transform*, implementasi dapat dilihat pada Kode Sumber 4.2.4. Pada baris 1, variabel *data* yang merepresentasikan citra yang akan diproses, dikonversi terlebih dahulu dari format BGR menjadi HSV (*Hue*, *Saturation*, *Value*) menggunakan fungsi *cvtColor* dari OpenCV. Kemudian, pada baris 2 *data* diubah menjadi *type float32*. Pada baris 3, nilai *delta* untuk pemberian *hue* ditentukan dengan cara acak pada rentang tertentu menggunakan fungsi *random.uniform* dengan parameter *self.lower* sebagai batas bawah dan *self.upper* sebagai batas atas. Setelah itu pada baris 4, seluruh nilai pada warna pada citra *data[1]* (*Saturation*) dikalikan dengan nilai *delta* yang didapat. Pada baris 5, apabila nilai *saturation* ada yang melebihi 255, maka akan diubah menjadi 255. Pada baris 6, apabila nilai *saturation* ada yang kurang dari 0, maka akan diubah menjadi 0. Variabel *data* kemudian diubah menjadi type *int* kembali pada baris 7 dan dikonversi kembali menjadi format BGR pada baris 8.

```
1.  data = cv2.cvtColor(data, cv2.COLOR_BGR2HSV)
2.  data = data.astype(np.float32)
3.  delta = random.uniform(self.lower, self.upper)
4.  data[1] *= delta
5.  data[1][data[1]>255] = 255
6.  data[1][data[1]<0] = 0
7.  data = data.astype(np.uint8)
8.  data = cv2.cvtColor(data, cv2.COLOR_HSV2BGR)
```

Kode Sumber 4.2.4 Implementasi *saturation transform*

Untuk proses *horizontal flip*, implementasi dapat dilihat pada Kode Sumber 4.2.5. Pada baris 1, variabel *data* yang merepresentasikan citra yang akan diproses, dikonversi terlebih

dibalik secara *horizontal* menggunakan fungsi *flip* dari OpenCV dengan parameter *data* sebagai citra yang akan diproses dan 1 sebagai arah *flip* (*horizontal*).

```
1.  data = cv2.flip(data, 1)
```

Kode Sumber 4.2.5 Implementasi *horizontal flip*

Untuk proses *resize*, implementasi dapat dilihat pada Kode Sumber 4.2.6. Pada baris 1, variabel *alg* berfungsi untuk menentukan algoritma interpolasi yang akan digunakan dengan bantuin fungsi *random.choice* terhadap *array* yang berisi algoritma-algoritma OpenCV antara lain *cv2.INTER_LANCZOS4*, *cv2.INTER_AREA*, *cv2.INTER_LINEAR*, *cv2.INTER_NEAREST*, *cv2.INTER_CUBIC*. Kemudian pada baris 2, citra akan di-*resize* dengan fungsi *resize* dari OpenCV dengan parameter *data* sebagai citra yang akan diproses, (*self.width*, *self.height*) sebagai ukuran yang diinginkan, dan *interpolation* sebagai algoritma interpolasi.

```
1.  alg = random.choice(self.algorithms)
2.  resized = cv2.resize(data, (self.width,
    self.height), interpolation=alg)
```

Kode Sumber 4.2.6 Implementasi *resize*

4.2.2 Implementasi Pembangunan Arsitektur

Pada bagian ini akan dijabarkan implementasi fungsi-fungsi pada tahap pembangunan model. Arsitektur SSD dibangun dengan mengimplementasi desain VGG16. Implementasi pemuatan arsitektur VGG16 tercantum pada Kode Sumber 4.2.7.

```
1.  def __load_vgg(self, vgg_dir):
2.      sess = self.session
3.      graph = tf.saved_model.loader.load(sess,
    ['vgg16'], vgg_dir+'/vgg')
4.      self.image_input =
    sess.graph.get_tensor_by_name('image_input:0')
```

```
5.      self.keep_prob   =
    sess.graph.get_tensor_by_name('keep_prob:0')
6.      self.vgg_conv4_3 =
    sess.graph.get_tensor_by_name('conv4_3/Relu:0')
7.      self.vgg_conv5_3 =
    sess.graph.get_tensor_by_name('conv5_3/Relu:0')
8.      self.vgg_fc6_w   =
    sess.graph.get_tensor_by_name('fc6/weights:0')
9.      self.vgg_fc6_b   =
    sess.graph.get_tensor_by_name('fc6/biases:0')
10.     self.vgg_fc7_w   =
    sess.graph.get_tensor_by_name('fc7/weights:0')
11.     self.vgg_fc7_b   =
    sess.graph.get_tensor_by_name('fc7/biases:0')
12.
13.     layers = ['conv1_1', 'conv1_2', 'conv2_1',
    'conv2_2', 'conv3_1',
14.               'conv3_2', 'conv3_3', 'conv4_1',
    'conv4_2', 'conv4_3',
15.               'conv5_1', 'conv5_2', 'conv5_3']
16.
17.     for l in layers:
18.         self.l2_loss +=
    sess.graph.get_tensor_by_name(l+'/L2Loss:0')
```

Kode Sumber 4.2.7 Implementasi pemuatan arsitektur VGG16

Implementasi proses modifikasi tercantum pada Kode Sumber 4.2.8. Baris 3 merupakan bentuk modifikasi terhadap pool5 yang awalnya berukuran 2x2 dengan stride 2 menjadi 3x3 dengan stride 1. Modifikasi pada *fc6* menjadi *covolutional layer* dengan *à trous convolution* diimplementasikan pada baris 6 hingga 21 dengan keterangan baris 7 hingga 14 berfungsi untuk melenyapkan weights dan baris 15 hingga 21 berfungsi untuk membuat *feature maps*. Sedangkan untuk modifikasi pada *fc7* dapat dilihat pada implementasi baris 22 hingga 37 dengan keterangan baris 23 hingga 29 berfungsi untuk melenyapkan weights dan baris 30 hingga 37 berfungsi untuk membuat *feature maps*.

```
1.  def __build_vgg_mods_a_trous(self):
2.      sess = self.session
3.      self.mod_pool5 =
    tf.nn.max_pool(self.vgg_conv5_3, ksize=[1, 3, 3,
    1],
4.                                      strides=[1, 1,
    1, 1], padding='SAME',
5.
    name='mod_pool5')
6.      with tf.variable_scope('mod_conv6'):
7.          orig_w, orig_b = sess.run([self.vgg_fc6_w,
    self.vgg_fc6_b])
8.          mod_w = np.zeros((3, 3, 512, 1024))
9.          mod_b = np.zeros(1024)
10.         for i in range(1024):
11.             mod_b[i] = orig_b[4*i]
12.             for h in range(3):
13.                 for w in range(3):
14.                     mod_w[h, w, :, i] = orig_w[3*h,
    3*w, :, 4*i]
15.         w = array2tensor(mod_w, 'filter')
16.         b = array2tensor(mod_b, 'biases')
17.         x = tf.nn.atrous_conv2d(self.mod_pool5, w,
    rate=6, padding='SAME')
18.         x = tf.nn.bias_add(x, b)
19.         x = tf.nn.relu(x)
20.         self.mod_conv6 = x
21.         self.l2_loss += tf.nn.l2_loss(w)
22.     with tf.variable_scope('mod_conv7'):
23.         orig_w, orig_b = sess.run([self.vgg_fc7_w,
    self.vgg_fc7_b])
24.         mod_w = np.zeros((1, 1, 1024, 1024))
25.         mod_b = np.zeros(1024)
26.         for i in range(1024):
27.             mod_b[i] = orig_b[4*i]
28.             for j in range(1024):
29.                 mod_w[:, :, j, i] = orig_w[:, :,
    4*j, 4*i]
30.         w = array2tensor(mod_w, 'filter')
31.         b = array2tensor(mod_b, 'biases')
32.         x = tf.nn.conv2d(self.mod_conv6, w,
    strides=[1, 1, 1, 1],
33.                          padding='SAME')
```

```
34.         x = tf.nn.bias_add(x, b)
35.         x = tf.nn.relu(x)
36.         self.mod_conv7 = x
37.         self.l2_loss += tf.nn.l2_loss(w)
```

Kode Sumber 4.2.8 Implementasi modifikasi arsitektur

Kemudian, pembangunan model untuk pelatihan yang berdasarkan pada *pre-defined* VGG16 diimplementasikan pada Kode Sumber 4.2.9. Untuk rincian setiap fungsi dapat dilihat berturut-turut pada Kode Sumber 4.2.7, Kode Sumber 4.2.8, Kode Sumber 4.2.10, Kode Sumber 4.2.11, Kode Sumber 4.2.12, dan Kode Sumber 4.2.13.

```
1.  def build_from_vgg(self, vgg_dir, num_classes,
    a_trous=True,
2.                         progress_hook='tqdm'):
3.      self.num_classes = num_classes+1
4.      self.num_vars = num_classes+5
5.      self.l2_loss = 0
6.      self.__load_vgg(vgg_dir)
7.      self.__build_vgg_mods_a_trous()
8.      self.__build_ssd_layers()
9.      self.__build_norms()
10.     self.__select_feature_maps()
11.     self.__build_classifiers()
12.     self.__built = True
```

Kode Sumber 4.2.9 Implementasi fungsi *build_from_vgg*

Untuk pembangunan *layer-layer* SSD yang tercantum pada Kode Sumber 4.2.10.

```
1.  def __build_ssd_layers(self):
2.      stride10 = 1
3.      padding10 = 'VALID'
4.      if len(self.preset.maps) >= 7:
5.          stride10 = 2
6.          padding10 = 'SAME'
```

```
7.      x, l2  = conv_map(self.mod_conv7,    256, 1, 1,
    'conv8_1')
8.      self.ssd_conv8_1 = self.__with_loss(x, l2)
9.      x, l2 = conv_map(self.ssd_conv8_1,  512, 3, 2,
    'conv8_2')
10.     self.ssd_conv8_2 = self.__with_loss(x, l2)
11.     x, l2  = conv_map(self.ssd_conv8_2,  128, 1, 1,
    'conv9_1')
12.     self.ssd_conv9_1 = self.__with_loss(x, l2)
13.     x, l2 = conv_map(self.ssd_conv9_1,  256, 3, 2,
    'conv9_2')
14.     self.ssd_conv9_2 = self.__with_loss(x, l2)
15.     x, l2 = conv_map(self.ssd_conv9_2,  128, 1, 1,
    'conv10_1')
16.     self.ssd_conv10_1 = self.__with_loss(x, l2)
17.     x, l2 = conv_map(self.ssd_conv10_1, 256, 3,
    stride10, 'conv10_2', padding10)
18.     self.ssd_conv10_2 = self.__with_loss(x, l2)
19.     x, l2 = conv_map(self.ssd_conv10_2, 128, 1, 1,
    'conv11_1')
20.     self.ssd_conv11_1 = self.__with_loss(x, l2)
21.     x, l2 = conv_map(self.ssd_conv11_1, 256, 3, 1,
    'conv11_2', 'VALID')
22.     self.ssd_conv11_2 = self.__with_loss(x, l2)
23.     if len(self.preset.maps) < 7:
24.         return
25.     x, l2 = conv_map(self.ssd_conv11_2, 128, 1, 1,
    'conv12_1')
26.     paddings = [[0, 0], [0, 1], [0, 1], [0, 0]]
27.     x = tf.pad(x, paddings, "CONSTANT")
28.     self.ssd_conv12_1 = self.__with_loss(x, l2)
29.     x, l2 = conv_map(self.ssd_conv12_1, 256, 3, 1,
    'conv12_2', 'VALID')
30.     self.ssd_conv12_2 = self.__with_loss(x, l2)
```

Kode Sumber 4.2.10 Implementasi fungsi *__build_ssd_layers*

Untuk implementasi proses *normalization* dapat dilihat pada Kode Sumber 4.2.11.

```
1.  def __build_norms(self):
```

```
2.      x = l2_normalization(self.vgg_conv4_3, 20,
    512, 'l2_norm_conv4_3')
3.      self.norm_conv4_3 = x
```

Kode Sumber 4.2.11 Implementasi fungsi *__build_norms*

Implementasi untuk pemilihan *feature maps* dapat dilihat pada Kode Sumber 4.2.12.

```
1.  def __select_feature_maps(self):
2.      self.__maps = [
3.          self.norm_conv4_3,
4.          self.mod_conv7,
5.          self.ssd_conv8_2,
6.          self.ssd_conv9_2,
7.          self.ssd_conv10_2,
8.          self.ssd_conv11_2]
9.      if len(self.preset.maps) == 7:
10.         self.__maps.append(self.ssd_conv12_2)
```

Kode Sumber 4.2.12 Implementasi fungsi *__select_feature_maps*

Implementasi untuk pembangunan *classifiers* dapat dilihat pada Kode Sumber 4.2.13.

```
1.  def __build_classifiers(self):
2.      with tf.variable_scope('classifiers'):
3.          self.__classifiers = []
4.          for i in range(len(self.__maps)):
5.              fmap = self.__maps[i]
6.              map_size = self.preset.maps[i].size
7.              for j in
    range(2+len(self.preset.maps[i].aspect_ratios)):
8.                  name = 'classifier{}_{}'.format(i,
    j)
9.                  clsfier, l2 = classifier(fmap,
    self.num_vars, map_size, name)
10.
    self.__classifiers.append(self.__with_loss(clsfier,
    l2))
11.     with tf.variable_scope('output'):
```

```
12.         output      = tf.concat(self.__classifiers,
    axis=1, name='output')
13.         self.logits = output[:,:,:self.num_classes]
14.     with tf.variable_scope('result'):
15.         self.classifier =
    tf.nn.softmax(self.logits)
16.         self.locator    =
    output[:,:,self.num_classes:]
17.         self.result     =
    tf.concat([self.classifier, self.locator],
18.                                     axis=-1,
    name='result')
```

Kode Sumber 4.2.13 Implementasi fungsi *__build_classifiers*

Kemudian, implementasi untuk rangkaian proses pembangunan *optimizers* dapat dilihat pada Kode Sumber 4.2.14. Baris 6 hingga 9 merupakan proses untuk pemisahan *ground truth tensor* dengan keterangan baris 7 adalah proses klasifikasi *ground truth tensor* dengan *shape* (*batch_size*, *num_anchors*, *num_classes*), baris 8 merupakan proses lokalisasi *ground truth tensor* dengan *shape* (*batch_size*, *num_anchors*, 4), dan baris 9 merupakan *batch size* dengan *shape scalar*.

Baris 10 hingga 17 merupakan proses untuk komputasi *match counters* dengan keterangan baris 11 hingga 12 adalah jumlah *anchors* per *sample* dengan *shape* (*batch_size*), baris 13 adalah jumlah *anchors* negatif (*not-matched*) per *sample* dengan *shape* (*batch_size*) yang diperoleh dengan menghitung *box-box* pada kelas *background* pada setiap *sample*, baris 14 adalah jumlah *anchors* positif (*matched*) per *sample* dengan *shape* (*batch_size*), dan baris 15 hingga 17 merupakan jumlah *anchors* positif per *sample* dengan *shape* (*batch_size*) yang termasuk pada *division-safe*.

Baris 18 hingga 20 merupakan proses untuk komputasi *masks* dengan keterangan baris 19 adalah *boolean tensor* untuk menentukan apakah *anchor* positif dengan *shape* (*batch_size*, *num_anchors*) dan baris 20 adalah *boolean tensor* untuk

menentukan apakah *anchor* negatif dengan *shape* (*batch_size*, *num_anchors*).

Baris 21 hingga 63 merupakan proses untuk menghitung *confidence loss*. Baris 22 hingga 23 merupakan *cross-entropy sensor* dengan *shape* (*batch_size*, *num_anchors*). Baris 24 hingga 25 berfungsi untuk menyimpan total *loss* dari semua *anchor* positif. Baris 26 hingga 27 berfungsi untuk mencari tahu *anchor* negatif dengan *confidence loss* tertinggi. Baris 28 berfungsi untuk mencari tahu banyaknya *anchor* negatif yang ingin disimpan. Baris 29 hingga 35 berfungsi untuk memberi *mask* pada *anchor* negatif yang berlebihan dan menghitung total *loss*-nya dengan keterangan baris 29 merupakan vektor *transpose* dari negatif maksimum per *sample* dengan *shape* (*batch_size*, 1), baris 30 merupakan *range tensor* yang berentang dari 0 hingga *num_anchors* – 1 dengan *shape* (*num_anchors*), baris 31 merupakan *row* dari *range* sama seperti baris sebelumnya namun diubah menjadi *int64* dan sebuah *row* matriks dengan *shape* (1, *num_anchors*), baris 32 merupakan *mask* dari negatif maksimum dengan *shape* (*batch_size*, *num_anchors*), baris 33 hingga 34 merupakan negatif maksimum dengan *shape* (*batch_size*, *num_anchors*), dan baris 35 merupakan total dari negatif maksimum pada tiap *sample* dengan *shape* (*batch_size*). Baris 36 hingga 42 berfungsi untuk menghitung *confidence loss* pada setiap elemen dengan keterangan baris 36 merupakan total *confidence loss* untuk setiap *sample* dengan *shape* (*batch_size*), baris 37 hingga 40 merupakan total *confidence loss* yang dinormalisasi berdasarkan jumlah positif per *sample* dengan *shape* (*batch_size*), dan baris 41 hingga 42 merupakan *mean confidence loss* pada *batch* dengan *shape scalar*.

Baris 43 hingga 55 merupakan proses untuk menghitung *localization loss*. Baris 44 merupakan perbedaan *element-wise* antara *localization loss* yang diprediksi dan *ground truth* dengan *shape* (*batch_size*, *num_anchors*, 4). Baris 45 adalah *smooth L1 loss* dengan *shape* (*batch_size*, *num_anchors*, 4). Baris 46 adalah total dari *localization loss* untuk setiap *anchor* dengan *shape* (*batch_size*, *num_anchors*). Baris 47 hingga 48 adalah *postive*

localization dengan *shape* (*batch_size*, *num_anchors*). Baris 49 merupakan total *loss* dari *anchor* positif dengan *shape* (*batch_size*). Baris 50 hingga 53 merupakan total *localization loss* yang dinormalisasi berdasarkan jumlah positif per *sample* dengan *shape* (*batch_size*). Baris 54 hingga 55 merupakan *mean localization loss* pada *batch* dengan *shape scalar*.

Baris 56 hingga 63 merupakan proses untuk menghitung total *loss*. Baris 57 hingga 59 merupakan total dari *localization loss* dan *confidence loss* dengan *shape* (*batch_size*). Baris 60 hingga 61 merupakan *L2 loss* dengan *shape scalar* dan baris 62 hingga 63 merupakan *final loss* dengan *shape scalar*. Kemudian, baris 64 hingga 67 berfungsi untuk membangun *optimizer* dan baris 68 hingga 74 berfungsi untuk menyimpan *tensor-tensor*.

```
1.  def build_optimizer(self, learning_rate=0.001,
    weight_decay=0.0005,
2.                          momentum=0.9,
    global_step=None):
3.      self.labels = tf.placeholder(tf.float32,
    name='labels',
4.                                   shape=[None, None,
    self.num_vars])
5.
6.      with tf.variable_scope('ground_truth'):
7.          gt_cl = self.labels[:,:,:self.num_classes]
8.          gt_loc = self.labels[:,:,self.num_classes:]
9.          batch_size = tf.shape(gt_cl)[0]
10.     with tf.variable_scope('match_counters'):
11.         total_num = tf.ones([batch_size],
    dtype=tf.int64) * \
12.
    tf.to_int64(self.preset.num_anchors)
13.         negatives_num =
    tf.count_nonzero(gt_cl[:,:,-1], axis=1)
14.         positives_num = total_num-negatives_num
15.         positives_num_safe =
    tf.where(tf.equal(positives_num, 0),
16.
    tf.ones([batch_size])*10e-15,
```

```
17.
    tf.to_float(positives_num))
18.     with tf.variable_scope('match_masks'):
19.         positives_mask = tf.equal(gt_cl[:,:,-1], 0)
20.         negatives_mask =
    tf.logical_not(positives_mask)
21.     with tf.variable_scope('confidence_loss'):
22.         ce =
    tf.nn.softmax_cross_entropy_with_logits_v2(labels=g
    t_cl,
23.
    logits=self.logits)
24.         positives = tf.where(positives_mask, ce,
    tf.zeros_like(ce))
25.         positives_sum = tf.reduce_sum(positives,
    axis=-1)
26.         negatives = tf.where(negatives_mask, ce,
    tf.zeros_like(ce))
27.         negatives_top = tf.nn.top_k(negatives,
    self.preset.num_anchors)[0]
28.         negatives_num_max =
    tf.minimum(negatives_num, 3*positives_num)
29.         negatives_num_max_t =
    tf.expand_dims(negatives_num_max, 1)
30.         rng = tf.range(0, self.preset.num_anchors,
    1)
31.         range_row = tf.to_int64(tf.expand_dims(rng,
    0))
32.         negatives_max_mask = tf.less(range_row,
    negatives_num_max_t)
33.         negatives_max =
    tf.where(negatives_max_mask, negatives_top,
34.
    tf.zeros_like(negatives_top))
35.         negatives_max_sum =
    tf.reduce_sum(negatives_max, axis=-1)
36.         confidence_loss = tf.add(positives_sum,
    negatives_max_sum)
37.         confidence_loss =
    tf.where(tf.equal(positives_num, 0),
38.
    tf.zeros([batch_size]),
```

```
39.
    tf.div(confidence_loss,
40.
    positives_num_safe))
41.         self.confidence_loss =
    tf.reduce_mean(confidence_loss,
42.
    name='confidence_loss')
43.     with tf.variable_scope('localization_loss'):
44.         loc_diff = tf.subtract(self.locator,
    gt_loc)
45.         loc_loss = smooth_l1_loss(loc_diff)
46.         loc_loss_sum = tf.reduce_sum(loc_loss,
    axis=-1)
47.         positive_locs = tf.where(positives_mask,
    loc_loss_sum,
48.
    tf.zeros_like(loc_loss_sum))
49.         localization_loss =
    tf.reduce_sum(positive_locs, axis=-1)
50.         localization_loss =
    tf.where(tf.equal(positives_num, 0),
51.
    tf.zeros([batch_size]),
52.
    tf.div(localization_loss,
53.
    positives_num_safe))
54.         self.localization_loss =
    tf.reduce_mean(localization_loss,
55.
    name='localization_loss')
56.     with tf.variable_scope('total_loss'):
57.         self.conf_and_loc_loss =
    tf.add(self.confidence_loss,
58.
    self.localization_loss,
59.
    name='sum_losses')
60.         self.l2_loss = tf.multiply(weight_decay,
    self.l2_loss,
61.                                    name='l2_loss')
```

```
62.         self.loss = tf.add(self.conf_and_loc_loss,
    self.l2_loss,
63.                            name='loss')
64.     with tf.variable_scope('optimizer'):
65.         optimizer =
    tf.train.MomentumOptimizer(learning_rate, momentum)
66.         optimizer = optimizer.minimize(self.loss,
    global_step=global_step,
67.
    name='optimizer')
68.     self.optimizer = optimizer
69.     self.losses = {
70.         'total': self.loss,
71.         'localization': self.localization_loss,
72.         'confidence': self.confidence_loss,
73.         'l2': self.l2_loss
74.     }
```

Kode Sumber 4.2.14 Implementasi fungsi *build_optimizer*

4.2.3 Implementasi Pelatihan

Setelah pembangunan arsitektur *Single Shot Detector*, dilakukan proses pelatihan dan pembuatan model. Implementasi untuk proses pelatihan data dapat dilihat pada Kode Sumber 4.2.15.

```
1.  with tf.Session() as sess:
2.      n_train_batches                                 =
    int(math.ceil(td.num_train/args.batch_size))
3.      global_step = None
4.      lr_values = args.lr_values.split(';')
5.      lr_values = [float(x) for x in lr_values]
6.      lr_boundaries = args.lr_boundaries.split(';')
7.      lr_boundaries = [int(x) for x in lr_boundaries]
8.      ret = compute_lr(lr_values, lr_boundaries)
9.      learning_rate, global_step = ret
10.     net = SSDVGG(sess, td.preset)
11.     net.build_from_vgg(args.vgg_dir,
    td.num_classes)
12.
    net.build_optimizer(learning_rate=learning_rate,
13.                         global_step=global_step,
```

```
14.
    weight_decay=args.weight_decay,
15.                           momentum=args.momentum)
16.     initialize_uninitialized_variables(sess)
17.     summary_writer                                    =
    tf.summary.FileWriter(args.tensorboard_dir,
18.
    sess.graph)
19.     saver = tf.train.Saver(max_to_keep=20)
20.     anchors = get_anchors_for_preset(td.preset)
21.     training_ap_calc = APCalculator()
22.     restore = False
23.     training_ap       =       PrecisionSummary(sess,
    summary_writer, 'training',
24.
    td.lname2id.keys(), restore)
25.     training_imgs        =        ImageSummary(sess,
    summary_writer, 'training',
26.                                    td.label_colors,
    restore)
27.     training_loss         =         LossSummary(sess,
    summary_writer, 'training',
28.                                   td.num_train,
    restore)
29.     net_summary_ops = net.build_summaries(restore)
30.     net_summary = sess.run(net_summary_ops)
31.     summary_writer.add_summary(net_summary, 0)
32.     summary_writer.flush()
33.     print('[i] Training...')
34.     for e in range(start_epoch, args.epochs):
35.         training_imgs_samples = []
36.         generator                                     =
    td.train_generator(args.batch_size,
    args.num_workers)
37.         description       =       '[i]       Train
    {:>2}/{}'.format(e+1, args.epochs)
38.         for  x,  y,  gt_boxes  in  tqdm(generator,
    total=n_train_batches,
39.
    desc=description, unit='batches'):
40.             if len(training_imgs_samples) < 3:
41.                 saved_images = np.copy(x[:3])
42.             feed = {net.image_input: x,
```

```
43.                     net.labels: y}
44.             result,       loss_batch,       _       =
    sess.run([net.result, net.losses,
45.
    net.optimizer],
46.
    feed_dict=feed)
47.             training_loss.add(loss_batch,
    x.shape[0])
48.             for i in range(result.shape[0]):
49.                 boxes    =    decode_boxes(result[i],
    anchors, 0.5, td.lid2name)
50.                 boxes = suppress_overlaps(boxes)
51.
    training_ap_calc.add_detections(gt_boxes[i], boxes)
52.                 if len(training_imgs_samples) < 3:
53.
    training_imgs_samples.append((saved_images[i],
    boxes))
54.         training_loss.push(e+1)
55.         net_summary = sess.run(net_summary_ops)
56.         summary_writer.add_summary(net_summary,
    e+1)
57.         APs = training_ap_calc.compute_aps()
58.         mAP = APs2mAP(APs)
59.         training_ap.push(e+1, mAP, APs)
60.         mAP = APs2mAP(APs)
61.         training_ap_calc.clear()
62.         training_imgs.push(e+1,
    training_imgs_samples)
63.         summary_writer.flush()
64.         print('[i] mAP:', mAP)
65.         if (e+1) % args.checkpoint_interval == 0:
66.             checkpoint                              =
    '{}/e{}.ckpt'.format(args.name, e+1)
67.             saver.save(sess, checkpoint)
68.             print('[i]      Checkpoint      saved:',
    checkpoint)
69.     checkpoint = '{}/final.ckpt'.format(args.name)
70.     saver.save(sess, checkpoint)
```

Kode Sumber 4.2.15 Implementasi proses pelatihan SSD

Baris 2 berfungsi untuk menghitung ukuran tiap *batch*. Baris 4 hingga 5 merupakan pemisahan *learning rate values* yang terdiri dari *array* yang berisi *'0.00075;0.0001;0.00001'* dan mengubahnya menjadi *type float*. Baris 6 hingga 7 merupakan pemisahan *learning rate boundaries* yang terdiri dari *array* yang berisi *'320000;400000'* dan mengubahnya menjadi *type int*. Pada baris 10, variabel *net* merupakan inisialisasi dari pemanggilan *class SSDVGG* yang menyimpan fungsi-fungsi untuk pembentukan arsitektur. Baris 12 dan 13 berturut-turut memanggil fungsi *build_from_vgg* dan *build_optimizer* yang telah dijelaskan pada tahap sebelumnya. Baris 16 berfungsi untuk inisiasi *weights* yang belum terinisiasi oleh yang lain, seperti *metagraph* dan *checkpoint*. Baris 20 berfungsi untuk mendapatkan *anchor-anchor* dari *preset* yang digunakan. Baris 21 berfungsi untuk memanggil *class APCalculator* yang berfungsi untuk menghitung *average precision*. Baris 34 hingga 68 merupakan proses pelatihan dan pembangunan model. Kemudian, baris 69 hingga 70 berfungsi untuk menyimpan *final checkpoint* dari atau *checkpoint* dari *epoch* terakhir.

4.2.4 Implementasi Pengujian

Pada tahap pengujian, data yang digunakan sebagai data uji adalah data video hasil dari rekam kamera CCTV. Sebelum diuji, setiap video akan melalui praproses yang telah dijelaskan pada tahap sebelumnya terlebih dahulu untuk menyamakan ukuran. Setelah itu, dilakukan pengujian yang implementasinya dapat dilihat pada Kode Sumber 4.2.16.

```
1.  graph_def = tf.GraphDef()
2.  with open(args.model, 'rb') as f:
3.      serialized = f.read()
4.      graph_def.ParseFromString(serialized)
5.  with open(args.test_data, 'rb') as f:
6.      data = pickle.load(f)
7.      preset = data['preset']
8.      colors = data['colors']
9.      lid2name = data['lid2name']
```

```
    anchors = get_anchors_for_preset(preset)
if not os.path.exists(args.output_dir):
    os.makedirs(args.output_dir)
with tf.Session() as sess:
    tf.import_graph_def(graph_def, name='detector')
    img_input =
sess.graph.get_tensor_by_name('detector/image_input
:0')
    result =
sess.graph.get_tensor_by_name('detector/result/resu
lt:0')
    files = sys.argv[1:]
    for i in tqdm(range(0, len(files),
args.batch_size)):
        batch_names = files[i:i+args.batch_size]
        batch_imgs = []
        batch = []
        for f in batch_names:
            img = cv2.imread(f)
            batch_imgs.append(img)
            img = cv2.resize(img, (300, 300))
            batch.append(img)
        batch = np.array(batch)
        feed = {img_input: batch}
        enc_boxes = sess.run(result,
feed_dict=feed)
        for i in range(len(batch_names)):
            boxes = decode_boxes(enc_boxes[i],
anchors, 0.5, lid2name, None)
            boxes = suppress_overlaps(boxes)[:200]
            name = os.path.basename(batch_names[i])
            with open(os.path.join(args.output_dir,
name+'.txt'), 'w') as f:
                for box in boxes:
                    draw_box(batch_imgs[i], box[1],
colors[box[1].label])
                    box_data = '{} {} {} {} {}
{}\n'.format(box[1].label,
                        box[1].labelid,
box[1].center.x, box[1].center.y,
                        box[1].size.w,
box[1].size.h)
                    f.write(box_data)
```

```
41.
    cv2.imwrite(os.path.join(args.output_dir, name),
42.                          batch_imgs[i])
```

Kode Sumber 4.2.16 Implementasi proses pengujian SSD

Baris 1 hingga 10 merupakan proses untuk memuat model dan data uji. Baris 11 hingga 12 berfungsi untuk membuat direktori untuk menyimpan hasil keluaran atau deteksi. Kemudian, baris 13 higga 42 merupakan proses untuk menjalankan pengujian atau proses deteksi plat nomor kendaraan dalam kumpulan *batch*.

4.3 Implementasi Tahap Segmentasi

Pada subbab ini akan dijelaskan praproses data dan proses segmentasi karakter.

4.3.1 Implementasi Praproses

Pada subbab ini, akan dijabarkan implementasi pada tahap praproses data yaitu pembacaan *grayscale*, pemberian *Gaussian Blur*, pengaplikasian CLAHE, *Otsu thresholding*, serta konversi citra ke Biner.

4.3.1.1 Implementasi Pembacaan Grayscale

Proses perubahan kanal citra diimplementasikan pada Kode Sumber 4.3.1. Proses perubahan kanal citra yang awalnya 3 kanal menjadi 1 kanal dilakukan pada citra dengan memanfaatkan fungsi *imread* dari OpenCV, dimana parameter yang digunakan adalah *files* sebagai *input* citra yang akan diubah kanalnya dan *cv2.IMREAD_GRAYSCALE* sebagai mode pembacaan kanal.

```
1.  img = cv2.imread(files, cv2.IMREAD_GRAYSCALE)
```

Kode Sumber 4.3.1 Implementasi pembacaan *grayscale*

4.3.1.2 Implementasi Pemberian Gaussian Blur

Proses pemberian efek *Gaussian Blur* pada citra bertujuan untuk menghilangkan *noise* yang dapat mempengaruhi proses

segmentasi. Tahapan ini diimplementasikan pada Kode Sumber 4.3.2. Pemberian *Gaussian Blur* pada citra dilakukan dengan memanfaatkan fungsi *GaussianBlur* dari OpenCV, dimana parameter yang digunakan adalah *img* sebagai *input* citra sebelumnya yang akan diubah teksturnya, *(3, 3)* sebagai kernel *size*, dan *0* sebagai nilai standar deviasi kernel.

```
1.  img = cv2.GaussianBlur(img, (3,3),0)
```

Kode Sumber 4.3.2 Implementasi pemberian *Gaussian Blur*

4.3.1.3 Implementasi Pengaplikasian CLAHE

Proses pengaplikasian CLAHE pada citra bertujuan menyamaratakan tingkat kontras supaya mendapatkan hasil yang maksimal ketika proses konversi citra ke biner nantinya. CLAHE adalah *Contrast Limited Adaptive Histogram Equalization* yang mana merupakan perluasan dari metode *Histogram Equalization* itu sendiri. CLAHE diaplikasikan dengan cara menentukan kernel matriks dan bekerja dengan menggantikan nilai intensitas setiap *pixel* citra masukan dengan rata-rata dari nilai pembobotan kernel untuk setiap *pixel-pixel* tetangganya dan *pixel* itu sendiri. Tahapan ini diimplementasikan pada Kode Sumber 4.4.3. Pengaplikasian CLAHE pada citra dilakukan dengan memanfaatkan fungsi *createCLAHE* dari OpenCV, dimana parameter yang digunakan adalah *clipLimit* sebagai nilai dimana CLAHE membatasi amplifikasi dengan memotong histogram dan *tileGridSize* merujuk pada ukuran *grid* untuk *histogram equalization*.

```
1.  clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize
    = (8,8))
2.  img = clahe.apply(img)
```

Kode Sumber 4.3.3 Implementasi pengaplikasian CLAHE

4.3.1.4 Implementasi Otsu Thresholding

Proses *Otsu thresholding* merupakan proses pencarian nilai terbaik yang membuat sebuah citra 1 kanal menjadi sebuah citra biner. Dalam hal ini, fungsi *threshold_otsu* Scikit-image akan digunakan untuk melakukan proses ini seperti pada Kode Sumber 4.3.4. Fungsi *threshold_otsu* pada Scikit-image sendiri akan memberikan *return* nilai *threshold* dengan rentang antara 0 hingga 255.

```
1.  threshold_value = threshold_otsu(img)
```

Kode Sumber 4.3.4 Implementasi *Threshold Otsu*

4.3.1.5 Implementasi Konversi Citra ke Biner

Citra biner merupakan citra yang setiap pikselnya hanya bernilai 1 atau 0. Tahapan ini diimplementasikan pada Kode Sumber 4.3.5. Proses ini membandingkan setiap nilai piksel pada citra dengan *threshold_value*. Apabila nilai piksel yang dibandingkan kurang dari sama dengan *threshold_value*, maka piksel saat ini akan bernilai 0. Apabila nilai piksel yang dibandingkan lebih dari *threshold_value*, maka piksel saat ini akan bernilai 1.

```
1.  binary_image = out > threshold_value
```

Kode Sumber 4.3.5 Implementasi konversi citra ke biner

4.3.2 Implementasi Segmentasi Citra

Proses segmentasi citra plat nomor kendaraan untuk mendapatkan kandidat karakter diimplementasikan pada Kode Sumber 4.3.6. Fungsi yang digunakan adalah fungsi *measure.label* dan *measure.regionprops* dari *library* Scikit-image.

```
1.  scale = 0.35
2.  license_plate = binary_image
```

```
3.  img_h, img_w = license_plate.shape
4.  img_scale = img_h/img_w
5.  labelled_plate = measure.label(license_plate)
6.  min_w = 0.025 * img_scale/scale
7.  max_w = 0.10 * img_scale/scale
8.  min_h = 0.29 * scale/img_scale
9.  max_h = 0.47 * scale/img_scale
10. character_dimensions = (min_h*img_h, max_h*img_h,
    min_w*img_w, max_w*img_w)
11. min_height, max_height, min_width, max_width =
    character_dimensions
12. for regions in regionprops(labelled_plate):
13.     y0, x0, y1, x1 = regions.bbox
14.     region_height = y1 - y0
15.     region_width = x1 - x0
16.     if region_height > min_height and region_height
    < max_height and region_width > min_width and
    region_width < max_width:
17.         characters.append([y0,x0,y1,x1])
```

Kode Sumber 4.3.6 Implementasi proses segmentasi citra

Baris 1 merupakan penetapan awal standar skala rasio height terhadap *width* pada plat nomor kendaraan pada variabel *scale*. Untuk mendeteksi *blob* itu sendiri, dapat dilihat pada implementasi baris 5 menggunakan bantuan fungsi *measure.label* dari *library* Scikit-image. Fungsi ini mengambil 1 parameter yaitu citra yang akan dipindai, yang mana merupakan citra yang telah dijadikan citra biner dan telah melalui praproses pada tahap sebelumnya. Baris 6 hingga 9 merupakan *threshold* untuk menentukan apakah *blob* yang terdeteksi merupakan kandidat karakter. Hasil dari fungsi *measure.label* kemudian diidentifikasi pada implementasi baris 12 hingga 17 apakah memenuhi rentang *threshold* yang ditetapkan dengan bantuan fungsi *measure.regionprops* dari *library* Scikit-image. Rasio-rasio yang digunakan di sini didapatkan berdasarkan percobaan pada area plat yang telah didapatkan dan dilakukan *cropping* secara manual serta perhitungan langsung dari program dengan melihat semua kandidat area tanpa syarat.

4.4 Implementasi Pengenalan Karakter

Pada subbab ini akan dijabarkan implementasi pada tahap pengenalan karakter yaitu pembangunan arsitektur *Recurrent Neural Network* (RNN), pelatihan, dan pengujian.

4.4.1 Implementasi Pembangunan Arsitektur

Pada bagian ini akan dijabarkan implementasi fungsi-fungsi pada tahap pembangunan model. Arsitektur RNN dimulai dengan mengimplementasi desain *SimpleRNN layer* dari Keras-TensorFlow. Implementasi tercantum pada Kode Sumber 4.4.1 dengan fungsi *SimpleRNN* dengan arsitektur sesuai dengan spesifikasi yang dijelaskan pada Bab 3. Lalu pada tahap selanjutnya, hasil *SimpleRNN* akan diteruskan ke 1 *Fully Connected layer* hingga akhirnya akan dimasukkan ke fungsi aktivasi *Softmax*. Pembangunan arsitektur memanfaatkan Keras.

```
1.  model = Sequential()
2.  model.add(SimpleRNN(hidden_units,
3.            kernel_initializer                          =
    initializers.RandomNormal(stddev = 0.001),
4.                        recurrent_initializer           =
    initializers.Identity(gain = 1.0),
5.                        Activation = 'relu',
6.                        input_shape                     =
    x_train.shape[1:]))
7.  model.add(Dense(num_classes))
8.  model.add(Activation('softmax'))
```

Kode Sumber 4.4.1 Implementasi pembangunan arsitektur RNN

SimpleRNN adalah implementasi arsitektur yang mengadaptasi *Elman network* yang disediakan oleh Keras-TensorFlow. *SimpleRNN* menggunakan 5 parameter. Berikut penjelasan parameter-parameter tersebut:

1. Parameter *hidden_units* adalah dimensi dari *space* keluaran.
2. Parameter *kernel_initializer* adalah *initializer* untuk matriks bobot kernel, digunakan untuk transformasi linear masukan.

3. Parameter *recurrent_initializer* adalah *initializer* untuk matriks bobot *recurrent_kernel*, digunakan untuk transformasi linear dari kondisi berulang.
4. Parameter a*ctivation* adalah fungsi aktivasi apa yang digunakan, terdapat banyak pilihan fungsi aktivasi yang disediakan oleh Keras, salah satunya adalah 'relu' yakni fungsi ReLU.
5. Parameter *input_shape* untuk menentukan ukuran masukan.

Pada baris 7, *Dense* adalah implementasi *Fully Connected layer*. *Dense* menggunakan 2 parameter. Berikut penjelasan parameter-parameter tersebut:

1. Parameter *num_classes* adalah jumlah neuron.
2. Parameter *activation* adalah fungsi aktivasi apa yang digunakan. Terdapat banyak pilihan fungsi aktivasi yang disediakan oleh Keras, ada 'relu' atau fungsi ReLU dan 'softmax' atau fungsi *Softmax* yang biasa digunakan pada akhir arsitektur untuk mengklasifikasikan label kelas. Fungsi aktivasi dapat dituliskan dalam baris yang sama atau pada baris selanjutnya seperti baris 8 pada implementasi.

Pemanggilan fungsi pembangunan RNN diimplementasikan pada Kode Sumber 4.4.2.

```
1.  model.compile(loss='categorical_crossentropy',
2.                optimizer=rmsprop,
3.                metrics=['accuracy'])
```

Kode Sumber 4.4.2 Pemanggilan fungsi pembangunan arsitektur RNN

4.4.2 Implementasi Pelatihan

Setelah pembangunan arsitektur *Recurrent Neural Network*, dilakukan proses pelatihan dan pembuatan model. Data akan dibagi menjadi 2, 70% data latih, dan 30% data validasi, yaitu data latih dan data validasi dengan jumlah data latih sebanyak 187,914 citra

dan jumlah data validasi sebanyak 91,333 citra. Implementasi pemisahan data dapat dilihat pada Kode Sumber 4.4.3.

```
71. X_trains = []
72. y_trains = []
73. root = new/train'
74. folder = [f for f in listdir(root) if
    isdir(join(root, f))]
75. for num, i in enumerate(folder):
76.     mypath_files = []
77.     mypath = root + '/' + i
78.     files = [f for f in listdir(mypath) if
    isfile(join(mypath, f))]
79.     for fn in files:
80.         filename = mypath + '/' + fn
81.         mypath_files.append([fn, filename])
82.     random.shuffle(mypath_files)
83.     new_dir = 'new/test/' + i + '/'
84.     os.makedirs(new_dir)
85.     num_split = len(mypath_files)//2
86.     for x in tqdm(mypath_files[:num_split],
    desc=new_dir, unit='img'):
87.         img = cv2.imread(x[1])
88.         img_dir = new_dir + x[0]
89.         cv2.imwrite(img_dir, img)
90.         os.remove(x[1])
```

Kode Sumber 4.4.3 Implementasi pemisahan data

Baris 3 merupakan inisialisasi penetapan *path* data awal disimpan pada variabel *root*. Data awal mula-mula disimpan di folder ‘new/train/’ dengan subfolder 0-9 dan a-z di dalamnya yang menandakan kelas-kelas data. Baris 4 berfungsi untuk mendapatkan seluruh subfolder pada ‘new/train/’ dan menyimpannya pada array variabel *folder*. Mulai dari baris 5, akan dilakukan iterasi terhadap setiap folder yang tersimpan pada variabel *folder*. Pada baris 6, variabel *mypath_files* berfungsi untuk menampung seluruh data pada setiap kelas. *Path* untuk setiap *file* akan didapatkan terlebih dahulu pada baris 8 dan disimpan pada variabel *files*. Pada baris 12, variabel *mypath_files* akan di-*random*

supaya data tersebar secara merata. Kemudian, akan dibuat folder baru untuk menampung data validasi pada baris 14. Pada baris 15, variabel *num_split* menandakan jumlah *file* yang akan ditetapkan sebagai data validasi. Karena pembagian data latih dan data validasi berrasio 7:3, maka ukuran dari variabel *mypath_files* dikalikan dengan 0.3. Setelah itu, pemindahan *file* sebanyak jumlah num_*split* dari folder *train* ke folder *test* dilakukan pada baris 16 hingga 20. Iterasi dilakukan hingga semua kelas berhasil dipecah datanya.

Proses pelatihan memanfaatkan data latih untuk membangun model RNN. Pelatihan menggunakan *loss function* berupa *Cross Entropy* dan *optimizer* berupa *RMSProp Optimizer* dengan *learning rate* yang telah ditentukan sebelumnya. Hal ini diimplementasikan pada Kode Sumber 4.4.4 di baris 1 dan 2. Proses pelatihan akan dijalankan pada *batch size* 128 dan jumlah *epoch* 50. Dalam setiap akhir *epoch* terdapat proses pengujian model terhadap data validasi dan didapatkan nilai akurasi. Pada baris 5 dilakukan proses pelatihan dengan memanggil fungsi *fit* dari Keras, dimana *batch_size* adalah ukuran *batch*, *epochs* adalah jumlah *epoch* yang akan dijalankan, *validation_data* adalah data validasi dan label kelasnya.

```
1.  rmsprop = RMSprop(learning_rate=learning_rate)
2.  model.compile(loss='categorical_crossentropy',
3.                optimizer=rmsprop,
4.                metrics=['accuracy'])
5.  history = model.fit(x_train, y_train,
6.            batch_size=batch_size,
7.            epochs=epochs,
8.            verbose=1,
9.            validation_data=(x_test, y_test))
```

Kode Sumber 4.4.4 Implementasi proses pelatihan RNN

Setelah proses pelatihan selesai, model akan diuji dengan menggunakan data validasi. Hal ini diimplementasikan pada Kode

Sumber 4.4.5. Fungsi *evaluate* akan mengevaluasi model dengan data validasi dan labelnya.

```
1.  scores = model.evaluate(x_test, y_test, verbose=0)
```

Kode Sumber 4.4.5 Evaluasi pelatihan RNN

Pada fungsi *evaluate*, didapatkan nilai akurasi model terhadap data validasi. Namun *library* Keras ini tidak memiliki fitur untuk menghitung *precision* dan *recall*. Sehingga perlu dilakukan cara alternatif untuk melakukan evaluasi terhadap model agar mendapatkan nilai *precision* dan *recall*. Dengan memanfaatkan fungsi *predict* dari *library* Keras terhadap *x_test* (data validasi). Hal ini diimplementasikan pada Kode Sumber 4.4.6. Pada baris 7, fungsi *predict* digunakan untuk mendapatkan nilai prediksi untuk masing-masing kelas dan disimpan pada variabel *y_pred*. Pada baris 1-5, dibuat sebuah fungsi *return_to_label* untuk mengembalikan bentuk data prediksi dari Keras yang masih berupa tipe kelas ketegorikal menjadi tipe kelas yang numerik. Lalu pada baris 9 hingga 10, fungsi *return_to_label* digunakan untuk mengembalikan *pred_y* dan *test_y* ke tipe kelas numerik. Baris 12 memanfaatkan fungsi *classification_report* dari *library* Scikit-learn untuk mengevaluasi nilai prediksi (*pred*) terhadap nilai sesungguhnya (*test*) untuk masing-masing kelas sehingga didapatkan nilai *precision* dan *recall*.

```
1.  def return_to_label(y):
2.      label = []
3.      for i in range(len(y)):
4.          label.append(np.argmax(y[i]))
5.      return label
6.
7.  y_pred = model.predict(x_test)
8.
9.  test = return_to_label(y_test)
10. pred = return_to_label(y_pred)
11.
12. print(classification_report(test, pred))
```

Kode Sumber 4.4.6 Evaluasi *precision* dan *recall*

4.4.3 Implementasi Pengujian

Pada tahap pengujian, data yang digunakan sebagai data uji adalah citra-citra karakter hasil keluaran dari proses segmentasi karakter. Sebelum diuji, setiap citra akan melalui praproses terlebih dahulu untuk menyamakan ukuran. Praproses ini terdiri dari pemberian *padding* dan *resize* ukuran gambar menjadi 32x32. Implementasi praproses untuk data uji dapat dilihat pada Kode Sumber 4.4.7.

```
1.  post_char = []
2.  for c in char_plates:
3.      pc = []
4.      for img in c:
5.          BLACK = [0,0,0]
6.          h, w = img.shape[:2]
7.          if h > w:
8.              top = int(h*0.05)
9.              left = int((h + 2 * top - w) / 2.0)
10.         else:
11.             left = int(w*0.05)
12.             top = int((w + 2 * left - h) / 2.0)
13.         img = np.float32(img)
14.         constant =
    cv2.copyMakeBorder(img,top,top,left,left,cv2.BORDER
    _CONSTANT,value=BLACK) # top,bottom,left,right
15.         size = 32
16.         constant = misc.imresize(constant, (size,
    size))
17.         pc.append(constant)
18.     post_char.append(pc)
```

Kode Sumber 4.4.7 Implementasi praproses data uji

Pada baris 1, variabel *post_char* berfungsi untuk menampung citra karakter setelah di-praproses. Mula-mula, dilakukan iterasi pada variabel *char_plates* yang berisi citra sebelum di-praproses yang dapat dilihat pada baris 2. Setiap *item*

pada variabel *char_plates* berisi kumpulan citra karakter dari satu plat nomor kendaraan, sehingga harus dilakukan iterasi kembali untuk mendapatkan setiap karakter pada baris 4. Pada baris 5, variabel *BLACK* ditetapkan dengan nilai RGB dari warna hitam sebagai warna *padding*. Baris 6 berfungsi untuk mendapatkan ukuran *height* dan *width* pada citra asal dan disimpan berturut-turut pada variabel *h* dan *w*. Baris 7 hingga 12 merupakan proses penetapan ukuran *height* dan *width* tambahan yang diperlukan untuk proses *padding*.

Karena rasio yang ingin dicapai antara *width* dan *height* adalah 1:1, maka apabila *height* lebih panjang daripada *width*, maka panjang citra akan ditambah *padding* sebanyak 5% pada bagian atas (*top*) dan bawah (*bottom*), sedangkan bagian kanan (*right*) dan kiri (*left*) akan ditambah sebanyak 110% ukuran *height* dikurangi ukuran *width* kemudian dibagi 2 agar seimbang. Apabila *width* yang berukuran lebih panjang daripada *height*, maka dilakukan sebaliknya. Penambahan *padding* sendiri dilakukan dengan bantuan fungsi *copyMakeBorder* dari *library* OpenCV pada baris 14 dengan parameter sebagai berikut:

1. Parameter *img* adalah citra awal yang akan diproses.
2. Parameter *top, top* berturut-turut adalah dalah lebar *border* dalam jumlah piksel dalam arah atas dan lebar *border* dalam jumlah piksel ke arah bawah.
3. Parameter *left, left* berturut-turut adalah dalah lebar *border* dalam jumlah piksel dalam arah kiri dan lebar *border* dalam jumlah piksel ke arah kanan.
4. Parameter *cv2.BORDER_CONSTANT* adalah jenis *border* dari *library* OpenCV yang akan ditambahkan.
5. Parameter *value* adalah parameter opsional yang menggambarkan warna *border* apabila jenis *border* yang digunakan adalah *cv2.BORDER_CONSTANT*.

Setelah pemberian *padding*, citra kemudian di-*resize* dengan bantuan fungsi *imresize* dari *library* SciPy pada baris 16. Parameter yang digunakan yaitu *constant* sebagai citra yang akan di-*resize*

dan (*size*, *size*) sebagai ukuran yang diinginkan. Setelah itu, citra akhir ditampung dalam variabel *pc* yang berisi karakter-karakter pada baris 17. Kemudian, variabel *pc* akan disimpan pada variabel *post_char* pada baris 18, sehingga variabel *post_char* akan berisi kumpulan *array* yang berisi kumpulan karakter.

Sebelum melakukan pengujian, model hasil pembangunan pada tahap sebelumnya dimuat terlebih dahulu dengan fungsi *load_model* yang dapat dilihat pada Kode Sumber 4.4.8. Parameter yang digunakan yaitu nama *file* model yang telah disimpan sebelumnya.

```
1.  model = load_model('model.h5')
```

Kode Sumber 4.4.8 Implementasi pemuatan model RNN

Proses pengujian kemudian dilakukan untuk setiap karakter yang dapat dilihat pada Kode Sumber 4.4.9. Pada baris 1, dilakukan iterasi pada variabel *post_char* yang sebelumnya menyimpan kumpulan *array* yang berisi kumpulan karakter. Kemudian, pada baris 2 hingga 3, tipe data diubah menjadi *float32* supaya dapat diprediksi menggunakan fungsi *predict_classes* pada baris 4. Hasil prediksi kemudian diiterasi dan dipetakan menjadi kelas yang sebenarnya pada baris 5 hingga 6.

```
1.  for num, i in enumerate(post_char):
2.      test_real_image = np.array(i)
3.      test_real_image =
    test_real_image.astype('float32')
4.      y_pred = model.predict_classes(test_real_image)
5.      for num2, ch in enumerate(y_pred):
6.          print(char[ch],end=' ')
```

Kode Sumber 4.4.9 Pengenalan karakter dengan RNN

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, Scikit-learn, Scikit-image, dan SciPy.

5.2 Deskripsi Dataset

Pada tugas akhir ini, data yang digunakan sebagai masukan awal dari sistem deteksi posisi dan pengenalan plat nomor kendaraan dibagi menjadi dua, yaitu data untuk deteksi posisi plat nomor kendaraan menggunakan *Single Shot Detector* (SSD) dan data untuk pengenalan karakter pada plat nomor kendaraan menggunakan *Recurrent Neural Network* (RNN).

Data untuk deteksi posisi plat nomor kendaraan adalah data yang diambil secara mandiri untuk data latih dan data validasi. Data yang diambil terdiri dari 1 kelas dimana di masing-masing gambar terdapat minimal 1 plat nomor kendaraan beserta anotasi posisi plat. Anotasi-anotasi ini merupakan hasil pencocokan yang dilakukan secara mandiri. Untuk spesifikasi lengkap dataset skenario uji coba SSD dapat dilihat pada Tabel 5.1.

Tabel 5.1 Spesifikasi data latih uji coba SSD

Keterangan	**Spesifikasi**
Ukuran resolusi asli	1280 x 720
Ekstensi	.jpg
Jumlah gambar	280
Jumlah kelas	1 kelas
Ukuran file	200 - 300 kB
Kanal warna	3 (RGB)

Untuk pengenalan karakter, dataset yang digunakan adalah dataset yang terdiri dari 36 kelas di mana pada masing-masing kelas terdapat sekitar 6000 sampai dengan 8000 citra. Data ini merupakan sekumpulan citra karakter A sampai dengan Z serta angka 0 sampai dengan 9. Pembagian data latih dan data validasi menjadi 187,914 data validasi dan 91,333 data latih. Untuk spesifikasi lengkap dataset skenario uji coba RNN dapat dilihat pada Tabel 5.2.

Tabel 5.2 Spesifikasi data latih uji coba RNN

Keterangan	**Spesifikasi**
Ukuran resolusi	32 x 32
Ekstensi	.jpg
Jumlah gambar	279,247
Jumlah kelas	36 kelas
Jumlah gambar per kelas	6000 – 8000
Ukuran file	1 kB
Kanal warna	1 (*binary*)

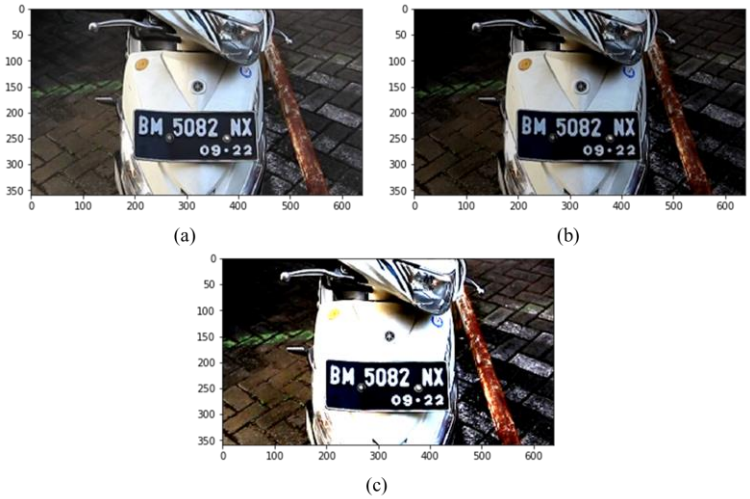
5.3 Hasil Praproses

Sebelum memasuki proses lokalisasi, segmentasi, dan pengenalan karakter yang telah dirancang sebelumnya, akan dilakukan praproses citra. Terdapat perbedaan antara praproses citra yang dilakukan pada setiap tahapan. Perbedaan ini terletak pada tujuan praproses citra itu sendiri. Pada tahap lokalisasi, tujuan praproses citra adalah untuk menambah variasi data latih pada

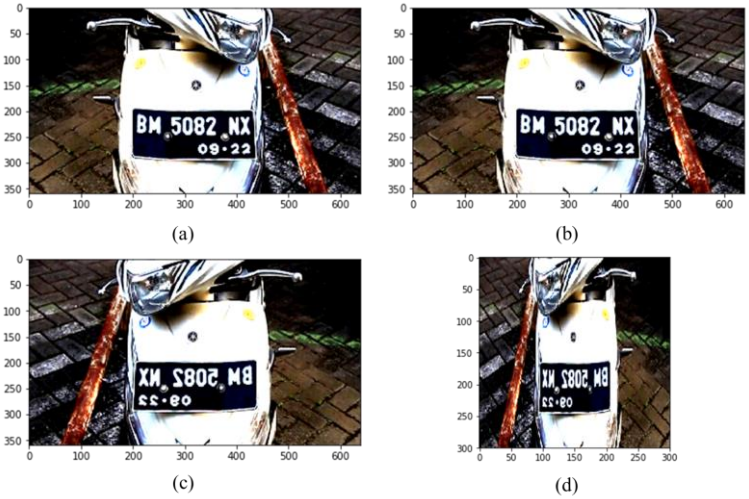
network *Single Shot Detector* (SSD). Pada tahap segmentasi, praproses citra bertujuan untuk meningkatkan ketepatan segmentasi. Sedangkan pada tahap pengenalan karakter, praproses citra bertujuan untuk menyamaratakan data uji yang merupakan hasil keluaran dari tahap segmentasi.

5.3.1 Praproses Data Latih Tahap Lokalisasi

Pada tahap praproses untuk pelatihan lokalisasi, citra awalnya akan dilakukan proses *brightness transform* untuk memodifikasi tingkat keterangan cahaya. Setelah dilakukan *brightness transform*, citra kemudian dilakukan *contrast transform* untuk mengubah kontras. Kemudian, citra dilakukan *hue transform* dan *saturation transform* untuk mengubah warna dan saturasi. Citra kemudian dilakukan *horizontal flip* agar semakin bervariasi. Setelah itu, maka citra akan di-*resize* menjadi ukuran 300x300, menyesuaikan dengan *input* SSD. Citra hasil dapat dilihat pada Gambar 5.1 dan Gambar 5.2.

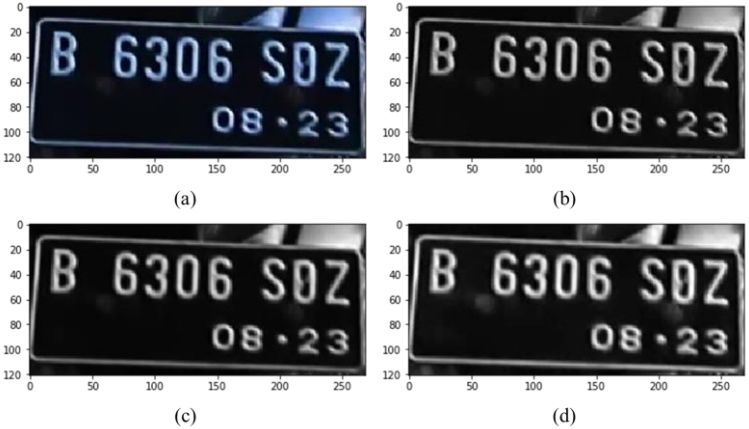


Gambar 5.1 (a) Citra asli; (b) Hasil *brigthness transform*; (c) Hasil *contrast transform.*



Gambar 5.2 (a) Hasil *hue transform*; (b) Hasil *saturation transform*; (c) Hasil *horizontal flip*; (d) Hasil *resize.*

5.3.2 Praproses Data Tahap Segmentasi

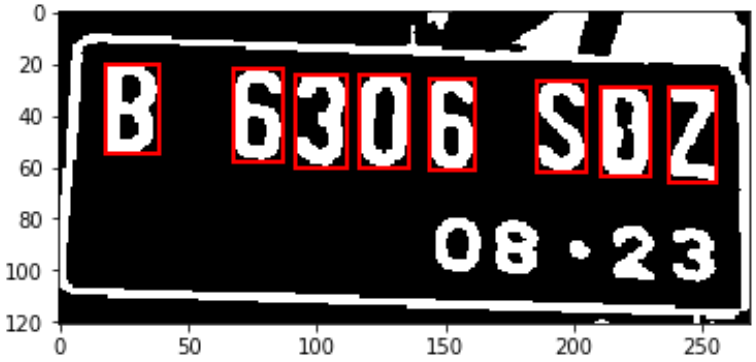


Gambar 5.3 (a) Citra asli; (b) Hasil pembacaan *grayscale*; (c) Hasil *Gaussian Blur*; (d) Hasil CLAHE.

Pada tahap praproses untuk segmentasi, citra awalnya akan dilakukan proses pembacaan *grayscale* untuk mengubah kanal citra yang awalnya BGR (citra 3 kanal) menjadi *grayscale* (citra 1 kanal). Setelah dilakukan pembacaan *grayscale*, citra kemudian dilakukan pemberian *Gaussian Blur* untuk menghilangkan *noise* yang dapat mempengaruhi proses segmentasi.

(a) (b)

Gambar 5.4 (a) Hasil citra biner dengan praproses; (b) Hasil citra biner tanpa praproses.



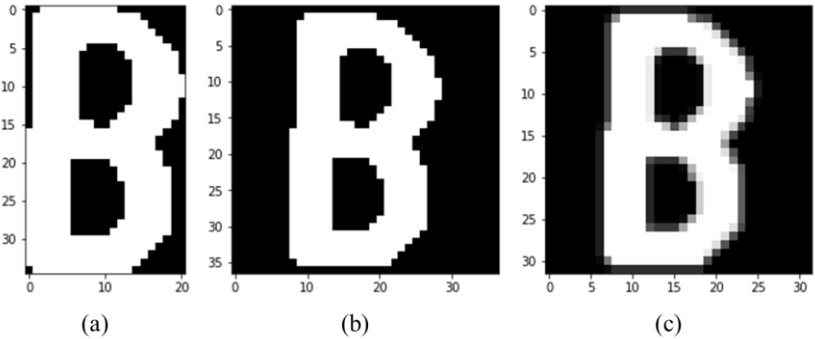
Gambar 5.5 Hasil proses segmentasi

Kemudian, citra diaplikasikan dengan CLAHE untuk mendapatkan kontras terbaik. Citra kemudian dilakukan *Otsu thresholding* mencari nilai terbaik untuk proses konversi biner. Kemudian citra *grayscale* ini nantinya akan dikonversi menjadi citra biner berdasarkan hasil dari *Otsu thresholding*. Citra hasil dan perbandingannya dapat dilihat pada Gambar 5.3 dan Gambar 5.4.

Untuk proses segmentasi, hasil karakter yang berhasil terdeteksi dapat dilihat pada Gambar 5.5.

5.3.3 Praproses Data Uji Tahap Pengenalan Karakter

Pada tahap praproses untuk pengujian pengenalan karakter, citra akan akan diterapkan proses *padding* untuk membuat rasio citra yang sebelumnya variatif menjadi 1:1. Setelah dilakukan *padding* maka citra akan di-*resize* menjadi ukuran 32x32, menyesuaikan dengan *input Recurrent Neural Network*. Citra hasil dapat dilihat pada Gambar 5.6.



Gambar 5.6 (a) Citra asli; (b) Hasil *padding*; (c) Hasil *resize*.

5.4 Skenario Uji Coba

Proses uji coba berguna untuk menemukan parameter-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. Pada subbab ini, skenario uji coba dibagi menjadi 2 dan 1 hasil uji coba yaitu:

1. Skenario Uji Coba pada *Single Shot Detector* (SSD)
2. Skenario Uji Coba pada *Recurrent Neural Network* (RNN)
3. Hasil Uji Coba pada Data Video

5.4.1 Skenario Uji Coba pada Single Shot Detector

Pada SSD, terdapat 2 macam skenario uji coba dan semuanya akan dicoba pada arsitektur SSD yang telah dirancang. Skenario uji coba yang akan dilakukan yaitu:

1. Uji Coba Parameter SSD
2. Uji Coba Augmentasi Anotasi pada Data Latih

Tabel 5.3 Spesifikasi parameter awal arsitektur SSD

Keterangan	**Parameter**
Jumlah *epoch*	50
Ukuran *batch*	8
Learning Rate	0.0001
Preset	VGG300

Setiap skenario nantinya akan diuji pada data uji yang terdiri dari kondisi siang, sore, dan malam. Tabel 5.3 berisi parameter-parameter awal arsitektur SSD 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.

5.4.1.1 Uji Coba Parameter Single Shot Detector

Uji coba penggantian parameter SSD digunakan untuk mengetahui parameter mana saja yang menghasilkan performa model terbaik. Parameter SSD yang akan dicoba untuk divariasikan antara lain:

1. Preset

Uji coba pertama adalah uji coba *preset* pada model yang akan dibuat. *Preset* yang akan digunakan pada uji coba ini antara lain *preset* VGG300 dan VGG512. Percobaan dilakukan pada *learning rate* 0.0001.

2. Learning rate

Uji coba *learning rate* divariasikan dengan harapan mendapatkan performa model yang lebih baik dari sebelumnya. Akan dicoba beberapa variasi *learning rate* yaitu 0.00075, 0.0001, 0.000075, dan 0.00001. Variasi *learning rate* tersebut ditentukan karena *learning rate* yang bernilai lebih besar dari sama dengan 0.001 akan menghasilkan nilai *confidence loss* bernilai NaN.

Uji coba penggantian *preset* pada arsitektur SSD menghasilkan perbandingan nilai akurasi pengujian serta lama waktu pelatihan yang dapat dilihat pada Tabel 5.4.

Tabel 5.4 Hasil uji coba penggantian *preset*

Preset	**Lama waktu pelatihan**	**Akurasi**
VGG300	2137 detik	**93.40%**
VGG512	1885 detik	90.39%

Pada pengujian *preset*, didapatkan *preset* VGG300 merupakan *preset* yang lebih optimal. Maka selanjutnya, pengujian *learning rate* dilakukan dengan menggunakan *preset* VGG300.

Uji coba penggantian *learning rate* pada arsitektur SSD menghasilkan perbandingan nilai akurasi dan waktu pelatihan masing-masing model dapat dilihat pada Tabel 5.5.

Tabel 5.5 Hasil uji coba penggantian *learning rate*

Learning Rate	**Lama waktu pelatihan**	**Akurasi**
0.00075	2112 detik	85.51%
0.0001	2137 detik	**93.40%**
0.000075	2157 detik	92.98%
0.00001	2176 detik	9.77%

Pada hasil uji coba, didapatkan bahwa nilai *learning rate* 0.0001 menghasilkan akurasi tertinggi dibandingkan percobaan yang lainnya.

5.4.1.2 Uji Coba Augmentasi Anotasi pada Data Latih

Uji coba augmentasi anotasi pada data latih SSD digunakan untuk mengetahui anotasi mana yang paling optimal menghasilkan performa model terbaik. Uji coba anotasi yang akan dicoba untuk divariasikan antara lain:

1. Anotasi asli

Anotasi awal yang telah dilakukan dan tidak diberi perubahan apa-apa.

2. Anotasi lebih lebar

Anotasi asli yang diperbesar ukurannya sebanyak 5% dari luas aslinya.

3. Anotasi lebih sempit

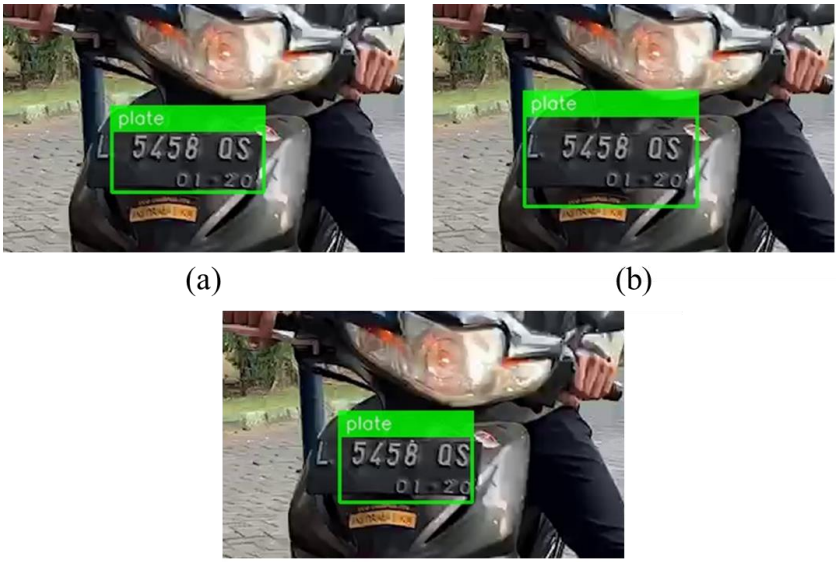
Anotasi asli yang diperkecil ukurannya sebanyak 5% dari luas aslinya.

Uji coba penggantian anotasi pada data latih SSD menghasilkan perbandingan nilai akurasi pengujian serta lama waktu pelatihan yang dapat dilihat pada Tabel 5.6.

Tabel 5.6 Hasil uji coba penggantian anotasi

Anotasi	**Lama waktu pelatihan**	**Akurasi**
Anotasi lebar	1567 detik	**96.99%**
Anotasi asli	2157 detik	93.40%
Anotasi sempit	2727 detik	89.97%

Pada pengujian anotasi, didapatkan bahwa anotasi lebar menghasilkan akurasi tertinggi dibandingkan percobaan yang lainnya. Pada Gambar 5.7, hasil dengan anotasi lebar lebih meng-*cover* bagian plat dibandingkan percobaan yang lain. Oleh karena itu, didapatkan anotasi lebar merupakan anotasi yang paling optimal.



(a) (b)

(c)

Gambar 5.7 (a) Hasil deteksi dengan anotasi asli; (b) Hasil deteksi dengan anotasi lebar; (c) Hasil deteksi dengan anotasi sempit.

5.4.2 Skenario Uji Coba pada Recurrent Neural Network (RNN)

Pada *Recurrent Neural Network* (RNN), terdapat 2 macam skenario uji coba dan semuanya akan dicoba pada arsitektur RNN yang telah dirancang.

Tabel 5.7 Spesifikasi awal parameter arsitektur RNN

Keterangan	**Parameter**
Jumlah *epoch*	100
Ukuran *batch*	128
Learning Rate	0.0001
Optimizer	RMSProp

Skenario uji coba yang akan dilakukan yaitu:

1. Uji Coba Parameter RNN
2. Uji Coba K-Folds *Cross Validation*

Setiap skenario nantinya akan diuji pada data uji yang terdiri dari kondisi siang, sore, dan malam. Tabel 5.7 berisi parameter-parameter awal arsitektur RNN 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.

5.4.2.1 Uji Coba Parameter Recurrant Neural Network

Uji coba penggantian parameter RNN digunakan untuk mengetahui parameter mana saja yang menghasilkan performa model terbaik. Parameter RNN yang akan dicoba untuk divariasikan antara lain:

1. Optimizer

Uji coba pertama adalah uji coba *optimizer* pada model yang akan dibuat. *Optimizer* yang akan digunakan pada uji coba ini antara lain SGD, Adagrad, RMSProp, dan Adam. Percobaan dilakukan pada *learning rate* 0.0001.

2. Learning rate

Uji coba *learning rate* divariasikan dengan harapan mendapatkan performa model yang lebih baik dari sebelumnya. Akan dicoba beberapa variasi *learning rate* yaitu 0.01, 0.001, dan 0.0001.

Tabel 5.8 Hasil uji coba penggantian *preset*

Optimizer	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***
SGD	4250 detik	19.12%	17.13%	19.12%
Adagrad	4186 detik	97.61%	97.62%	97.61%
RMSProp	4450 detik	99.16%	99.17%	99.16%
Adam	4169 detik	**99.57%**	**99.57%**	**99.57%**

Uji coba penggantian *optimizer* pada arsitektur RNN menghasilkan perbandingan nilai akurasi, *precision*, *recall* pengujian serta lama waktu pelatihan yang dapat dilihat pada Tabel 5.8.

Pada pengujian *optimizer*, didapatkan *optimizer* Adam merupakan *optimizer* paling optimal. Maka, untuk pengujian *learning rate* dilakukan dengan menggunakan *optimizer* Adam.

Uji coba penggantian *learning rate* pada arsitektur RNN menghasilkan perbandingan akurasi, *precision, recall,* dan waktu pelatihan masing-masing model dapat dilihat pada Tabel 5.9.

Tabel 5.9 Hasil uji coba penggantian *learning rate*

Learning Rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***
0.01	4142 detik	2.87%	3.00%	2.87%
0.001	4391 detik	93.56%	93.86%	93.56%
0.0001	4169 detik	**99.57%**	**99.57%**	**99.57%**

Pada hasil uji coba, didapatkan bahwa nilai *learning rate* 0.0001 menghasilkan akurasi yang tertinggi dibandingkan percobaan yang lainnya.

5.4.2.2 Uji Coba K-Folds Cross Validation

Tabel 5.10 Hasil uji coba K-Folds *Cross Validation*

Nilai k	**Akurasi**
3	99.61%
6	99.64%
9	**99.69%**
12	99.68%

Pada uji coba sebelumnya, didapatkan model terbaik berasal dari model yang menggunakan *optimizer* Adam dengan *learning rate* 0.0001. Maka, pada percobaan K-Folds *Cross Validation*, akan dilakukan dengan menggunakan arsitektur optimal yang menggunakan *optimizer* Adam dan nilai *learning rate* 0.0001.

Uji coba dilakukan dengan menggunakan *k* bernilai 3, 6, 9, dan 12. Nilai *k* akan membuat data dibagi sebanyak *k batch* secara rata. Hasil uji coba dapat dilihat pada Tabel 5.10. Pada hasil uji coba, didapatkan akurasi terbaik dengan nilai *k* berjumlah 9 dengan akurasi sebesar 99.69%.

5.4.3 Hasil Uji Coba pada Data Video

Uji coba pada data video dilakukan pada 3 kondisi data video yang berasal dari kamera CCTV untuk mengukur performa model yang telah dibuat. Data video yang digunakan merupakan data video kendaraan motor yang akan dikenali karakter-karakter dari plat nomornya. Ketiga kondisi data video terdiri dari kondisi siang, kondisi sore, dan kondisi malam.

Setiap kondisi terdiri dari masing-masing 10 data video. Untuk melakukan evaluasi performa, akan dilakukan pada hasil lokalisasi, segmentasi, pengenalan karakter, dan *majority vote*.

Untuk lokalisasi, hasil deteksi dinilai benar 100% apabila area yang terdeteksi mencakup seluruh karakter plat nomor kendaraan, sedangkan apabila hanya beberapa karakter yang terdeteksi, maka akurasi dinilai sebanyak jumlah yang terdeteksi dibagi dengan total karakter sebenarnya dalam skala 100%.

(a) (b)

Gambar 5.8 (a) Contoh hasil plat dengan cakupan 100%; (b) Contoh hasil plat dengan cakupan kurang dari 100%.

Untuk segmentasi, hasil deteksi dinilai benar apabila *region* yang terdeteksi merupakan *region* karakter. Kemudian, akurasi pengenalan karakter diuji pada setiap hasil segmentasi yang benar

untuk setiap *frame* dan untuk setiap videonya menggunakan *majority vote*.

5.4.3.1 Uji Coba Kondisi Siang

Uji coba dilakukan terhadap 10 data video yang nantinya akan dijadikan *frames* dengan jumlah *frame* tiap video dapat dilihat pada Tabel 5.11. Contoh hasil *frame* dari video dapat dilihat pada Gambar 5.9. Hasil uji coba keseluruhan untuk data video pada kondisi siang hari dapat dilihat pada Tabel 5.12.

Gambar 5.9 Hasil *frame* video siang

Tabel 5.11 Spesifikasi *frame* video siang

Video	Plat	Jumlah *Frame*
1	B 3578 TLW	20 *frame*
2	AG 3192 RAL	20 *frame*
3	AG 2351 FL	20 *frame*
4	L 2638 UI	20 *frame*
5	L 3673 DX	20 *frame*
6	L 6382 ML	20 *frame*
7	W 6770 WK	20 *frame*
8	L 4025 HL	20 *frame*

Video	Plat	Jumlah *Frame*
9	AE 3395 WR	20 *frame*
10	W 3440 WJ	20 *frame*

Akurasi lokalisasi, segmentasi, dan akurasi pengenalan karakter didapatkan dengan cara membandingkan secara visual label prediksi dengan citra itu sendiri, benar atau salah. Metode ini dilakukan karena setiap citra keluaran yang didapatkan tidak mempunyai *ground truth* untuk membandingkan apakah lokalisasi, segmentasi, atau pun prediksi karakter yang didapatkan sudah benar atau tidak.

Tabel 5.12 Hasil pengujian kondisi siang

Video	Lokalisasi	Segmentasi	*Majority Vote*	Pengenalan Karakter
1	**100.00%**	77.78%	B 3578 TLV	**100.00%**
2	83.89%	85.16%	AG 3192 6RA	96.30%
3	95.00%	20.53%	1 F	99.12%
4	**100.00%**	92.64%	**L 2638 UI**	99.29%
5	**100.00%**	86.43%	L 3673 X	95.48%
6	94.29%	**98.03%**	**L 6382 ML**	97.74%
7	**100.00%**	86.07%	**W 6770 WK**	96.86%
8	47.86%	91.67%	L 4025 H	92.86%
9	**100.00%**	91.11%	**AE 3395 WR**	**100.00%**
10	**100.00%**	92.32%	**W 3440 WJ**	**100.00%**

Didapatkan hasil evaluasi akurasi lokalisasi terbaik pada video 4, 5, 7, 9, dan 10 dengan akurasi sebesar 100.00%, hasil evaluasi akurasi segmentasi terbaik terdapat pada video 6 dimana didapatkan akurasi segmentasi video sebesar 98.03%, dan hasil evaluasi akurasi pengenalan karakter terbaik pada video 1, 9, dan 10 dengan ketepatan sebesar 100%.

5.4.3.2 Uji Coba Kondisi Sore

Uji coba dilakukan terhadap 10 data video yang nantinya akan dijadikan *frames* dengan jumlah *frame* tiap video dapat dilihat

pada Tabel 5.13. Contoh hasil *frame* dari video dapat dilihat pada Gambar 5.10. Hasil uji coba keseluruhan untuk data video pada kondisi siang hari dapat dilihat pada Tabel 5.14.

Tabel 5.13 Spesifikasi *frame* video sore

Video	Plat	Jumlah *Frame*
1	AG 3192 RAL	20 *frame*
2	N 5694 HZ	15 *frame*
3	L 3669 TT	19 *frame*
4	W 3440 WJ	20 *frame*
5	L 3673 DX	20 *frame*
6	AG 2351 FL	20 *frame*
7	H 6042 SV	20 *frame*
8	P 2083 WN	20 *frame*
9	M 2754 HK	20 *frame*
10	P 5129 KU	20 *frame*

Gambar 5.10 Hasil *frame* video sore

Akurasi lokalisasi, segmentasi, dan akurasi pengenalan karakter didapatkan dengan cara membandingkan secara visual label prediksi dengan citra itu sendiri, benar atau salah. Metode ini dilakukan karena setiap citra keluaran yang didapatkan tidak

mempunyai *ground truth* untuk membandingkan apakah lokalisasi, segmentasi, atau pun prediksi karakter yang didapatkan sudah benar atau tidak.

Didapatkan hasil evaluasi akurasi lokalisasi terbaik pada video 2, 3, 4, dan 9 dengan akurasi sebesar 100.00%, hasil evaluasi akurasi segmentasi terbaik terdapat pada video 8 dimana didapatkan akurasi segmentasi video sebesar 100.00%, dan hasil evaluasi akurasi pengenalan karakter terbaik pada video 3, 4, dan 9 dengan ketepatan sebesar 100%.

Tabel 5.14 Hasil pengujian kondisi sore

Video	Lokalisasi	Segmentasi	*Majority Vote*	Pengenalan Karakter
1	97.22%	54.67%	AG 3192 RA	91.46%
2	**100.00%**	77.63%	NN 5694 HZ	93.49%
3	**100.00%**	79.68%	**L 3669 TT**	**100.00%**
4	**100.00%**	99.29%	**W 3440 WJ**	**100.00%**
5	97.86%	76.53%	L 3673 X	94.00%
6	95.00%	90.44%	AG 2351 F1	94.00%
7	68.57%	78.56%	H 6042 S	94.32%
8	92.14%	**100.00%**	**P 2083 WN**	92.14%
9	**100.00%**	94.51%	**M 2754 HK**	**100.00%**
10	89.29%	89.46%	**P 5129 KU**	85.56%

5.4.3.3 Uji Coba Kondisi Malam

Uji coba dilakukan terhadap 10 data video yang nantinya akan dijadikan *frames* dengan jumlah *frame* tiap video dapat dilihat pada Tabel 5.15. Contoh hasil *frame* dari video dapat dilihat pada Gambar 5.11. Hasil uji coba keseluruhan untuk data video pada kondisi siang hari dapat dilihat pada Tabel 5.16.

Tabel 5.15 Spesifikasi *frame* video malam

Video	Plat	Jumlah *Frame*
1	W 3066 PA	20 *frame*
2	AG 2351 FL	20 *frame*

Video	Plat	Jumlah *Frame*
3	L 3673 DX	20 *frame*
4	N 3526 NBB	20 *frame*
5	H 6042 SV	20 *frame*
6	AG 3192 RAL	20 *frame*
7	AA 4427 MG	20 *frame*
8	W 3440 WJ	13 *frame*
9	S 3271 DU	20 *frame*
10	L 3669 TT	20 *frame*

Gambar 5.11 Hasil *frame* video malam

Tabel 5.16 Hasil pengujian kondisi malam

Video	Lokalisasi	Segmentasi	*Majority Vote*	Pengenalan Karakter
1	58.57%	28.44%	6	77.38%
2	**100.00%**	88.75%	**AG 2351 FL**	95.83%
3	95.00%	**100.00%**	**L 3673 DX**	92.48%
4	**100.00%**	99.38%	**N 3526 NBB**	98.13%
5	85.71%	95.89%	0 6042 S	83.33%
6	70.00%	99.38%	WAG 3192 WKABL	93.77%
7	92.76%	81.73%	A 4427 M	92.54%

Video	Lokalisasi	Segmentasi	*Majority Vote*	Pengenalan Karakter
8	81.32%	28.57%	W	86.82%
9	**100.00%**	94.91%	**S 3271 DU**	97.14%
10	**100.00%**	85.88%	**L 3669 TT**	**98.29%**

Akurasi lokalisasi, segmentasi, dan akurasi pengenalan karakter didapatkan dengan cara membandingkan secara visual label prediksi dengan citra itu sendiri, benar atau salah. Metode ini dilakukan karena setiap citra keluaran yang didapatkan tidak mempunyai *ground truth* untuk membandingkan apakah lokalisasi, segmentasi, atau pun prediksi karakter yang didapatkan sudah benar atau tidak.

Didapatkan hasil evaluasi akurasi lokalisasi terbaik pada video 2, 4, 9, dan 10 dengan akurasi sebesar 100.00%, hasil evaluasi akurasi segmentasi terbaik terdapat pada video 8 dimana didapatkan akurasi segmentasi video sebesar 100.00%, dan hasil evaluasi akurasi pengenalan karakter terbaik pada video 10 dengan ketepatan sebesar 98.29%.

5.5 Hasil dan Evaluasi

Pada uji coba penggantian parameter SSD, diperoleh hasil akurasi yang paling baik pada penggunaan *preset* VGG300 dengan akurasi sebesar 93.40%. *Preset* VGG300 tepat digunakan untuk arsitektur SSD yang telah dibangun karena menggunakan *input size* berukuran 300x300 yang cenderung lebih cepat konvergen serta menghasilkan performa yang lebih baik.

Selanjutnya, dilakukan uji coba pada nilai *learning rate* pada arsitektur SSD. Didapatkan hasil akurasi terbesar dengan mengubah nilai *learning rate* menjadi 0.0001. Akurasi model yang didapatkan sebesar 93.40%.

Pada uji coba augmentasi anotasi pada data latih untuk arsitektur SSD, diperoleh hasil akurasi terbaik sebesar 96.99% dengan menggunakan anotasi yang diperlebar. Anotasi yang diperlebar tepat digunakan untuk arsitektur SSD karena pada saat

pengujian, hasil deteksi cenderung lebih sempit dari apa yang telah ditetapkan pada saat pelatihan.

Pada uji coba penggantian parameter RNN, diperoleh hasil akurasi yang paling baik pada penggunaan Adam *optimizer* dengan akurasi sebesar 99.57%. Adam *optimizer* tepat digunakan untuk arsitektur RNN yang telah dibangun karena menggunakan *adaptive learning rate* yang cenderung lebih cepat konvergen serta menghasilkan performa yang lebih baik dibandingkan dengan *optimizer* yang menggunakan *learning rate* statis.

Selanjutnya, dilakukan uji coba pada nilai *learning rate* pada arsitektur RNN. Didapatkan hasil akurasi terbesar dengan mengubah nilai *learning rate* menjadi 0.0001. Akurasi model yang didapatkan sebesar 99.57%.

Pada uji coba K-Folds *Cross Validation*, akurasi yang hampir sama didapatkan untuk nilai *k* yang digunakan, yaitu 99.61% untuk nilai *k* 3, 99.64% untuk nilai *k* 6, 99.69% untuk nilai *k* 9, dan 99.68% untuk nilai *k* 12. Variasi nilai *k* yang digunakan tidak memperlihatkan akurasi yang jauh berbeda. Akan tetapi, dipilih *k* dengan nilai 9 karena dari segi akurasi yang lebih besar, nilai *k* lebih besar juga dapat berarti bias yang lebih kecil terhadap kelebihan estimasi terhadap *true expected error*. Meskipun demikian, dengan nilai *k* yang lebih besar, maka waktu komputasi dapat dikatakan lebih lama pula, karena jumlah data latih akan lebih banyak pada tiap *batch*-nya dibandingkan dengan jumlah data latih pada *batch* data yang menggunakan nilai *k* yang lebih kecil. Dalam kasus ini, *k* bernilai 12 memiliki data latih kurang lebih 255,976 data latih, sedangkan pada *k* bernilai 3 yang merupakan uji coba dengan nilai *k* terkecil, data latih pada *batch* ini sebesar kurang lebih 186,164 data.

Pada uji coba lokalisasi pada data video, didapatkan rata-rata akurasi lokalisasi sebesar 92.10% berasal dari video kondisi siang, 94.01% berasal dari video kondisi sore, dan 88.34% pada kondisi malam. Video pada kondisi malam mendapatkan rata-rata akurasi lokalisasi yang cukup berbeda dibandingkan dengan kondisi lainnya karena pada kondisi malam hari, kendaraan yang terekam

CCTV dengan keadaan lampu kendaraan menyala menyebabkan kualitas beberapa plat nomor kendaraan menjadi agak kabur seperti yang dapat dilihat pada Gambar 5.12 dibandingkan ketika lampu kendaraan tidak menyala pada Gambar 5.13.

Gambar 5.12 Contoh gambar dengan keadaan lampu kendaraan menyala

Gambar 5.13 Contoh gambar dengan keadaan lampu kendaraan tidak menyala

Pada data video secara keseluruhan, keadaan lain yang dapat menyebabkan plat kendaraan tidak terdeteksi atau kurang terdeteksi secara sempurna yaitu jarak plat kendaraan yang terlalu jauh dari kamera pada Gambar 5.14. Selain itu, beberapa video dengan kondisi cahaya *backlight* dan kontras CCTV yang dapat berubah-ubah dapat mempengaruhi hasil lokalisasi karena kurang teraturnya kontras seperti pada Gambar 5.15.

Gambar 5.14 Contoh gambar dengan plat kendaraan terlalu jauh dan silau

Pada uji coba segmentasi pada data video, didapatkan rata-rata akurasi segmentasi sebesar 81.88% berasal dari video kondisi siang, 84.39% berasal dari video kondisi sore, dan 81.79% pada video kondisi malam. Akurasi segmentasi terbaik berasal dari video 8 dari kondisi sore dan video 3 dari kondisi malam dengan akurasi sebesar 100.00% sedangkan akurasi di bawah 30.00% didapatkan dari berturut-turut video 3 pada kondisi siang, video 1 pada kondisi malam, dan video 8 pada kondisi malam. Video 8 dari kondisi sore dan video 3 dari kondisi malam pada Gambar 5.16 dan Gambar 5.17 mendapatkan akurasi yang tinggi dikarenakan oleh kondisi plat kendaraan yang sudah baik secara visual dan kontras

yang teratur, sehingga pada saat pemrosesan lebih sedikit noise yang diatasi dan citra tersegmentasi dengan baik.

Gambar 5.15 Contoh gambar dengan kualitas kontras yang kurang

(a) (b)

Gambar 5.16 Contoh gambar dari video 8 kondisi sore. (a) Gambar asli; (b) Gambar hasil segmentasi.

Sedangkan untuk video 3 pada kondisi siang, video 1 pada kondisi malam, dan video 8 pada kondisi malam pada Gambar 5.18, Gambar 5.19, dan Gambar 5.20 mendapatkan akurasi yang rendah dikarenakan oleh secara visual, karakter pada area plat tidak terlalu terlihat. Gangguan visual yang dimaksud dapat disebabkan oleh kontras awal area terlokalisasi yang terlalu rendah seperti pada Gambar 5.18 dan citra yang kabur akibat lampu kendaraan menyala pada malam hari seperti pada Gambar 5.20. Terdapat juga bagian-

bagian yang menyatu seperti pada Gambar 5.19 dan Gambar 5.20, meski pada implementasi telah dilakukan beberapa praproses, akan tetapi, detail citra secara jelas tetap tidak terlihat sehingga hasil prediksi bisa menghasilkan *False Positive* dan *False Negative* yang lebih banyak dibandingkan *True Positive*.

(a) (b)

Gambar 5.17 Contoh gambar dari video 3 kondisi malam. (a) Gambar asli; (b) Gambar hasil segmentasi.

(a) (b)

Gambar 5.18 Contoh gambar dari video 3 kondisi siang. (a) Gambar asli; (b) Gambar hasil segmentasi.

(a) (b)

Gambar 5.19 Contoh gambar dari video 1 kondisi malam. (a) Gambar asli; (b) Gambar hasil segmentasi.

(a) (b)

Gambar 5.20 Contoh gambar dari video 8 kondisi malam. (a) Gambar asli; (b) Gambar hasil segmentasi.

Pada uji coba pengenalan karakter dan *majority vote* pada data video, didapatkan rata-rata akurasi *majority vote* dan pengenalan karakter secara berturut-turut sebesar 86.39% dan 98.04% berasal dari video kondisi siang, 93.53% dan 94.55% berasal dari video kondisi sore, dan 74.17% dan 92.25% pada video kondisi malam. Akurasi rata-rata terbaik berasal dari data video kondisi sore karena hasil *majority vote* dipengaruhi oleh hasil lokalisasi dan hasil pengenalan karakter dipengaruhi oleh hasil segmentasi, dimana rata-rata data video pada kondisi sore memiliki performa yang baik pada hasil lokalisasi dan segmentasinya dibandingkan kondisi lainnya. Dalam uji coba pengenalan karakter, terjadi beberapa kali misklasifikasi pada beberapa karakter seperti angka 6, huruf W, ataupun pada huruf L. Misklasifikasi pada angka 8 terjadi karena pada dataset, variasi karakter angka 6 ini mirip dengan angka 8 pada dataset, seperti terlihat pada Gambar 5.21.

(a) (b) (c)

Gambar 5.21 (a) Karakter 6 pada data uji; (b) Karakter 6 pada dataset; (c) Karakter 8 pada dataset.

Gambar 5.22 (a) Karakter W pada data uji; (b) Karakter W pada dataset; (c) Karakter V pada dataset; (d) Karakter 4 pada dataset.

Gambar 5.23 (a) Karakter L pada data uji; (b) Karakter L pada dataset; (c) Karakter 1 pada dataset.

Dapat dilihat, secara visual karakter 6 pada uji secara *font* lebih menyatu dibandingkan karakter 6 pada dataset dan membuat karakter 6 pada data uji secara visual lebih mirip dengan angka 8. Untuk misklasifikasi pada huruf W dan L terjadi karena beberapa hasil dari proses lokalisasi yang tidak sempurna, mengakibatkan pemotongan karakter yang tidak menyeluruh atau setengah bagian saja pada proses segmentasi seperti terlihat pada Gambar 5.22 dan Gambar 5.23. Karakter W dan L yang terpotong mengakibatkan hasil klasifikasi berturut-turut menjadi karakter V atau 4 dan karakter 1.

Dari seluruh hasil uji coba yang telah dilakukan, pada Tabel 5.17 ditetapkan parameter optimal untuk arsitektur jaringan *Single Shot Detector* (SSD) dan pada Tabel 5.18 ditetapkan parameter optimal untuk arsitektur *Recurrent Neural Network* (RNN) dari seluruh uji coba tersebut. Performa waktu yang dibutuhkan masing-masing proses untuk setiap *frame* dari data video rata-rata sebesar 30 ms untuk proses lokalisasi, 2 ms untuk proses segmentasi, dan 13 ms untuk pengenalan karakter. Sedangkan performa waktu untuk sistem secara keseluruhan, dibutuhkan waktu rata-rata sebesar 48 ms untuk setiap *frame* dari data video.

Tabel 5.17 Parameter SSD optimal yang ditetapkan

Keterangan	**Parameter Optimal**
Arsitektur SSD *preset*	*Preset* VGG300
Learning rate	0.0001
Anotasi data latih	Anotasi lebar

Tabel 5.18 Parameter RNN optimal yang ditetapkan

Keterangan	**Parameter Optimal**
Arsitektur RNN *Optimizer*	Adam *optimizer*
Learning rate	0.0001
K-Folds Cross Validation	$K = 9$

(Halaman ini sengaja dikosongkan)

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas tentang kesimpulan yang didasari oleh hasil uji coba yang telah dilakukan terhadap *Single Shot Detector* (SSD) dan *Recurrent Neural Network* (RNN)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. Berdasarkan uji coba parameter pada arsitektur SSD yang digunakan, model yang dibangun menghasilkan akurasi yang paling baik yaitu sebesar 93.40% untuk *preset* VGG300 dan *learning rate* 0.0001.
2. Berdasarkan uji coba augmentasi anotasi pada data latih untuk SSD, model yang dibangun menghasilkan *coverage* dan akurasi yang paling baik yaitu sebesar 96.99% untuk anotasi dengan ukuran yang diperlebar.
3. Berdasarkan uji coba parameter pada arsitektur RNN yang digunakan, model yang dibangun menghasilkan akurasi yang paling baik yaitu sebesar 99.57% untuk *optimizer* Adam dan *learning rate* 0.0001.
4. Berdasarkan uji coba K-Folds *Cross Validation* pada arsitektur RNN yang digunakan, model yang dibangun menghasilkan akurasi yang paling baik yaitu sebesar 99.69% untuk ukuran *k* sebesar 9.
5. Metode segmentasi karakter sudah cukup berhasil melakukan segmentasi pada area plat kendaraan yang terdeteksi, dengan akurasi rata-rata segmentasi mencapai 82.69%.

6. Sistem deteksi posisi dan pengenalan plat nomor kendaraan telah berhasil diimplementasikan dengan akurasi model SSD dan akurasi RNN tertinggi berturut-turut 96.99% dan 99.69%, serta akurasi pengujian SSD dan RNN pada *majority vote* dan pengenalan karakter rata-rata berturut-turut 91.48%, 84.70%, dan 94.94% yang didapatkan dari uji coba preset VGG300 dengan *learning rate* 0.0001 serta anotasi yang diperlebar pada SSD dan Adam *optimizer* dengan *learning rate* 0.0001 dan *k* sebesar 9 pada RNN.

6.2 Saran

Saran yang diberikan untuk pengembangan sistem deteksi posisi dan pengenalan plat nomor kendaraan menggunakan SSD dan RNN pada data video, yaitu:

1. Menambah variasi kondisi dan posisi plat kendaraan pada data latih dalam lokalisasi berdasarkan kondisi sesungguhnya untuk menangani kondisi-kondisi pada realita.
2. Pengembangan sistem yang dapat melakukan segmentasi karakter lebih akurat.
3. Menambah variasi *font* pada karakter untuk memperbanyak variasi karakter yang ada.
4. Melakukan eksplorasi parameter selain *optimizer* dan *learning rate* yang dapat menambah performa arsitektur seperti *activation function*, jumlah dan *stride filter* konvolusi, ukuran *max pooling*.

DAFTAR PUSTAKA

[1] E. Forson, "Understanding SSD MultiBox—Real-Time Object Detection In Deep Learning," [Online]. Available: https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab. [Diakses 12 June 2019].

[2] L. Zheng, X. He, B. Samali dan L. T. Yang, "An algorithm for accuracy enhancement," *Journal of Computer and System Sciences*, pp. 245-255, 2013.

[3] S. C. Pau, "SEGMENTATION-FREE LICENSE PLATE RECOGNITION USING DEEP LEARNING," *A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Science (Hons.) Software Engineering* , 2017.

[4] F. Oladeji, "Developing a License Plate Recognition System with Machine Learning in Python," [Online]. Available: https://blog.devcenter.co/developing-a-license-plate-recognition-system-with-machine-learning-in-python-787833569ccd. [Diakses 27 May 2019].

[5] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu dan A. C. Berg, "SSD: Single Shot Multibox Detector," vol. 5, 2016.

[6] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 9(1), pp. 62-66, 1979.

[7] "Binary Image Analysis," dalam *Computer Vision*, 2000, pp. 63-75.

[8] M. Venkatachalam, "Recurrent Neural Networks," [Online]. Available: https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce. [Diakses 13 June 2019].

[9] S. Fadillah, "Penerapan Pengolahan Citra menggunakan Metode Deep Learning untuk Mendeteksi Kecacatan Permukaan Buah Manggis," Yogyakarta, 2017.

[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 "An Intuitive Explanation of Convolutional Neural Networks," Ujjwalkarn, 11 August 2016. [Online]. Available: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets. [Diakses 29 November 2018].

[12 I. W. Suartika, A. Y. Wijaya dan R. Soelaiman, "Klasifikasi Citra Menggunakan Convolutional pada Caltech 101," *JURNAL TEKNIK ITS,* vol. 5, 2016.

[13 S. Sena, "Pengenalan Deep Learning Neural Network," 28 October 207. [Online]. Available: https://medium.com/@samuelsena/pengenalan-deep-learning-8fbb7d8028ac. [Diakses 30 November 2018].

[14 G. Hinton, *Neural Networks for Machine Learning.*

[15 S. Ruder, "Ruder.io," 19 January 2016. [Online]. Available: http://ruder.io/optimizing-gradient-descent/index.html#rmsprop. [Diakses 23 December 2018].

[16 A. Budhiraja, "Dropout in (Deep) Machine Learning," [Online]. Available: https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5. [Diakses 11 12 2018].

[17 A. Karpathy, “Convolutional Neural Networks for Visual Recognition,” Stanford University, [Online]. Available: http://cs231n.github.io/. [Diakses 30 November 2018].

[18 S. Narkhede, “Understanding Confusion Matrix,” [Online]. Available: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62. [Diakses 29 May 2019].

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

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

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

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

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

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

[25 “Matplotlib,” Matplotlib, [Online]. Available: https://matplotlib.org/index.html. [Diakses 30 November 2018].

[26 s.-i. d. team, scikit-image, [Online]. Available: https://scikit-image.org. [Diakses 8 11 2019].

[27 “Sci-Py.org,” Sci-Py.org, [Online]. Available: https://www.scipy.org/about.html. [Diakses 8 11 2019].

[28 P. Baskara, Pengenalan Nomor Polisi Kendaraan pada Data Video menggunakan Convolutional Neural Network, Surabaya: Institut Teknologi Sepuluh Nopember, 2019.

[29 A. Winarto, "Variations of SSD—Understanding Deconvolutional Single-Shot Detectors," Medium, 25 8 2018. [Online]. Available: https://medium.com/@amadeusw6/variations-of-ssd-understanding-deconvolutional-single-shot-detectors-c0afb8686d03. [Diakses 8 11 2019].

[30 "Elman Networks," Mnemosyne Studio, [Online]. Available: http://mnemstudio.org/neural-networks-elman.htm. [Diakses 8 11 2019].

LAMPIRAN

L.1 Hasil Uji Coba Preset VGG300 pada SSD

Frame	**Cakupan**	**Total**	**Akurasi**
Frame 1	9	9	100.00%
Frame 2	9	9	100.00%
Frame 3	8	9	88.89%
Frame 4	9	9	100.00%
Frame 5	8	8	100.00%
Frame 6	8	8	100.00%
Frame 7	6	7	85.71%
Frame 8	6	7	85.71%
Frame 9	5	7	71.43%
Frame 10	6	7	85.71%
Frame 11	7	7	100.00%
Frame 12	6	7	85.71%
Frame 13	7	7	100.00%
Frame 14	7	7	100.00%
Frame 15	8	8	100.00%
Frame 16	7	7	100.00%
Frame 17	6	7	85.71%
Frame 18	6	7	85.71%
Frame 19	7	7	100.00%
Rata-rata			93.40%

L.2 Hasil Uji Coba Preset VGG512 pada SSD

Frame	**Cakupan**	**Total**	**Akurasi**
Frame 1	9	9	100.00%

Frame	Cakupan	Total	Akurasi
Frame 2	9	9	100.00%
Frame 3	9	9	100.00%
Frame 4	8	9	88.89%
Frame 5	8	8	100.00%
Frame 6	8	8	100.00%
Frame 7	6	7	85.71%
Frame 8	6	7	85.71%
Frame 9	7	7	100.00%
Frame 10	7	7	100.00%
Frame 11	6	7	85.71%
Frame 12	6	7	85.71%
Frame 13	0	7	0.00%
Frame 14	7	7	100.00%
Frame 15	8	8	100.00%
Frame 16	7	7	100.00%
Frame 17	6	7	85.71%
Frame 18	7	7	100.00%
Frame 19	7	7	100.00%
Rata-rata			90.39%

L.3 Hasil Uji Coba Learning Rate 0.00075 pada SSD

Frame	Cakupan	Total	Akurasi
Frame 1	9	9	100.00%
Frame 2	9	9	100.00%
Frame 3	8	9	88.89%
Frame 4	9	9	100.00%
Frame 5	8	8	100.00%

Frame	Cakupan	Total	Akurasi
Frame 6	8	8	100.00%
Frame 7	7	7	100.00%
Frame 8	6	7	85.71%
Frame 9	6	7	85.71%
Frame 10	6	7	85.71%
Frame 11	7	7	100.00%
Frame 12	7	7	100.00%
Frame 13	0	7	0.00%
Frame 14	6	7	85.71%
Frame 15	8	8	100.00%
Frame 16	6	7	85.71%
Frame 17	6	7	85.71%
Frame 18	5	7	71.43%
Frame 19	3.5	7	50.00%
Rata-rata			85.51%

L.4 Hasil Uji Coba Learning Rate 0.0001 pada SSD

Frame	Cakupan	Total	Akurasi
Frame 1	9	9	100.00%
Frame 2	9	9	100.00%
Frame 3	8	9	88.89%
Frame 4	9	9	100.00%
Frame 5	8	8	100.00%
Frame 6	8	8	100.00%
Frame 7	6	7	85.71%
Frame 8	6	7	85.71%
Frame 9	5	7	71.43%

Frame	Cakupan	Total	Akurasi
Frame 10	6	7	85.71%
Frame 11	7	7	100.00%
Frame 12	6	7	85.71%
Frame 13	7	7	100.00%
Frame 14	7	7	100.00%
Frame 15	8	8	100.00%
Frame 16	7	7	100.00%
Frame 17	6	7	85.71%
Frame 18	6	7	85.71%
Frame 19	7	7	100.00%
Rata-rata			93.40%

L.5 Hasil Uji Coba Learning Rate 0.000075 pada SSD

Frame	Cakupan	Total	Akurasi
Frame 1	9	9	100.00%
Frame 2	8	9	88.89%
Frame 3	8	9	88.89%
Frame 4	8	9	88.89%
Frame 5	8	8	100.00%
Frame 6	8	8	100.00%
Frame 7	6	7	85.71%
Frame 8	6	7	85.71%
Frame 9	7	7	100.00%
Frame 10	7	7	100.00%
Frame 11	7	7	100.00%
Frame 12	7	7	100.00%
Frame 13	6	7	85.71%

Frame	Cakupan	Total	Akurasi
Frame 14	7	7	100.00%
Frame 15	8	8	100.00%
Frame 16	6	7	85.71%
Frame 17	6	7	85.71%
Frame 18	5	7	71.43%
Frame 19	7	7	100.00%
Rata-rata			92.98 %

L.6 Hasil Uji Coba Learning Rate 0.00001 pada SSD

Frame	Cakupan	Total	Akurasi
Frame 1	0	9	0.00%
Frame 2	0	9	0.00%
Frame 3	0	9	0.00%
Frame 4	0	9	0.00%
Frame 5	0	8	0.00%
Frame 6	8	8	100.00%
Frame 7	0	7	0.00%
Frame 8	0	7	0.00%
Frame 9	0	7	0.00%
Frame 10	0	7	0.00%
Frame 11	0	7	0.00%
Frame 12	0	7	0.00%
Frame 13	0	7	0.00%
Frame 14	0	7	0.00%
Frame 15	0	8	0.00%
Frame 16	0	7	0.00%
Frame 17	0	7	0.00%

Frame	Cakupan	Total	Akurasi
Frame 18	0	7	0.00%
Frame 19	6	7	85.71%
Rata-rata			9.77%

L.7 Hasil Uji Coba Anotasi Lebar pada SSD

Frame	Cakupan	Total	Akurasi
Frame 1	9	9	100.00%
Frame 2	9	9	100.00%
Frame 3	9	9	100.00%
Frame 4	9	9	100.00%
Frame 5	8	8	100.00%
Frame 6	8	8	100.00%
Frame 7	6	7	85.71%
Frame 8	6	7	85.71%
Frame 9	6	7	85.71%
Frame 10	7	7	100.00%
Frame 11	7	7	100.00%
Frame 12	7	7	100.00%
Frame 13	7	7	100.00%
Frame 14	7	7	100.00%
Frame 15	8	8	100.00%
Frame 16	7	7	100.00%
Frame 17	7	7	100.00%
Frame 18	6	7	85.71%
Frame 19	7	7	100.00%
Rata-rata			96.99%

L.8 Hasil Uji Coba Anotasi Asli pada SSD

Frame	**Cakupan**	**Total**	**Akurasi**
Frame 1	9	9	100.00%
Frame 2	9	9	100.00%
Frame 3	8	9	88.89%
Frame 4	9	9	100.00%
Frame 5	8	8	100.00%
Frame 6	8	8	100.00%
Frame 7	6	7	85.71%
Frame 8	6	7	85.71%
Frame 9	5	7	71.43%
Frame 10	6	7	85.71%
Frame 11	7	7	100.00%
Frame 12	6	7	85.71%
Frame 13	7	7	100.00%
Frame 14	7	7	100.00%
Frame 15	8	8	100.00%
Frame 16	7	7	100.00%
Frame 17	6	7	85.71%
Frame 18	6	7	85.71%
Frame 19	7	7	100.00%
Rata-rata			93.40%

L.9 Hasil Uji Coba Anotasi Sempit pada SSD

Frame	**Cakupan**	**Total**	**Akurasi**
Frame 1	8	9	88.89%
Frame 2	9	9	100.00%

Frame	Cakupan	Total	Akurasi
Frame 3	8	9	88.89%
Frame 4	8	9	88.89%
Frame 5	8	8	100.00%
Frame 6	8	8	100.00%
Frame 7	6	7	85.71%
Frame 8	6	7	85.71%
Frame 9	6	7	85.71%
Frame 10	6	7	85.71%
Frame 11	6	7	85.71%
Frame 12	6	7	85.71%
Frame 13	6	7	85.71%
Frame 14	6	7	85.71%
Frame 15	8	8	100.00%
Frame 16	6	7	85.71%
Frame 17	6	7	85.71%
Frame 18	6	7	85.71%
Frame 19	7	7	100.00%
Rata-rata			89.97%

L.10 Hasil Uji Coba Optimizer SGD pada RNN

Kelas	*Precision*	*Recall*	F1-Score	*Support*
0	0.0000	0.0000	0.0000	3293
1	0.3931	0.4197	0.4060	3765
2	0.0000	0.0000	0.0000	3294
3	0.0000	0.0000	0.0000	3756
4	0.1667	0.0003	0.0006	3294
5	0.0000	0.0000	0.0000	2344

Kelas	***Precision***	***Recall***	**F1-Score**	***Support***
6	0.0000	0.0000	0.0000	1938
7	0.2123	0.0666	0.1014	2493
8	0.0000	0.0000	0.0000	2484
9	0.2174	0.0078	0.0150	1932
A	0.2911	0.0564	0.0944	2484
B	0.0000	0.0000	0.0000	2202
C	0.3945	0.5697	0.4662	1931
D	0.0023	0.0004	0.0007	2485
E	0.2520	0.8059	0.3839	1937
F	0.3021	0.5326	0.3855	3040
G	0.6387	0.2100	0.3161	2214
H	0.1576	0.0603	0.0872	1925
I	0.0995	0.6789	0.1736	2485
J	0.4869	0.6408	0.5534	3046
K	0.3404	0.0072	0.0141	2215
L	0.4319	0.5952	0.5006	2762
M	0.4207	0.1575	0.2292	1937
N	0.0000	0.0000	0.0000	1939
O	0.2860	0.1730	0.2156	1931
P	0.2481	0.3613	0.2942	2214
Q	0.0392	0.0083	0.0136	1938
R	0.0831	0.4681	0.1411	3044
S	0.0868	0.0213	0.0343	3046
T	0.1112	0.4043	0.1744	2768
U	0.0775	0.0072	0.0132	2763
V	0.1240	0.3464	0.1826	3323
W	0.1591	0.0281	0.0478	2491
X	0.1452	0.0316	0.0519	1931

Kelas	*Precision*	*Recall*	F1-Score	*Support*
Y	0.0016	0.0005	0.0007	2209
Z	0.1028	0.0149	0.0261	2480
Rata-rata	0.1713	0.1912	0.1396	91333

L.11 Hasil Uji Coba Optimizer Adagrad pada RNN

Kelas	*Precision*	*Recall*	F1-Score	*Support*
0	0.9766	0.9241	0.9496	3293
1	0.9910	0.9918	0.9914	3765
2	0.9836	0.9654	0.9744	3294
3	0.9909	0.9811	0.9860	3756
4	0.9864	0.9715	0.9789	3294
5	0.9914	0.9829	0.9871	2344
6	0.9565	0.9634	0.9599	1938
7	0.9538	0.9679	0.9608	2493
8	0.9303	0.9614	0.9456	2484
9	0.9823	0.9741	0.9782	1932
A	0.9820	0.9879	0.9849	2484
B	0.9442	0.9214	0.9327	2202
C	0.9901	0.9808	0.9854	1931
D	0.9564	0.9622	0.9593	2485
E	0.9875	0.9809	0.9842	1937
F	0.9837	0.9911	0.9874	3040
G	0.9807	0.9869	0.9838	2214
H	0.9691	0.9621	0.9656	1925
I	0.9815	0.9827	0.9821	2485
J	0.9891	0.9862	0.9877	3046
K	0.9816	0.9869	0.9842	2215

Kelas	*Precision*	*Recall*	F1-Score	*Support*
L	0.9656	0.9848	0.9751	2762
M	0.9892	0.9907	0.9899	1937
N	0.9765	0.9644	0.9704	1939
O	0.9748	0.9612	0.9679	1931
P	0.9852	0.9950	0.9901	2214
Q	0.9308	0.9654	0.9478	1938
R	0.9765	0.9813	0.9789	3044
S	0.9777	0.9921	0.9848	3046
T	0.9767	0.9859	0.9813	2768
U	0.9671	0.9779	0.9725	2763
V	0.9865	0.9922	0.9893	3323
W	0.9872	0.9880	0.9876	2491
X	0.9772	0.9751	0.9762	1931
Y	0.9793	0.9864	0.9829	2209
Z	0.9705	0.9698	0.9701	2480
Rata-rata	0.9762	0.9761	0.9761	91333

L.12 Hasil Uji Coba Optimizer RMSProp pada RNN

Kelas	*Precision*	*Recall*	F1-Score	*Support*
0	0.9893	0.9554	0.9720	3293
1	0.9986	0.9814	0.9900	3765
2	0.9939	0.9961	0.9950	3294
3	0.9995	0.9963	0.9979	3756
4	0.9957	0.9900	0.9928	3294
5	0.9961	0.9881	0.9921	2344
6	0.9979	0.9923	0.9951	1938
7	0.9872	0.9904	0.9888	2493

Kelas	*Precision*	*Recall*	F1-Score	*Support*
8	0.9872	0.9952	0.9912	2484
9	0.9974	0.9974	0.9974	1932
A	0.9964	0.9952	0.9958	2484
B	0.9941	0.9927	0.9934	2202
C	0.9969	0.9959	0.9964	1931
D	0.9882	0.9807	0.9844	2485
E	0.9969	0.9948	0.9959	1937
F	0.9941	0.9961	0.9951	3040
G	0.9905	0.9892	0.9898	2214
H	0.9932	0.9844	0.9888	1925
I	0.9891	0.9875	0.9883	2485
J	0.9922	0.9974	0.9948	3046
K	0.9910	0.9991	0.9951	2215
L	0.9867	0.9946	0.9906	2762
M	0.9979	0.9985	0.9982	1937
N	0.9867	0.9959	0.9913	1939
O	0.9708	0.9990	0.9847	1931
P	0.9977	0.9977	0.9977	2214
Q	0.9558	0.9923	0.9737	1938
R	0.9984	0.9980	0.9982	3044
S	0.9980	0.9970	0.9975	3046
T	0.9881	0.9884	0.9883	2768
U	0.9834	0.9844	0.9839	2763
V	0.9928	0.9955	0.9941	3323
W	0.9992	0.9968	0.9980	2491
X	0.9990	0.9860	0.9924	1931
Y	0.9757	0.9991	0.9873	2209
Z	0.9872	0.9940	0.9906	2480

Kelas	*Precision*	*Recall*	F1-Score	*Support*
Rata-rata	0.9917	0.9916	0.9916	91333

L.13 Hasil Uji Coba Optimizer Adam pada RNN

Kelas	*Precision*	*Recall*	F1-Score	*Support*
0	0.9941	0.9681	0.9809	3293
1	0.9992	0.9976	0.9984	3765
2	0.9979	0.9985	0.9982	3294
3	0.9987	0.9984	0.9985	3756
4	0.9964	0.9945	0.9954	3294
5	0.9983	0.9962	0.9972	2344
6	0.9985	0.9995	0.9990	1938
7	0.9826	0.9976	0.9900	2493
8	0.9948	0.9936	0.9942	2484
9	0.9979	0.9984	0.9982	1932
A	0.9984	0.9980	0.9982	2484
B	0.9954	0.9900	0.9927	2202
C	0.9995	0.9990	0.9992	1931
D	0.9763	0.9936	0.9848	2485
E	0.9995	0.9974	0.9984	1937
F	0.9967	0.9993	0.9980	3040
G	1.0000	1.0000	1.0000	2214
H	0.9928	0.9964	0.9946	1925
I	0.9880	0.9968	0.9924	2485
J	1.0000	0.9974	0.9987	3046
K	0.9991	0.9986	0.9989	2215
L	0.9946	0.9964	0.9955	2762
M	1.0000	0.9990	0.9995	1937

Kelas	*Precision*	*Recall*	F1-Score	*Support*
N	0.9969	0.9995	0.9982	1939
O	0.9923	0.9974	0.9948	1931
P	0.9995	0.9977	0.9986	2214
Q	0.9948	0.9835	0.9891	1938
R	0.9993	0.9967	0.9980	3044
S	0.9987	0.9990	0.9989	3046
T	0.9960	0.9877	0.9918	2768
U	0.9812	0.9996	0.9903	2763
V	0.9982	0.9988	0.9985	3323
W	0.9988	0.9996	0.9992	2491
X	0.9984	0.9969	0.9977	1931
Y	0.9982	0.9991	0.9986	2209
Z	0.9931	0.9903	0.9917	2480
Rata-rata	0.9957	0.9957	0.9957	91333

L.14 Hasil Uji Coba Learning Rate 0.01 pada RNN

Kelas	*Precision*	*Recall*	F1-Score	*Support*
0	0.0476	0.0109	0.0178	3293
1	0.0338	0.0013	0.0026	3765
2	0.0365	0.0121	0.0182	3294
3	0.0566	0.0008	0.0016	3756
4	0.0465	0.0024	0.0046	3294
5	0.0367	0.0094	0.0149	2344
6	0.0238	0.0114	0.0154	1938
7	0.0311	0.0193	0.0238	2493
8	0.0410	0.0076	0.0129	2484
9	0.0218	0.0150	0.0178	1932

Kelas	***Precision***	***Recall***	**F1-Score**	***Support***
A	0.0241	0.0036	0.0063	2484
B	0.0261	0.2084	0.0463	2202
C	0.0197	0.0057	0.0088	1931
D	0.0383	0.0145	0.0210	2485
E	0.0241	0.0031	0.0055	1937
F	0.0328	0.1592	0.0544	3040
G	0.0288	0.0172	0.0215	2214
H	0.0176	0.0384	0.0242	1925
I	0.0256	0.0004	0.0008	2485
J	0.0307	0.0381	0.0340	3046
K	0.0172	0.0027	0.0047	2215
L	0.0299	0.0196	0.0236	2762
M	0.0202	0.0754	0.0318	1937
N	0.0187	0.0129	0.0153	1939
O	0.0163	0.0104	0.0127	1931
P	0.0275	0.0072	0.0114	2214
Q	0.0000	0.0000	0.0000	1938
R	0.0309	0.0493	0.0380	3044
S	0.0341	0.0394	0.0366	3046
T	0.0280	0.0177	0.0217	2768
U	0.0295	0.0130	0.0181	2763
V	0.0361	0.1086	0.0542	3323
W	0.0220	0.0181	0.0199	2491
X	0.0085	0.0010	0.0018	1931
Y	0.0243	0.0041	0.0070	2209
Z	0.0304	0.0476	0.0371	2480
Rata-rata	0.0300	0.0287	0.0197	91333

L.15 Hasil Uji Coba Learning Rate 0.001 pada RNN

Kelas	*Precision*	*Recall*	F1-Score	*Support*
0	0.8409	0.9338	0.8849	3293
1	0.9780	0.8741	0.9231	3765
2	0.9846	0.9696	0.9771	3294
3	0.9948	0.9633	0.9788	3756
4	0.9281	0.9794	0.9530	3294
5	0.9811	0.9723	0.9766	2344
6	0.9244	0.9716	0.9474	1938
7	0.7948	0.9414	0.8619	2493
8	0.9781	0.7206	0.8299	2484
9	0.9614	0.9933	0.9771	1932
A	0.9135	0.9561	0.9343	2484
B	0.8075	0.9446	0.8707	2202
C	0.9691	0.9425	0.9556	1931
D	0.9170	0.8089	0.8595	2485
E	0.9685	0.8895	0.9273	1937
F	0.9623	0.9487	0.9554	3040
G	0.9084	0.9228	0.9155	2214
H	0.9066	0.9429	0.9244	1925
I	0.8899	0.9404	0.9145	2485
J	0.9677	0.9833	0.9754	3046
K	0.8749	0.9666	0.9185	2215
L	0.9135	0.8867	0.8999	2762
M	0.9948	0.9892	0.9920	1937
N	0.9692	0.9103	0.9388	1939
O	0.9421	0.9518	0.9469	1931

Kelas	*Precision*	*Recall*	F1-Score	*Support*
P	0.9707	0.9860	0.9783	2214
Q	0.9722	0.7946	0.8745	1938
R	0.9278	0.9790	0.9527	3044
S	0.9646	0.9754	0.9700	3046
T	0.9085	0.9581	0.9327	2768
U	0.9623	0.9595	0.9609	2763
V	0.9619	0.9874	0.9745	3323
W	0.9593	0.9358	0.9474	2491
X	0.9934	0.8519	0.9172	1931
Y	0.9860	0.9226	0.9532	2209
Z	0.8986	0.9536	0.9253	2480
Rata-rata	0.9386	0.9356	0.9353	91333

L.16 Hasil Uji Coba Learning Rate 0.0001 pada RNN

Kelas	*Precision*	*Recall*	F1-Score	*Support*
0	0.9941	0.9681	0.9809	3293
1	0.9992	0.9976	0.9984	3765
2	0.9979	0.9985	0.9982	3294
3	0.9987	0.9984	0.9985	3756
4	0.9964	0.9945	0.9954	3294
5	0.9983	0.9962	0.9972	2344
6	0.9985	0.9995	0.9990	1938
7	0.9826	0.9976	0.9900	2493
8	0.9948	0.9936	0.9942	2484
9	0.9979	0.9984	0.9982	1932
A	0.9984	0.9980	0.9982	2484
B	0.9954	0.9900	0.9927	2202

Kelas	***Precision***	***Recall***	**F1-Score**	***Support***
C	0.9995	0.9990	0.9992	1931
D	0.9763	0.9936	0.9848	2485
E	0.9995	0.9974	0.9984	1937
F	0.9967	0.9993	0.9980	3040
G	1.0000	1.0000	1.0000	2214
H	0.9928	0.9964	0.9946	1925
I	0.9880	0.9968	0.9924	2485
J	1.0000	0.9974	0.9987	3046
K	0.9991	0.9986	0.9989	2215
L	0.9946	0.9964	0.9955	2762
M	1.0000	0.9990	0.9995	1937
N	0.9969	0.9995	0.9982	1939
O	0.9923	0.9974	0.9948	1931
P	0.9995	0.9977	0.9986	2214
Q	0.9948	0.9835	0.9891	1938
R	0.9993	0.9967	0.9980	3044
S	0.9987	0.9990	0.9989	3046
T	0.9960	0.9877	0.9918	2768
U	0.9812	0.9996	0.9903	2763
V	0.9982	0.9988	0.9985	3323
W	0.9988	0.9996	0.9992	2491
X	0.9984	0.9969	0.9977	1931
Y	0.9982	0.9991	0.9986	2209
Z	0.9931	0.9903	0.9917	2480
Rata-rata	0.9957	0.9957	0.9957	91333

L.17 Hasil Uji Coba 3-Folds Cross Validation pada RNN

Folds	Akurasi
1	0.9961
2	0.9937
3	0.9937

L.18 Hasil Uji Coba 6-Folds Cross Validation pada RNN

Folds	Akurasi
1	0.9960
2	0.9948
3	0.9959
4	0.9963
5	0.9964
6	0.9950

L.19 Hasil Uji Coba 9-Folds Cross Validation pada RNN

Folds	Akurasi
1	0.9969
2	0.9951
3	0.9962
4	0.9958
5	0.9944
6	0.9966
7	0.9965
8	0.9967
9	0.9969

L.20 Hasil Uji Coba 12-Folds Cross Validation pada RNN

Folds	Akurasi
1	0.9966
2	0.9966
3	0.9954
4	0.9957
5	0.9959
6	0.9955
7	0.9968
8	0.9963
9	0.9962
10	0.9958
11	0.9960
12	0.9956

L.21 Hasil Uji Coba Lokalisasi pada Video Siang

Video	*Frame*	Cakupan	Total	Akurasi
Video 1	*Frame* 1	8	8	100.00%
Video 1	*Frame* 2	8	8	100.00%
Video 1	*Frame* 3	8	8	100.00%
Video 1	*Frame* 4	8	8	100.00%
Video 1	*Frame* 5	8	8	100.00%
Video 1	*Frame* 6	8	8	100.00%
Video 1	*Frame* 7	8	8	100.00%
Video 1	*Frame* 8	8	8	100.00%
Video 1	*Frame* 9	8	8	100.00%
Video 1	*Frame* 10	8	8	100.00%

Video	*Frame*	Cakupan	Total	Akurasi
Video 1	*Frame* 11	8	8	100.00%
Video 1	*Frame* 12	8	8	100.00%
Video 1	*Frame* 13	8	8	100.00%
Video 1	*Frame* 14	8	8	100.00%
Video 1	*Frame* 15	8	8	100.00%
Video 1	*Frame* 16	8	8	100.00%
Video 1	*Frame* 17	8	8	100.00%
Video 1	*Frame* 18	8	8	100.00%
Video 1	*Frame* 19	8	8	100.00%
Video 1	*Frame* 20	8	8	100.00%
Video 2	*Frame* 1	9	9	100.00%
Video 2	*Frame* 2	8	9	88.89%
Video 2	*Frame* 3	8	9	88.89%
Video 2	*Frame* 4	8	9	88.89%
Video 2	*Frame* 5	8	9	88.89%
Video 2	*Frame* 6	8	9	88.89%
Video 2	*Frame* 7	9	9	100.00%
Video 2	*Frame* 8	9	9	100.00%
Video 2	*Frame* 9	8	9	88.89%
Video 2	*Frame* 10	9	9	100.00%
Video 2	*Frame* 11	8	9	88.89%
Video 2	*Frame* 12	8	9	88.89%
Video 2	*Frame* 13	9	9	100.00%
Video 2	*Frame* 14	8	9	88.89%
Video 2	*Frame* 15	8	9	88.89%
Video 2	*Frame* 16	8	9	88.89%
Video 2	*Frame* 17	9	9	100.00%
Video 2	*Frame* 18	9	9	100.00%

Video	*Frame*	Cakupan	Total	Akurasi
Video 2	*Frame* 19	0	9	0.00%
Video 2	*Frame* 20	0	9	0.00%
Video 3	*Frame* 1	0	8	0.00%
Video 3	*Frame* 2	8	8	100.00%
Video 3	*Frame* 3	8	8	100.00%
Video 3	*Frame* 4	8	8	100.00%
Video 3	*Frame* 5	8	8	100.00%
Video 3	*Frame* 6	8	8	100.00%
Video 3	*Frame* 7	8	8	100.00%
Video 3	*Frame* 8	8	8	100.00%
Video 3	*Frame* 9	8	8	100.00%
Video 3	*Frame* 10	8	8	100.00%
Video 3	*Frame* 11	8	8	100.00%
Video 3	*Frame* 12	8	8	100.00%
Video 3	*Frame* 13	8	8	100.00%
Video 3	*Frame* 14	8	8	100.00%
Video 3	*Frame* 15	8	8	100.00%
Video 3	*Frame* 16	8	8	100.00%
Video 3	*Frame* 17	8	8	100.00%
Video 3	*Frame* 18	8	8	100.00%
Video 3	*Frame* 19	8	8	100.00%
Video 3	*Frame* 20	8	8	100.00%
Video 4	*Frame* 1	7	7	100.00%
Video 4	*Frame* 2	7	7	100.00%
Video 4	*Frame* 3	7	7	100.00%
Video 4	*Frame* 4	7	7	100.00%
Video 4	*Frame* 5	7	7	100.00%
Video 4	*Frame* 6	7	7	100.00%

Video	*Frame*	Cakupan	Total	Akurasi
Video 4	*Frame* 7	7	7	100.00%
Video 4	*Frame* 8	7	7	100.00%
Video 4	*Frame* 9	7	7	100.00%
Video 4	*Frame* 10	7	7	100.00%
Video 4	*Frame* 11	7	7	100.00%
Video 4	*Frame* 12	7	7	100.00%
Video 4	*Frame* 13	7	7	100.00%
Video 4	*Frame* 14	7	7	100.00%
Video 4	*Frame* 15	7	7	100.00%
Video 4	*Frame* 16	7	7	100.00%
Video 4	*Frame* 17	7	7	100.00%
Video 4	*Frame* 18	7	7	100.00%
Video 4	*Frame* 19	7	7	100.00%
Video 4	*Frame* 20	7	7	100.00%
Video 5	*Frame* 1	7	7	100.00%
Video 5	*Frame* 2	7	7	100.00%
Video 5	*Frame* 3	7	7	100.00%
Video 5	*Frame* 4	7	7	100.00%
Video 5	*Frame* 5	7	7	100.00%
Video 5	*Frame* 6	7	7	100.00%
Video 5	*Frame* 7	7	7	100.00%
Video 5	*Frame* 8	7	7	100.00%
Video 5	*Frame* 9	7	7	100.00%
Video 5	*Frame* 10	7	7	100.00%
Video 5	*Frame* 11	7	7	100.00%
Video 5	*Frame* 12	7	7	100.00%
Video 5	*Frame* 13	7	7	100.00%
Video 5	*Frame* 14	7	7	100.00%

Video	*Frame*	Cakupan	Total	Akurasi
Video 5	*Frame* 15	7	7	100.00%
Video 5	*Frame* 16	7	7	100.00%
Video 5	*Frame* 17	7	7	100.00%
Video 5	*Frame* 18	7	7	100.00%
Video 5	*Frame* 19	7	7	100.00%
Video 5	*Frame* 20	7	7	100.00%
Video 6	*Frame* 1	0	7	0.00%
Video 6	*Frame* 2	7	7	100.00%
Video 6	*Frame* 3	7	7	100.00%
Video 6	*Frame* 4	7	7	100.00%
Video 6	*Frame* 5	7	7	100.00%
Video 6	*Frame* 6	7	7	100.00%
Video 6	*Frame* 7	7	7	100.00%
Video 6	*Frame* 8	7	7	100.00%
Video 6	*Frame* 9	7	7	100.00%
Video 6	*Frame* 10	7	7	100.00%
Video 6	*Frame* 11	7	7	100.00%
Video 6	*Frame* 12	7	7	100.00%
Video 6	*Frame* 13	7	7	100.00%
Video 6	*Frame* 14	7	7	100.00%
Video 6	*Frame* 15	7	7	100.00%
Video 6	*Frame* 16	7	7	100.00%
Video 6	*Frame* 17	7	7	100.00%
Video 6	*Frame* 18	7	7	100.00%
Video 6	*Frame* 19	7	7	100.00%
Video 6	*Frame* 20	6	7	85.71%
Video 7	*Frame* 1	7	7	100.00%
Video 7	*Frame* 2	7	7	100.00%

Video	*Frame*	Cakupan	Total	Akurasi
Video 7	*Frame* 3	7	7	100.00%
Video 7	*Frame* 4	7	7	100.00%
Video 7	*Frame* 5	7	7	100.00%
Video 7	*Frame* 6	7	7	100.00%
Video 7	*Frame* 7	7	7	100.00%
Video 7	*Frame* 8	7	7	100.00%
Video 7	*Frame* 9	7	7	100.00%
Video 7	*Frame* 10	7	7	100.00%
Video 7	*Frame* 11	7	7	100.00%
Video 7	*Frame* 12	7	7	100.00%
Video 7	*Frame* 13	7	7	100.00%
Video 7	*Frame* 14	7	7	100.00%
Video 7	*Frame* 15	7	7	100.00%
Video 7	*Frame* 16	7	7	100.00%
Video 7	*Frame* 17	7	7	100.00%
Video 7	*Frame* 18	7	7	100.00%
Video 7	*Frame* 19	7	7	100.00%
Video 7	*Frame* 20	7	7	100.00%
Video 8	*Frame* 1	7	7	100.00%
Video 8	*Frame* 2	6	7	85.71%
Video 8	*Frame* 3	7	7	100.00%
Video 8	*Frame* 4	7	7	100.00%
Video 8	*Frame* 5	6	7	85.71%
Video 8	*Frame* 6	7	7	100.00%
Video 8	*Frame* 7	0	7	0.00%
Video 8	*Frame* 8	7	7	100.00%
Video 8	*Frame* 9	6	7	85.71%
Video 8	*Frame* 10	0	7	0.00%

Video	*Frame*	Cakupan	Total	Akurasi
Video 8	*Frame* 11	0	7	0.00%
Video 8	*Frame* 12	7	7	100.00%
Video 8	*Frame* 13	7	7	100.00%
Video 8	*Frame* 14	0	7	0.00%
Video 8	*Frame* 15	0	7	0.00%
Video 8	*Frame* 16	0	7	0.00%
Video 8	*Frame* 17	0	7	0.00%
Video 8	*Frame* 18	0	7	0.00%
Video 8	*Frame* 19	0	7	0.00%
Video 8	*Frame* 20	0	7	0.00%
Video 9	*Frame* 1	8	8	100.00%
Video 9	*Frame* 2	8	8	100.00%
Video 9	*Frame* 3	8	8	100.00%
Video 9	*Frame* 4	8	8	100.00%
Video 9	*Frame* 5	8	8	100.00%
Video 9	*Frame* 6	8	8	100.00%
Video 9	*Frame* 7	8	8	100.00%
Video 9	*Frame* 8	8	8	100.00%
Video 9	*Frame* 9	8	8	100.00%
Video 9	*Frame* 10	8	8	100.00%
Video 9	*Frame* 11	8	8	100.00%
Video 9	*Frame* 12	8	8	100.00%
Video 9	*Frame* 13	8	8	100.00%
Video 9	*Frame* 14	8	8	100.00%
Video 9	*Frame* 15	8	8	100.00%
Video 9	*Frame* 16	8	8	100.00%
Video 9	*Frame* 17	8	8	100.00%
Video 9	*Frame* 18	8	8	100.00%

Video	*Frame*	Cakupan	Total	Akurasi
Video 9	*Frame* 19	8	8	100.00%
Video 9	*Frame* 20	8	8	100.00%
Video 10	*Frame* 1	7	7	100.00%
Video 10	*Frame* 2	7	7	100.00%
Video 10	*Frame* 3	7	7	100.00%
Video 10	*Frame* 4	7	7	100.00%
Video 10	*Frame* 5	7	7	100.00%
Video 10	*Frame* 6	7	7	100.00%
Video 10	*Frame* 7	7	7	100.00%
Video 10	*Frame* 8	7	7	100.00%
Video 10	*Frame* 9	7	7	100.00%
Video 10	*Frame* 10	7	7	100.00%
Video 10	*Frame* 11	7	7	100.00%
Video 10	*Frame* 12	7	7	100.00%
Video 10	*Frame* 13	7	7	100.00%
Video 10	*Frame* 14	7	7	100.00%
Video 10	*Frame* 15	7	7	100.00%
Video 10	*Frame* 16	7	7	100.00%
Video 10	*Frame* 17	7	7	100.00%
Video 10	*Frame* 18	7	7	100.00%
Video 10	*Frame* 19	7	7	100.00%
Video 10	*Frame* 20	7	7	100.00%

L.22 Hasil Uji Coba Segmentasi pada Video Siang

Video	*Frame*	TP	FP	FN	TN	Akurasi	Prec.	*Recall*
V1	F1	7	1	1	0	77.78%	87.50%	87.50%
V1	F2	7	1	1	0	77.78%	87.50%	87.50%

Video	*Frame*	TP	FP	FN	TN	Akurasi	Prec.	*Recall*
V1	F3	7	1	1	0	77.78%	87.50%	87.50%
V1	F4	7	1	1	0	77.78%	87.50%	87.50%
V1	F5	7	1	1	0	77.78%	87.50%	87.50%
V1	F6	7	1	1	0	77.78%	87.50%	87.50%
V1	F7	7	1	1	0	77.78%	87.50%	87.50%
V1	F8	7	1	1	0	77.78%	87.50%	87.50%
V1	F9	7	1	1	0	77.78%	87.50%	87.50%
V1	F10	7	1	1	0	77.78%	87.50%	87.50%
V1	F11	7	1	1	0	77.78%	87.50%	87.50%
V1	F12	7	1	1	0	77.78%	87.50%	87.50%
V1	F13	7	1	1	0	77.78%	87.50%	87.50%
V1	F14	7	1	1	0	77.78%	87.50%	87.50%
V1	F15	7	1	1	0	77.78%	87.50%	87.50%
V1	F16	7	1	1	0	77.78%	87.50%	87.50%
V1	F17	7	1	1	0	77.78%	87.50%	87.50%
V1	F18	7	1	1	0	77.78%	87.50%	87.50%
V1	F19	7	1	1	0	77.78%	87.50%	87.50%
V1	F20	7	1	1	0	77.78%	87.50%	87.50%
V2	F1	9	1	0	0	90.00%	90.00%	100.00%
V2	F2	9	0	0	0	100.00%	100.00%	100.00%
V2	F3	9	1	0	0	90.00%	90.00%	100.00%
V2	F4	9	4	0	0	69.23%	69.23%	100.00%
V2	F5	9	3	0	0	75.00%	75.00%	100.00%
V2	F6	9	2	0	0	81.82%	81.82%	100.00%
V2	F7	9	1	0	0	90.00%	90.00%	100.00%
V2	F8	9	1	0	0	90.00%	90.00%	100.00%
V2	F9	9	2	0	0	81.82%	81.82%	100.00%
V2	F10	9	1	0	0	90.00%	90.00%	100.00%

Video	*Frame*	TP	FP	FN	TN	Akurasi	Prec.	*Recall*
V2	F11	8	1	0	0	88.89%	88.89%	100.00%
V2	F12	9	2	0	0	81.82%	81.82%	100.00%
V2	F13	9	2	0	0	81.82%	81.82%	100.00%
V2	F14	8	2	0	0	80.00%	80.00%	100.00%
V2	F15	9	2	0	0	81.82%	81.82%	100.00%
V2	F16	9	1	0	0	90.00%	90.00%	100.00%
V2	F17	8	1	0	0	88.89%	88.89%	100.00%
V2	F18	9	2	0	0	81.82%	81.82%	100.00%
V3	F1	4	0	4	0	50.00%	100.00%	50.00%
V3	F2	4	2	4	0	40.00%	66.67%	50.00%
V3	F3	6	3	2	0	54.55%	66.67%	75.00%
V3	F4	5	0	3	0	62.50%	100.00%	62.50%
V3	F5	6	2	2	0	60.00%	75.00%	75.00%
V3	F6	3	3	5	0	27.27%	50.00%	37.50%
V3	F7	1	0	7	0	12.50%	100.00%	12.50%
V3	F8	0	3	8	0	0.00%	0.00%	0.00%
V3	F9	0	2	8	0	0.00%	0.00%	0.00%
V3	F10	0	1	8	0	0.00%	0.00%	0.00%
V3	F11	0	2	8	0	0.00%	0.00%	0.00%
V3	F12	0	0	8	0	0.00%	100.00%	0.00%
V3	F13	0	0	8	0	0.00%	100.00%	0.00%
V3	F14	1	0	7	0	12.50%	100.00%	12.50%
V3	F15	0	0	8	0	0.00%	100.00%	0.00%
V3	F16	1	0	7	0	12.50%	100.00%	12.50%
V3	F17	2	1	6	0	22.22%	66.67%	25.00%
V3	F18	2	0	6	0	25.00%	100.00%	25.00%
V3	F19	1	1	7	0	11.11%	50.00%	12.50%
V4	F1	7	0	0	0	100.00%	100.00%	100.00%

Video	*Frame*	TP	FP	FN	TN	Akurasi	Prec.	*Recall*
V4	F2	7	0	0	0	100.00%	100.00%	100.00%
V4	F3	7	0	0	0	100.00%	100.00%	100.00%
V4	F4	7	0	0	0	100.00%	100.00%	100.00%
V4	F5	7	0	0	0	100.00%	100.00%	100.00%
V4	F6	7	1	0	0	87.50%	87.50%	100.00%
V4	F7	7	0	0	0	100.00%	100.00%	100.00%
V4	F8	7	1	0	0	87.50%	87.50%	100.00%
V4	F9	7	2	0	0	77.78%	77.78%	100.00%
V4	F10	7	1	0	0	87.50%	87.50%	100.00%
V4	F11	7	0	0	0	100.00%	100.00%	100.00%
V4	F12	7	0	0	0	100.00%	100.00%	100.00%
V4	F13	7	0	0	0	100.00%	100.00%	100.00%
V4	F14	7	0	0	0	100.00%	100.00%	100.00%
V4	F15	2	1	5	0	25.00%	66.67%	28.57%
V4	F16	7	1	0	0	87.50%	87.50%	100.00%
V4	F17	7	0	0	0	100.00%	100.00%	100.00%
V4	F18	7	0	0	0	100.00%	100.00%	100.00%
V4	F19	7	0	0	0	100.00%	100.00%	100.00%
V4	F20	7	0	0	0	100.00%	100.00%	100.00%
V5	F1	6	0	1	0	85.71%	100.00%	85.71%
V5	F2	6	0	1	0	85.71%	100.00%	85.71%
V5	F3	6	0	1	0	85.71%	100.00%	85.71%
V5	F4	6	0	1	0	85.71%	100.00%	85.71%
V5	F5	6	0	1	0	85.71%	100.00%	85.71%
V5	F6	7	0	0	0	100.00%	100.00%	100.00%
V5	F7	4	0	3	0	57.14%	100.00%	57.14%
V5	F8	4	0	3	0	57.14%	100.00%	57.14%
V5	F9	7	0	0	0	100.00%	100.00%	100.00%

Video	*Frame*	TP	FP	FN	TN	Akurasi	Prec.	*Recall*
V5	F10	6	0	1	0	85.71%	100.00%	85.71%
V5	F11	6	0	1	0	85.71%	100.00%	85.71%
V5	F12	6	0	1	0	85.71%	100.00%	85.71%
V5	F13	7	0	0	0	100.00%	100.00%	100.00%
V5	F14	7	0	0	0	100.00%	100.00%	100.00%
V5	F15	7	0	0	0	100.00%	100.00%	100.00%
V5	F16	7	0	0	0	100.00%	100.00%	100.00%
V5	F17	6	0	1	0	85.71%	100.00%	85.71%
V5	F18	6	0	1	0	85.71%	100.00%	85.71%
V5	F19	5	0	2	0	71.43%	100.00%	71.43%
V5	F20	6	0	1	0	85.71%	100.00%	85.71%
V6	F1	7	0	0	0	100.00%	100.00%	100.00%
V6	F2	7	1	0	0	87.50%	87.50%	100.00%
V6	F3	7	0	0	0	100.00%	100.00%	100.00%
V6	F4	7	0	0	0	100.00%	100.00%	100.00%
V6	F5	7	0	0	0	100.00%	100.00%	100.00%
V6	F6	7	0	0	0	100.00%	100.00%	100.00%
V6	F7	7	1	0	0	87.50%	87.50%	100.00%
V6	F8	7	1	0	0	87.50%	87.50%	100.00%
V6	F9	7	0	0	0	100.00%	100.00%	100.00%
V6	F10	7	0	0	0	100.00%	100.00%	100.00%
V6	F11	7	0	0	0	100.00%	100.00%	100.00%
V6	F12	7	0	0	0	100.00%	100.00%	100.00%
V6	F13	7	0	0	0	100.00%	100.00%	100.00%
V6	F14	7	0	0	0	100.00%	100.00%	100.00%
V6	F15	7	0	0	0	100.00%	100.00%	100.00%
V6	F16	7	0	0	0	100.00%	100.00%	100.00%
V6	F17	7	0	0	0	100.00%	100.00%	100.00%

Video	*Frame*	TP	FP	FN	TN	Akurasi	Prec.	*Recall*
V6	F18	7	0	0	0	100.00%	100.00%	100.00%
V6	F19	6	0	0	0	100.00%	100.00%	100.00%
V7	F1	0	0	7	0	0.00%	100.00%	0.00%
V7	F2	6	0	1	0	85.71%	100.00%	85.71%
V7	F3	7	0	0	0	100.00%	100.00%	100.00%
V7	F4	7	1	0	0	87.50%	87.50%	100.00%
V7	F5	7	0	0	0	100.00%	100.00%	100.00%
V7	F6	7	0	0	0	100.00%	100.00%	100.00%
V7	F7	5	0	2	0	71.43%	100.00%	71.43%
V7	F8	7	0	0	0	100.00%	100.00%	100.00%
V7	F9	7	0	0	0	100.00%	100.00%	100.00%
V7	F10	6	0	1	0	85.71%	100.00%	85.71%
V7	F11	7	1	0	0	87.50%	87.50%	100.00%
V7	F12	7	0	0	0	100.00%	100.00%	100.00%
V7	F13	7	0	0	0	100.00%	100.00%	100.00%
V7	F14	7	0	0	0	100.00%	100.00%	100.00%
V7	F15	2	0	5	0	28.57%	100.00%	28.57%
V7	F16	7	1	0	0	87.50%	87.50%	100.00%
V7	F17	7	0	0	0	100.00%	100.00%	100.00%
V7	F18	7	0	0	0	100.00%	100.00%	100.00%
V7	F19	7	0	0	0	100.00%	100.00%	100.00%
V7	F20	7	1	0	0	87.50%	87.50%	100.00%
V8	F1	7	0	0	0	100.00%	100.00%	100.00%
V8	F2	6	0	0	0	100.00%	100.00%	100.00%
V8	F3	7	0	0	0	100.00%	100.00%	100.00%
V8	F4	7	0	0	0	100.00%	100.00%	100.00%
V8	F5	6	0	0	0	100.00%	100.00%	100.00%
V8	F6	7	1	0	0	87.50%	87.50%	100.00%

Video	*Frame*	TP	FP	FN	TN	Akurasi	Prec.	*Recall*
V8	F7	7	1	0	0	87.50%	87.50%	100.00%
V8	F8	4	0	2	0	66.67%	100.00%	66.67%
V8	F9	7	1	0	0	87.50%	87.50%	100.00%
V8	F10	7	1	0	0	87.50%	87.50%	100.00%
V9	F1	0	0	8	0	0.00%	100.00%	0.00%
V9	F2	8	0	0	0	100.00%	100.00%	100.00%
V9	F3	8	0	0	0	100.00%	100.00%	100.00%
V9	F4	8	1	0	0	88.89%	88.89%	100.00%
V9	F5	8	0	0	0	100.00%	100.00%	100.00%
V9	F6	8	1	0	0	88.89%	88.89%	100.00%
V9	F7	8	1	0	0	88.89%	88.89%	100.00%
V9	F8	8	1	0	0	88.89%	88.89%	100.00%
V9	F9	8	1	0	0	88.89%	88.89%	100.00%
V9	F10	8	1	0	0	88.89%	88.89%	100.00%
V9	F11	8	0	0	0	100.00%	100.00%	100.00%
V9	F12	8	0	0	0	100.00%	100.00%	100.00%
V9	F13	8	0	0	0	100.00%	100.00%	100.00%
V9	F14	8	0	0	0	100.00%	100.00%	100.00%
V9	F15	8	1	0	0	88.89%	88.89%	100.00%
V9	F16	8	0	0	0	100.00%	100.00%	100.00%
V9	F17	8	0	0	0	100.00%	100.00%	100.00%
V9	F18	8	0	0	0	100.00%	100.00%	100.00%
V9	F19	8	0	0	0	100.00%	100.00%	100.00%
V9	F20	8	0	0	0	100.00%	100.00%	100.00%
V10	F1	1	0	6	0	14.29%	100.00%	14.29%
V10	F2	7	0	0	0	100.00%	100.00%	100.00%
V10	F3	7	0	0	0	100.00%	100.00%	100.00%
V10	F4	7	0	0	0	100.00%	100.00%	100.00%

Video	*Frame*	TP	FP	FN	TN	Akurasi	Prec.	*Recall*
V10	F5	7	0	0	0	100.00%	100.00%	100.00%
V10	F6	7	0	0	0	100.00%	100.00%	100.00%
V10	F7	5	0	2	0	71.43%	100.00%	71.43%
V10	F8	7	0	0	0	100.00%	100.00%	100.00%
V10	F9	7	0	0	0	100.00%	100.00%	100.00%
V10	F10	7	0	0	0	100.00%	100.00%	100.00%
V10	F11	7	0	0	0	100.00%	100.00%	100.00%
V10	F12	7	0	0	0	100.00%	100.00%	100.00%
V10	F13	7	1	0	0	87.50%	87.50%	100.00%
V10	F14	7	0	0	0	100.00%	100.00%	100.00%
V10	F15	7	0	0	0	100.00%	100.00%	100.00%
V10	F16	7	0	0	0	100.00%	100.00%	100.00%
V10	F17	7	0	0	0	100.00%	100.00%	100.00%
V10	F18	7	0	0	0	100.00%	100.00%	100.00%
V10	F19	6	0	1	0	85.71%	100.00%	85.71%
V10	F20	7	1	0	0	87.50%	87.50%	100.00%

L.23 Hasil Uji Coba Majority Vote pada Video Siang

Vid.	*Majority Vote*	TP	FP	FN	TN	Akurasi	Prec.	*Recall*
V1	B 3578 TLV	7	1	0	0	87.50%	87.50%	100.00%
V2	AG 3192 6RA	8	1	1	0	80.00%	88.89%	88.89%
V3	1 F	2	0	6	0	25.00%	100.00%	25.00%
V4	L 2638 UI	7	0	0	0	100.00%	100.00%	100.00%
V5	L 3673 X	6	0	1	0	85.71%	100.00%	85.71%
V6	L 6382 ML	7	0	0	0	100.00%	100.00%	100.00%
V7	W 6770 WK	7	0	0	0	100.00%	100.00%	100.00%
V8	L 4025 H	6	0	1	0	85.71%	100.00%	85.71%

Vid.	*Majority Vote*	TP	FP	FN	TN	Akurasi	Prec.	*Recall*
V9	AE 3395 WR	8	0	0	0	100.00%	100.00%	100.00%
V10	W 3440 WJ	7	0	0	0	100.00%	100.00%	100.00%

L.24 Hasil Uji Coba Pengenalan Karakter pada Video Siang

Video	*Frame*	Benar	Total	Akurasi
V1	F1	7	7	100.00%
V1	F2	7	7	100.00%
V1	F3	7	7	100.00%
V1	F4	7	7	100.00%
V1	F5	7	7	100.00%
V1	F6	7	7	100.00%
V1	F7	7	7	100.00%
V1	F8	7	7	100.00%
V1	F9	7	7	100.00%
V1	F10	7	7	100.00%
V1	F11	7	7	100.00%
V1	F12	7	7	100.00%
V1	F13	7	7	100.00%
V1	F14	7	7	100.00%
V1	F15	7	7	100.00%
V1	F16	7	7	100.00%
V1	F17	7	7	100.00%
V1	F18	7	7	100.00%
V1	F19	7	7	100.00%
V1	F20	7	7	100.00%
V2	F1	8	9	88.89%
V2	F2	8	9	88.89%

Video	*Frame*	Benar	Total	Akurasi
V2	F3	9	9	100.00%
V2	F4	9	9	100.00%
V2	F5	9	9	100.00%
V2	F6	9	9	100.00%
V2	F7	8	9	88.89%
V2	F8	8	9	88.89%
V2	F9	9	9	100.00%
V2	F10	8	9	88.89%
V2	F11	8	8	100.00%
V2	F12	9	9	100.00%
V2	F13	9	9	100.00%
V2	F14	8	8	100.00%
V2	F15	9	9	100.00%
V2	F16	8	9	88.89%
V2	F17	8	8	100.00%
V2	F18	9	9	100.00%
V3	F1	4	4	100.00%
V3	F2	4	4	100.00%
V3	F3	5	6	83.33%
V3	F4	5	5	100.00%
V3	F5	6	6	100.00%
V3	F6	3	3	100.00%
V3	F7	1	1	100.00%
V3	F8	0	0	100.00%
V3	F9	0	0	100.00%
V3	F10	0	0	100.00%
V3	F11	0	0	100.00%
V3	F12	0	0	100.00%

Video	*Frame*	Benar	Total	Akurasi
V3	F13	0	0	100.00%
V3	F14	1	1	100.00%
V3	F15	0	0	100.00%
V3	F16	1	1	100.00%
V3	F17	2	2	100.00%
V3	F18	2	2	100.00%
V3	F19	1	1	100.00%
V4	F1	7	7	100.00%
V4	F2	7	7	100.00%
V4	F3	7	7	100.00%
V4	F4	7	7	100.00%
V4	F5	7	7	100.00%
V4	F6	7	7	100.00%
V4	F7	7	7	100.00%
V4	F8	7	7	100.00%
V4	F9	7	7	100.00%
V4	F10	7	7	100.00%
V4	F11	7	7	100.00%
V4	F12	7	7	100.00%
V4	F13	7	7	100.00%
V4	F14	7	7	100.00%
V4	F15	2	2	100.00%
V4	F16	7	7	100.00%
V4	F17	7	7	100.00%
V4	F18	6	7	85.71%
V4	F19	7	7	100.00%
V4	F20	7	7	100.00%
V5	F1	6	6	100.00%

Video	*Frame*	Benar	Total	Akurasi
V5	F2	6	6	100.00%
V5	F3	6	6	100.00%
V5	F4	6	6	100.00%
V5	F5	6	6	100.00%
V5	F6	7	7	100.00%
V5	F7	4	4	100.00%
V5	F8	4	4	100.00%
V5	F9	7	7	100.00%
V5	F10	6	6	100.00%
V5	F11	6	6	100.00%
V5	F12	4	6	66.67%
V5	F13	6	7	85.71%
V5	F14	6	7	85.71%
V5	F15	6	7	85.71%
V5	F16	6	7	85.71%
V5	F17	6	6	100.00%
V5	F18	6	6	100.00%
V5	F19	5	5	100.00%
V5	F20	6	6	100.00%
V6	F1	6	7	85.71%
V6	F2	7	7	100.00%
V6	F3	7	7	100.00%
V6	F4	7	7	100.00%
V6	F5	6	7	85.71%
V6	F6	7	7	100.00%
V6	F7	7	7	100.00%
V6	F8	7	7	100.00%
V6	F9	6	7	85.71%

Video	*Frame*	Benar	Total	Akurasi
V6	F10	7	7	100.00%
V6	F11	7	7	100.00%
V6	F12	7	7	100.00%
V6	F13	7	7	100.00%
V6	F14	7	7	100.00%
V6	F15	7	7	100.00%
V6	F16	7	7	100.00%
V6	F17	7	7	100.00%
V6	F18	7	7	100.00%
V6	F19	6	6	100.00%
V7	F1	0	0	100.00%
V7	F2	6	6	100.00%
V7	F3	6	7	85.71%
V7	F4	7	7	100.00%
V7	F5	7	7	100.00%
V7	F6	6	7	85.71%
V7	F7	4	5	80.00%
V7	F8	7	7	100.00%
V7	F9	7	7	100.00%
V7	F10	6	6	100.00%
V7	F11	7	7	100.00%
V7	F12	7	7	100.00%
V7	F13	7	7	100.00%
V7	F14	7	7	100.00%
V7	F15	2	2	100.00%
V7	F16	7	7	100.00%
V7	F17	7	7	100.00%
V7	F18	7	7	100.00%

Video	*Frame*	Benar	Total	Akurasi
V7	F19	7	7	100.00%
V7	F20	6	7	85.71%
V8	F1	6	7	85.71%
V8	F2	6	6	100.00%
V8	F3	6	7	85.71%
V8	F4	6	7	85.71%
V8	F5	6	6	100.00%
V8	F6	6	7	85.71%
V8	F7	7	7	100.00%
V8	F8	4	4	100.00%
V8	F9	6	7	85.71%
V8	F10	7	7	100.00%
V9	F1	0	0	100.00%
V9	F2	8	8	100.00%
V9	F3	8	8	100.00%
V9	F4	8	8	100.00%
V9	F5	8	8	100.00%
V9	F6	8	8	100.00%
V9	F7	8	8	100.00%
V9	F8	8	8	100.00%
V9	F9	8	8	100.00%
V9	F10	8	8	100.00%
V9	F11	8	8	100.00%
V9	F12	8	8	100.00%
V9	F13	8	8	100.00%
V9	F14	8	8	100.00%
V9	F15	8	8	100.00%
V9	F16	8	8	100.00%

Video	*Frame*	Benar	Total	Akurasi
V9	F17	8	8	100.00%
V9	F18	8	8	100.00%
V9	F19	8	8	100.00%
V9	F20	8	8	100.00%
V10	F1	1	1	100.00%
V10	F2	7	7	100.00%
V10	F3	7	7	100.00%
V10	F4	7	7	100.00%
V10	F5	7	7	100.00%
V10	F6	7	7	100.00%
V10	F7	5	5	100.00%
V10	F8	7	7	100.00%
V10	F9	7	7	100.00%
V10	F10	7	7	100.00%
V10	F11	7	7	100.00%
V10	F12	7	7	100.00%
V10	F13	7	7	100.00%
V10	F14	7	7	100.00%
V10	F15	7	7	100.00%
V10	F16	7	7	100.00%
V10	F17	7	7	100.00%
V10	F18	7	7	100.00%
V10	F19	6	6	100.00%
V10	F20	7	7	100.00%

L.25 Hasil Uji Coba Lokalisasi pada Video Sore

Video	*Frame*	Cakupan	Total	Akurasi
Video 1	*Frame* 1	9	9	100.00%
Video 1	*Frame* 2	8	9	88.89%
Video 1	*Frame* 3	9	9	100.00%
Video 1	*Frame* 4	9	9	100.00%
Video 1	*Frame* 5	9	9	100.00%
Video 1	*Frame* 6	9	9	100.00%
Video 1	*Frame* 7	9	9	100.00%
Video 1	*Frame* 8	8	9	88.89%
Video 1	*Frame* 9	9	9	100.00%
Video 1	*Frame* 10	9	9	100.00%
Video 1	*Frame* 11	9	9	100.00%
Video 1	*Frame* 12	8	9	88.89%
Video 1	*Frame* 13	9	9	100.00%
Video 1	*Frame* 14	8	9	88.89%
Video 1	*Frame* 15	9	9	100.00%
Video 1	*Frame* 16	8	9	88.89%
Video 1	*Frame* 17	9	9	100.00%
Video 1	*Frame* 18	9	9	100.00%
Video 1	*Frame* 19	9	9	100.00%
Video 1	*Frame* 20	9	9	100.00%
Video 2	*Frame* 1	7	7	100.00%
Video 2	*Frame* 2	7	7	100.00%
Video 2	*Frame* 3	7	7	100.00%
Video 2	*Frame* 4	7	7	100.00%
Video 2	*Frame* 5	7	7	100.00%

Video	*Frame*	Cakupan	Total	Akurasi
Video 2	*Frame* 6	7	7	100.00%
Video 2	*Frame* 7	7	7	100.00%
Video 2	*Frame* 8	7	7	100.00%
Video 2	*Frame* 9	7	7	100.00%
Video 2	*Frame* 10	7	7	100.00%
Video 2	*Frame* 11	7	7	100.00%
Video 2	*Frame* 12	7	7	100.00%
Video 2	*Frame* 13	7	7	100.00%
Video 2	*Frame* 14	7	7	100.00%
Video 2	*Frame* 15	7	7	100.00%
Video 3	*Frame* 1	7	7	100.00%
Video 3	*Frame* 2	7	7	100.00%
Video 3	*Frame* 3	7	7	100.00%
Video 3	*Frame* 4	7	7	100.00%
Video 3	*Frame* 5	7	7	100.00%
Video 3	*Frame* 6	7	7	100.00%
Video 3	*Frame* 7	7	7	100.00%
Video 3	*Frame* 8	7	7	100.00%
Video 3	*Frame* 9	7	7	100.00%
Video 3	*Frame* 10	7	7	100.00%
Video 3	*Frame* 11	7	7	100.00%
Video 3	*Frame* 12	7	7	100.00%
Video 3	*Frame* 13	7	7	100.00%
Video 3	*Frame* 14	7	7	100.00%
Video 3	*Frame* 15	7	7	100.00%
Video 3	*Frame* 16	7	7	100.00%
Video 3	*Frame* 17	7	7	100.00%
Video 3	*Frame* 18	7	7	100.00%

Video	*Frame*	Cakupan	Total	Akurasi
Video 3	*Frame* 19	7	7	100.00%
Video 4	*Frame* 1	7	7	100.00%
Video 4	*Frame* 2	7	7	100.00%
Video 4	*Frame* 3	7	7	100.00%
Video 4	*Frame* 4	7	7	100.00%
Video 4	*Frame* 5	7	7	100.00%
Video 4	*Frame* 6	7	7	100.00%
Video 4	*Frame* 7	7	7	100.00%
Video 4	*Frame* 8	7	7	100.00%
Video 4	*Frame* 9	7	7	100.00%
Video 4	*Frame* 10	7	7	100.00%
Video 4	*Frame* 11	7	7	100.00%
Video 4	*Frame* 12	7	7	100.00%
Video 4	*Frame* 13	7	7	100.00%
Video 4	*Frame* 14	7	7	100.00%
Video 4	*Frame* 15	7	7	100.00%
Video 4	*Frame* 16	7	7	100.00%
Video 4	*Frame* 17	7	7	100.00%
Video 4	*Frame* 18	7	7	100.00%
Video 4	*Frame* 19	7	7	100.00%
Video 4	*Frame* 20	7	7	100.00%
Video 5	*Frame* 1	7	7	100.00%
Video 5	*Frame* 2	7	7	100.00%
Video 5	*Frame* 3	7	7	100.00%
Video 5	*Frame* 4	7	7	100.00%
Video 5	*Frame* 5	7	7	100.00%
Video 5	*Frame* 6	7	7	100.00%
Video 5	*Frame* 7	6	7	85.71%

Video	*Frame*	Cakupan	Total	Akurasi
Video 5	*Frame* 8	7	7	100.00%
Video 5	*Frame* 9	7	7	100.00%
Video 5	*Frame* 10	7	7	100.00%
Video 5	*Frame* 11	6	7	85.71%
Video 5	*Frame* 12	7	7	100.00%
Video 5	*Frame* 13	7	7	100.00%
Video 5	*Frame* 14	7	7	100.00%
Video 5	*Frame* 15	7	7	100.00%
Video 5	*Frame* 16	7	7	100.00%
Video 5	*Frame* 17	7	7	100.00%
Video 5	*Frame* 18	7	7	100.00%
Video 5	*Frame* 19	7	7	100.00%
Video 5	*Frame* 20	6	7	85.71%
Video 6	*Frame* 1	7	8	87.50%
Video 6	*Frame* 2	7	8	87.50%
Video 6	*Frame* 3	8	8	100.00%
Video 6	*Frame* 4	8	8	100.00%
Video 6	*Frame* 5	7	8	87.50%
Video 6	*Frame* 6	7	8	87.50%
Video 6	*Frame* 7	7	8	87.50%
Video 6	*Frame* 8	7	8	87.50%
Video 6	*Frame* 9	7	8	87.50%
Video 6	*Frame* 10	7	8	87.50%
Video 6	*Frame* 11	8	8	100.00%
Video 6	*Frame* 12	8	8	100.00%
Video 6	*Frame* 13	8	8	100.00%
Video 6	*Frame* 14	8	8	100.00%
Video 6	*Frame* 15	8	8	100.00%

Video	*Frame*	Cakupan	Total	Akurasi
Video 6	*Frame* 16	8	8	100.00%
Video 6	*Frame* 17	8	8	100.00%
Video 6	*Frame* 18	8	8	100.00%
Video 6	*Frame* 19	8	8	100.00%
Video 6	*Frame* 20	8	8	100.00%
Video 7	*Frame* 1	0	7	0.00%
Video 7	*Frame* 2	7	7	100.00%
Video 7	*Frame* 3	7	7	100.00%
Video 7	*Frame* 4	0	7	0.00%
Video 7	*Frame* 5	6	7	85.71%
Video 7	*Frame* 6	6	7	85.71%
Video 7	*Frame* 7	0	7	0.00%
Video 7	*Frame* 8	7	7	100.00%
Video 7	*Frame* 9	7	7	100.00%
Video 7	*Frame* 10	6	7	85.71%
Video 7	*Frame* 11	6	7	85.71%
Video 7	*Frame* 12	6	7	85.71%
Video 7	*Frame* 13	6	7	85.71%
Video 7	*Frame* 14	6	7	85.71%
Video 7	*Frame* 15	7	7	100.00%
Video 7	*Frame* 16	7	7	100.00%
Video 7	*Frame* 17	6	7	85.71%
Video 7	*Frame* 18	0	7	0.00%
Video 7	*Frame* 19	6	7	85.71%
Video 7	*Frame* 20	0	7	0.00%
Video 8	*Frame* 1	6	7	85.71%
Video 8	*Frame* 2	6	7	85.71%
Video 8	*Frame* 3	6	7	85.71%

Video	*Frame*	Cakupan	Total	Akurasi
Video 8	*Frame* 4	7	7	100.00%
Video 8	*Frame* 5	7	7	100.00%
Video 8	*Frame* 6	6	7	85.71%
Video 8	*Frame* 7	6	7	85.71%
Video 8	*Frame* 8	6	7	85.71%
Video 8	*Frame* 9	6	7	85.71%
Video 8	*Frame* 10	6	7	85.71%
Video 8	*Frame* 11	7	7	100.00%
Video 8	*Frame* 12	7	7	100.00%
Video 8	*Frame* 13	7	7	100.00%
Video 8	*Frame* 14	7	7	100.00%
Video 8	*Frame* 15	6	7	85.71%
Video 8	*Frame* 16	7	7	100.00%
Video 8	*Frame* 17	7	7	100.00%
Video 8	*Frame* 18	7	7	100.00%
Video 8	*Frame* 19	6	7	85.71%
Video 8	*Frame* 20	6	7	85.71%
Video 9	*Frame* 1	7	7	100.00%
Video 9	*Frame* 2	7	7	100.00%
Video 9	*Frame* 3	7	7	100.00%
Video 9	*Frame* 4	7	7	100.00%
Video 9	*Frame* 5	7	7	100.00%
Video 9	*Frame* 6	7	7	100.00%
Video 9	*Frame* 7	7	7	100.00%
Video 9	*Frame* 8	7	7	100.00%
Video 9	*Frame* 9	7	7	100.00%
Video 9	*Frame* 10	7	7	100.00%
Video 9	*Frame* 11	7	7	100.00%

Video	*Frame*	Cakupan	Total	Akurasi
Video 9	*Frame* 12	7	7	100.00%
Video 9	*Frame* 13	7	7	100.00%
Video 9	*Frame* 14	7	7	100.00%
Video 9	*Frame* 15	7	7	100.00%
Video 9	*Frame* 16	7	7	100.00%
Video 9	*Frame* 17	7	7	100.00%
Video 9	*Frame* 18	7	7	100.00%
Video 9	*Frame* 19	7	7	100.00%
Video 9	*Frame* 20	7	7	100.00%
Video 10	*Frame* 1	0	7	0.00%
Video 10	*Frame* 2	6	7	85.71%
Video 10	*Frame* 3	7	7	100.00%
Video 10	*Frame* 4	6	7	85.71%
Video 10	*Frame* 5	7	7	100.00%
Video 10	*Frame* 6	7	7	100.00%
Video 10	*Frame* 7	7	7	100.00%
Video 10	*Frame* 8	7	7	100.00%
Video 10	*Frame* 9	7	7	100.00%
Video 10	*Frame* 10	7	7	100.00%
Video 10	*Frame* 11	7	7	100.00%
Video 10	*Frame* 12	7	7	100.00%
Video 10	*Frame* 13	6	7	85.71%
Video 10	*Frame* 14	6	7	85.71%
Video 10	*Frame* 15	6	7	85.71%
Video 10	*Frame* 16	7	7	100.00%
Video 10	*Frame* 17	6	7	85.71%
Video 10	*Frame* 18	6	7	85.71%
Video 10	*Frame* 19	6	7	85.71%

Video	*Frame*	Cakupan	Total	Akurasi
Video 10	*Frame* 20	7	7	100.00%

L.26 Hasil Uji Coba Segmentasi pada Video Sore

Video	*Frame*	TP	FP	FN	TN	Akurasi	Prec.	*Recall*
V1	F1	0	1	9	0	0.00%	0.00%	0.00%
V1	F2	3	0	6	0	33.33%	100.00%	33.33%
V1	F3	3	0	6	0	33.33%	100.00%	33.33%
V1	F4	4	1	5	0	40.00%	80.00%	44.44%
V1	F5	3	0	6	0	33.33%	100.00%	33.33%
V1	F6	7	2	2	0	63.64%	77.78%	77.78%
V1	F7	0	0	9	0	0.00%	100.00%	0.00%
V1	F8	2	0	7	0	22.22%	100.00%	22.22%
V1	F9	9	1	0	0	90.00%	90.00%	100.00%
V1	F10	9	0	0	0	100.00%	100.00%	100.00%
V1	F11	9	0	0	0	100.00%	100.00%	100.00%
V1	F12	8	0	0	0	100.00%	100.00%	100.00%
V1	F13	6	0	3	0	66.67%	100.00%	66.67%
V1	F14	8	0	0	0	100.00%	100.00%	100.00%
V1	F15	0	0	9	0	0.00%	100.00%	0.00%
V1	F16	8	1	0	0	88.89%	88.89%	100.00%
V1	F17	5	0	4	0	55.56%	100.00%	55.56%
V1	F18	9	0	0	0	100.00%	100.00%	100.00%
V1	F19	3	1	6	0	30.00%	75.00%	33.33%
V1	F20	4	1	6	0	36.36%	80.00%	40.00%
V2	F1	3	1	4	0	37.50%	75.00%	42.86%
V2	F2	4	0	3	0	57.14%	100.00%	57.14%
V2	F3	7	1	0	0	87.50%	87.50%	100.00%

Video	*Frame*	TP	FP	FN	TN	Akurasi	Prec.	*Recall*
V2	F4	7	1	0	0	87.50%	87.50%	100.00%
V2	F5	7	2	0	0	77.78%	77.78%	100.00%
V2	F6	7	1	0	0	87.50%	87.50%	100.00%
V2	F7	7	2	0	0	77.78%	77.78%	100.00%
V2	F8	7	1	0	0	87.50%	87.50%	100.00%
V2	F9	7	1	0	0	87.50%	87.50%	100.00%
V2	F10	1	0	6	0	14.29%	100.00%	14.29%
V2	F11	7	1	0	0	87.50%	87.50%	100.00%
V2	F12	7	0	0	0	100.00%	100.00%	100.00%
V2	F13	7	1	0	0	87.50%	87.50%	100.00%
V2	F14	7	0	0	0	100.00%	100.00%	100.00%
V2	F15	7	1	0	0	87.50%	87.50%	100.00%
V3	F1	2	0	5	0	28.57%	100.00%	28.57%
V3	F2	6	0	2	0	75.00%	100.00%	75.00%
V3	F3	7	0	0	0	100.00%	100.00%	100.00%
V3	F4	7	3	0	0	70.00%	70.00%	100.00%
V3	F5	7	1	0	0	87.50%	87.50%	100.00%
V3	F6	6	1	1	0	75.00%	85.71%	85.71%
V3	F7	5	0	2	0	71.43%	100.00%	71.43%
V3	F8	7	1	0	0	87.50%	87.50%	100.00%
V3	F9	7	0	0	0	100.00%	100.00%	100.00%
V3	F10	7	2	0	0	77.78%	77.78%	100.00%
V3	F11	7	1	0	0	87.50%	87.50%	100.00%
V3	F12	7	0	0	0	100.00%	100.00%	100.00%
V3	F13	7	1	0	0	87.50%	87.50%	100.00%
V3	F14	3	0	4	0	42.86%	100.00%	42.86%
V3	F15	7	1	0	0	87.50%	87.50%	100.00%
V3	F16	7	0	0	0	100.00%	100.00%	100.00%

Video	*Frame*	TP	FP	FN	TN	Akurasi	Prec.	*Recall*
V3	F17	6	1	1	0	75.00%	85.71%	85.71%
V3	F18	6	1	1	0	75.00%	85.71%	85.71%
V3	F19	6	0	1	0	85.71%	100.00%	85.71%
V4	F1	7	0	0	0	100.00%	100.00%	100.00%
V4	F2	7	0	0	0	100.00%	100.00%	100.00%
V4	F3	7	0	0	0	100.00%	100.00%	100.00%
V4	F4	7	0	0	0	100.00%	100.00%	100.00%
V4	F5	7	0	0	0	100.00%	100.00%	100.00%
V4	F6	7	0	0	0	100.00%	100.00%	100.00%
V4	F7	7	0	0	0	100.00%	100.00%	100.00%
V4	F8	6	0	1	0	85.71%	100.00%	85.71%
V4	F9	7	0	0	0	100.00%	100.00%	100.00%
V4	F10	7	0	0	0	100.00%	100.00%	100.00%
V4	F11	7	0	0	0	100.00%	100.00%	100.00%
V4	F12	7	0	0	0	100.00%	100.00%	100.00%
V4	F13	7	0	0	0	100.00%	100.00%	100.00%
V4	F14	7	0	0	0	100.00%	100.00%	100.00%
V4	F15	7	0	0	0	100.00%	100.00%	100.00%
V4	F16	7	0	0	0	100.00%	100.00%	100.00%
V4	F17	7	0	0	0	100.00%	100.00%	100.00%
V4	F18	7	0	0	0	100.00%	100.00%	100.00%
V4	F19	7	0	0	0	100.00%	100.00%	100.00%
V4	F20	7	0	0	0	100.00%	100.00%	100.00%
V5	F1	6	0	1	0	85.71%	100.00%	85.71%
V5	F2	6	0	1	0	85.71%	100.00%	85.71%
V5	F3	6	1	1	0	75.00%	85.71%	85.71%
V5	F4	6	1	1	0	75.00%	85.71%	85.71%
V5	F5	6	2	1	0	66.67%	75.00%	85.71%

Video	*Frame*	TP	FP	FN	TN	Akurasi	Prec.	*Recall*
V5	F6	5	2	2	0	55.56%	71.43%	71.43%
V5	F7	6	0	1	0	85.71%	100.00%	85.71%
V5	F8	6	0	1	0	85.71%	100.00%	85.71%
V5	F9	6	2	1	0	66.67%	75.00%	85.71%
V5	F10	6	0	1	0	85.71%	100.00%	85.71%
V5	F11	6	1	1	0	75.00%	85.71%	85.71%
V5	F12	6	1	1	0	75.00%	85.71%	85.71%
V5	F13	6	1	1	0	75.00%	85.71%	85.71%
V5	F14	6	1	1	0	75.00%	85.71%	85.71%
V5	F15	6	0	1	0	85.71%	100.00%	85.71%
V5	F16	6	2	1	0	66.67%	75.00%	85.71%
V5	F17	6	0	1	0	85.71%	100.00%	85.71%
V5	F18	6	1	1	0	75.00%	85.71%	85.71%
V5	F19	6	1	1	0	75.00%	85.71%	85.71%
V5	F20	6	1	1	0	75.00%	85.71%	85.71%
V6	F1	8	0	0	0	100.00%	100.00%	100.00%
V6	F2	8	3	0	0	72.73%	72.73%	100.00%
V6	F3	8	0	0	0	100.00%	100.00%	100.00%
V6	F4	7	0	1	0	87.50%	100.00%	87.50%
V6	F5	5	1	3	0	55.56%	83.33%	62.50%
V6	F6	6	1	2	0	66.67%	85.71%	75.00%
V6	F7	8	0	0	0	100.00%	100.00%	100.00%
V6	F8	8	0	0	0	100.00%	100.00%	100.00%
V6	F9	8	0	0	0	100.00%	100.00%	100.00%
V6	F10	8	0	0	0	100.00%	100.00%	100.00%
V6	F11	8	0	0	0	100.00%	100.00%	100.00%
V6	F12	8	0	0	0	100.00%	100.00%	100.00%
V6	F13	8	0	0	0	100.00%	100.00%	100.00%

Video	*Frame*	TP	FP	FN	TN	Akurasi	Prec.	*Recall*
V6	F14	8	0	0	0	100.00%	100.00%	100.00%
V6	F15	8	0	0	0	100.00%	100.00%	100.00%
V6	F16	7	0	1	0	87.50%	100.00%	87.50%
V6	F17	6	0	2	0	75.00%	100.00%	75.00%
V6	F18	7	0	1	0	87.50%	100.00%	87.50%
V6	F19	7	0	1	0	87.50%	100.00%	87.50%
V6	F20	8	1	0	0	88.89%	88.89%	100.00%
V7	F1	7	1	0	0	87.50%	87.50%	100.00%
V7	F2	6	1	0	0	85.71%	85.71%	100.00%
V7	F3	6	1	0	0	85.71%	85.71%	100.00%
V7	F4	2	2	5	0	22.22%	50.00%	28.57%
V7	F5	5	3	2	0	50.00%	62.50%	71.43%
V7	F6	5	0	1	0	83.33%	100.00%	83.33%
V7	F7	7	1	0	0	87.50%	87.50%	100.00%
V7	F8	7	2	0	0	77.78%	77.78%	100.00%
V7	F9	7	0	0	0	100.00%	100.00%	100.00%
V7	F10	1	2	6	0	11.11%	33.33%	14.29%
V7	F11	6	0	0	0	100.00%	100.00%	100.00%
V7	F12	6	0	0	0	100.00%	100.00%	100.00%
V7	F13	7	0	0	0	100.00%	100.00%	100.00%
V7	F14	6	0	0	0	100.00%	100.00%	100.00%
V7	F15	7	1	0	0	87.50%	87.50%	100.00%
V8	F1	7	0	0	0	100.00%	100.00%	100.00%
V8	F2	7	0	0	0	100.00%	100.00%	100.00%
V8	F3	7	0	0	0	100.00%	100.00%	100.00%
V8	F4	7	0	0	0	100.00%	100.00%	100.00%
V8	F5	7	0	0	0	100.00%	100.00%	100.00%
V8	F6	7	0	0	0	100.00%	100.00%	100.00%

Video	*Frame*	TP	FP	FN	TN	Akurasi	Prec.	*Recall*
V8	F7	7	0	0	0	100.00%	100.00%	100.00%
V8	F8	7	0	0	0	100.00%	100.00%	100.00%
V8	F9	7	0	0	0	100.00%	100.00%	100.00%
V8	F10	6	0	0	0	100.00%	100.00%	100.00%
V8	F11	7	0	0	0	100.00%	100.00%	100.00%
V8	F12	7	0	0	0	100.00%	100.00%	100.00%
V8	F13	7	0	0	0	100.00%	100.00%	100.00%
V8	F14	7	0	0	0	100.00%	100.00%	100.00%
V8	F15	6	0	0	0	100.00%	100.00%	100.00%
V8	F16	7	0	0	0	100.00%	100.00%	100.00%
V8	F17	7	0	0	0	100.00%	100.00%	100.00%
V8	F18	7	0	0	0	100.00%	100.00%	100.00%
V8	F19	7	0	0	0	100.00%	100.00%	100.00%
V8	F20	7	0	0	0	100.00%	100.00%	100.00%
V9	F1	7	0	0	0	100.00%	100.00%	100.00%
V9	F2	7	1	0	0	87.50%	87.50%	100.00%
V9	F3	7	1	0	0	87.50%	87.50%	100.00%
V9	F4	7	0	0	0	100.00%	100.00%	100.00%
V9	F5	7	1	0	0	87.50%	87.50%	100.00%
V9	F6	7	0	0	0	100.00%	100.00%	100.00%
V9	F7	7	0	0	0	100.00%	100.00%	100.00%
V9	F8	7	0	0	0	100.00%	100.00%	100.00%
V9	F9	7	0	0	0	100.00%	100.00%	100.00%
V9	F10	7	0	0	0	100.00%	100.00%	100.00%
V9	F11	7	0	0	0	100.00%	100.00%	100.00%
V9	F12	7	0	0	0	100.00%	100.00%	100.00%
V9	F13	7	0	0	0	100.00%	100.00%	100.00%
V9	F14	7	0	0	0	100.00%	100.00%	100.00%

Video	***Frame***	**TP**	**FP**	**FN**	**TN**	**Akurasi**	**Prec.**	***Recall***
V9	F15	7	0	0	0	100.00%	100.00%	100.00%
V9	F16	7	1	0	0	87.50%	87.50%	100.00%
V9	F17	7	1	0	0	87.50%	87.50%	100.00%
V9	F18	7	1	0	0	87.50%	87.50%	100.00%
V9	F19	7	2	0	0	77.78%	77.78%	100.00%
V9	F20	7	1	0	0	87.50%	87.50%	100.00%
V10	F1	6	0	0	0	100.00%	100.00%	100.00%
V10	F2	7	0	0	0	100.00%	100.00%	100.00%
V10	F3	7	0	0	0	100.00%	100.00%	100.00%
V10	F4	7	0	0	0	100.00%	100.00%	100.00%
V10	F5	2	0	5	0	28.57%	100.00%	28.57%
V10	F6	7	3	0	0	70.00%	70.00%	100.00%
V10	F7	7	4	0	0	63.64%	63.64%	100.00%
V10	F8	7	1	0	0	87.50%	87.50%	100.00%
V10	F9	7	1	0	0	87.50%	87.50%	100.00%
V10	F10	7	0	0	0	100.00%	100.00%	100.00%
V10	F11	7	0	0	0	100.00%	100.00%	100.00%
V10	F12	6	0	0	0	100.00%	100.00%	100.00%
V10	F13	6	0	0	0	100.00%	100.00%	100.00%
V10	F14	5	2	1	0	62.50%	71.43%	83.33%
V10	F15	7	0	0	0	100.00%	100.00%	100.00%
V10	F16	6	0	0	0	100.00%	100.00%	100.00%
V10	F17	7	0	0	0	100.00%	100.00%	100.00%
V10	F18	7	0	0	0	100.00%	100.00%	100.00%
V10	F19	7	0	0	0	100.00%	100.00%	100.00%

L.27 Hasil Uji Coba Majority Vote pada Video Sore

Vid.	*Majority Vote*	TP	FP	FN	TN	Akurasi	Prec.	*Recall*
V1	AG 3192 RA	8	0	1	0	88.89%	100.00%	88.89%
V2	NN 5694 HZ	7	1	0	0	87.50%	87.50%	100.00%
V3	L 3669 TT	7	0	0	0	100.00%	100.00%	100.00%
V4	W 3440 WJ	7	0	0	0	100.00%	100.00%	100.00%
V5	L 3673 X	6	0	1	0	85.71%	100.00%	85.71%
V6	AG 2351 F1	7	1	0	0	87.50%	87.50%	100.00%
V7	H 6042 S	6	0	1	0	85.71%	100.00%	85.71%
V8	P 2083 WN	7	0	0	0	100.00%	100.00%	100.00%
V9	M 2754 HK	7	0	0	0	100.00%	100.00%	100.00%
V10	P 5129 KU	7	0	0	0	100.00%	100.00%	100.00%

L.28 Hasil Uji Coba Pengenalan Karakter pada Video Sore

Video	*Frame*	Benar	Total	Akurasi
V1	F1	0	0	100.00%
V1	F2	3	3	100.00%
V1	F3	3	3	100.00%
V1	F4	3	4	75.00%
V1	F5	3	3	100.00%
V1	F6	5	7	71.43%
V1	F7	0	0	100.00%
V1	F8	2	2	100.00%
V1	F9	9	9	100.00%
V1	F10	9	9	100.00%
V1	F11	9	9	100.00%
V1	F12	8	8	100.00%

Video	*Frame*	Benar	Total	Akurasi
V1	F13	5	6	83.33%
V1	F14	8	8	100.00%
V1	F15	0	0	100.00%
V1	F16	8	8	100.00%
V1	F17	4	5	80.00%
V1	F18	7	9	77.78%
V1	F19	2	3	66.67%
V1	F20	3	4	75.00%
V2	F1	2	3	66.67%
V2	F2	2	4	50.00%
V2	F3	7	7	100.00%
V2	F4	7	7	100.00%
V2	F5	7	7	100.00%
V2	F6	7	7	100.00%
V2	F7	7	7	100.00%
V2	F8	7	7	100.00%
V2	F9	7	7	100.00%
V2	F10	1	1	100.00%
V2	F11	7	7	100.00%
V2	F12	7	7	100.00%
V2	F13	7	7	100.00%
V2	F14	6	7	85.71%
V2	F15	7	7	100.00%
V3	F1	2	2	100.00%
V3	F2	6	6	100.00%
V3	F3	7	7	100.00%
V3	F4	7	7	100.00%
V3	F5	7	7	100.00%

Video	*Frame*	Benar	Total	Akurasi
V3	F6	6	6	100.00%
V3	F7	5	5	100.00%
V3	F8	7	7	100.00%
V3	F9	7	7	100.00%
V3	F10	7	7	100.00%
V3	F11	7	7	100.00%
V3	F12	7	7	100.00%
V3	F13	7	7	100.00%
V3	F14	3	3	100.00%
V3	F15	7	7	100.00%
V3	F16	7	7	100.00%
V3	F17	6	6	100.00%
V3	F18	6	6	100.00%
V3	F19	6	6	100.00%
V4	F1	7	7	100.00%
V4	F2	7	7	100.00%
V4	F3	7	7	100.00%
V4	F4	7	7	100.00%
V4	F5	7	7	100.00%
V4	F6	7	7	100.00%
V4	F7	7	7	100.00%
V4	F8	6	6	100.00%
V4	F9	7	7	100.00%
V4	F10	7	7	100.00%
V4	F11	7	7	100.00%
V4	F12	7	7	100.00%
V4	F13	7	7	100.00%
V4	F14	7	7	100.00%

Video	*Frame*	Benar	Total	Akurasi
V4	F15	7	7	100.00%
V4	F16	7	7	100.00%
V4	F17	7	7	100.00%
V4	F18	7	7	100.00%
V4	F19	7	7	100.00%
V4	F20	7	7	100.00%
V5	F1	6	6	100.00%
V5	F2	6	6	100.00%
V5	F3	6	6	100.00%
V5	F4	6	6	100.00%
V5	F5	5	6	83.33%
V5	F6	4	5	80.00%
V5	F7	6	6	100.00%
V5	F8	5	6	83.33%
V5	F9	6	6	100.00%
V5	F10	5	6	83.33%
V5	F11	5	6	83.33%
V5	F12	6	6	100.00%
V5	F13	6	6	100.00%
V5	F14	6	6	100.00%
V5	F15	6	6	100.00%
V5	F16	6	6	100.00%
V5	F17	6	6	100.00%
V5	F18	5	6	83.33%
V5	F19	6	6	100.00%
V5	F20	5	6	83.33%
V6	F1	7	8	87.50%
V6	F2	8	8	100.00%

Video	*Frame*	Benar	Total	Akurasi
V6	F3	8	8	100.00%
V6	F4	7	7	100.00%
V6	F5	4	5	80.00%
V6	F6	6	6	100.00%
V6	F7	8	8	100.00%
V6	F8	8	8	100.00%
V6	F9	7	8	87.50%
V6	F10	7	8	87.50%
V6	F11	7	8	87.50%
V6	F12	7	8	87.50%
V6	F13	7	8	87.50%
V6	F14	7	8	87.50%
V6	F15	7	8	87.50%
V6	F16	7	7	100.00%
V6	F17	6	6	100.00%
V6	F18	7	7	100.00%
V6	F19	7	7	100.00%
V6	F20	8	8	100.00%
V7	F1	7	7	100.00%
V7	F2	6	6	100.00%
V7	F3	6	6	100.00%
V7	F4	2	2	100.00%
V7	F5	3	5	60.00%
V7	F6	5	5	100.00%
V7	F7	7	7	100.00%
V7	F8	7	7	100.00%
V7	F9	6	7	85.71%
V7	F10	1	1	100.00%

Video	*Frame*	Benar	Total	Akurasi
V7	F11	6	6	100.00%
V7	F12	6	6	100.00%
V7	F13	6	7	85.71%
V7	F14	5	6	83.33%
V7	F15	7	7	100.00%
V8	F1	6	7	85.71%
V8	F2	6	7	85.71%
V8	F3	6	7	85.71%
V8	F4	7	7	100.00%
V8	F5	7	7	100.00%
V8	F6	6	7	85.71%
V8	F7	6	7	85.71%
V8	F8	6	7	85.71%
V8	F9	6	7	85.71%
V8	F10	6	6	100.00%
V8	F11	6	7	85.71%
V8	F12	7	7	100.00%
V8	F13	7	7	100.00%
V8	F14	7	7	100.00%
V8	F15	6	6	100.00%
V8	F16	7	7	100.00%
V8	F17	6	7	85.71%
V8	F18	7	7	100.00%
V8	F19	6	7	85.71%
V8	F20	6	7	85.71%
V9	F1	7	7	100.00%
V9	F2	7	7	100.00%
V9	F3	7	7	100.00%

Video	*Frame*	Benar	Total	Akurasi
V9	F4	7	7	100.00%
V9	F5	7	7	100.00%
V9	F6	7	7	100.00%
V9	F7	7	7	100.00%
V9	F8	7	7	100.00%
V9	F9	7	7	100.00%
V9	F10	7	7	100.00%
V9	F11	7	7	100.00%
V9	F12	7	7	100.00%
V9	F13	7	7	100.00%
V9	F14	7	7	100.00%
V9	F15	7	7	100.00%
V9	F16	7	7	100.00%
V9	F17	7	7	100.00%
V9	F18	7	7	100.00%
V9	F19	7	7	100.00%
V9	F20	7	7	100.00%
V10	F1	6	6	100.00%
V10	F2	6	7	85.71%
V10	F3	6	7	85.71%
V10	F4	6	7	85.71%
V10	F5	0	2	0.00%
V10	F6	5	7	71.43%
V10	F7	7	7	100.00%
V10	F8	7	7	100.00%
V10	F9	6	7	85.71%
V10	F10	7	7	100.00%
V10	F11	7	7	100.00%

Video	*Frame*	Benar	Total	Akurasi
V10	F12	6	6	100.00%
V10	F13	6	6	100.00%
V10	F14	2	5	40.00%
V10	F15	7	7	100.00%
V10	F16	6	6	100.00%
V10	F17	6	7	85.71%
V10	F18	6	7	85.71%
V10	F19	7	7	100.00%

L.29 Hasil Uji Coba Lokalisasi pada Video Malam

Video	*Frame*	Cakupan	Total	Akurasi
Video 1	*Frame* 1	0	7	0.00%
Video 1	*Frame* 2	0	7	0.00%
Video 1	*Frame* 3	6	7	85.71%
Video 1	*Frame* 4	6	7	85.71%
Video 1	*Frame* 5	6	7	85.71%
Video 1	*Frame* 6	6	7	85.71%
Video 1	*Frame* 7	6	7	85.71%
Video 1	*Frame* 8	6	7	85.71%
Video 1	*Frame* 9	5	7	71.43%
Video 1	*Frame* 10	6	7	85.71%
Video 1	*Frame* 11	6	7	85.71%
Video 1	*Frame* 12	5	7	71.43%
Video 1	*Frame* 13	6	7	85.71%
Video 1	*Frame* 14	6	7	85.71%
Video 1	*Frame* 15	0	7	0.00%
Video 1	*Frame* 16	6	7	85.71%

Video	*Frame*	Cakupan	Total	Akurasi
Video 1	*Frame* 17	6	7	85.71%
Video 1	*Frame* 18	0	7	0.00%
Video 1	*Frame* 19	0	7	0.00%
Video 1	*Frame* 20	0	7	0.00%
Video 2	*Frame* 1	8	8	100.00%
Video 2	*Frame* 2	8	8	100.00%
Video 2	*Frame* 3	8	8	100.00%
Video 2	*Frame* 4	8	8	100.00%
Video 2	*Frame* 5	8	8	100.00%
Video 2	*Frame* 6	8	8	100.00%
Video 2	*Frame* 7	8	8	100.00%
Video 2	*Frame* 8	8	8	100.00%
Video 2	*Frame* 9	8	8	100.00%
Video 2	*Frame* 10	8	8	100.00%
Video 2	*Frame* 11	8	8	100.00%
Video 2	*Frame* 12	8	8	100.00%
Video 2	*Frame* 13	8	8	100.00%
Video 2	*Frame* 14	8	8	100.00%
Video 2	*Frame* 15	8	8	100.00%
Video 2	*Frame* 16	8	8	100.00%
Video 2	*Frame* 17	8	8	100.00%
Video 2	*Frame* 18	8	8	100.00%
Video 2	*Frame* 19	8	8	100.00%
Video 2	*Frame* 20	8	8	100.00%
Video 3	*Frame* 1	0	7	0.00%
Video 3	*Frame* 2	7	7	100.00%
Video 3	*Frame* 3	7	7	100.00%
Video 3	*Frame* 4	7	7	100.00%

Video	*Frame*	Cakupan	Total	Akurasi
Video 3	*Frame* 5	7	7	100.00%
Video 3	*Frame* 6	7	7	100.00%
Video 3	*Frame* 7	7	7	100.00%
Video 3	*Frame* 8	7	7	100.00%
Video 3	*Frame* 9	7	7	100.00%
Video 3	*Frame* 10	7	7	100.00%
Video 3	*Frame* 11	7	7	100.00%
Video 3	*Frame* 12	7	7	100.00%
Video 3	*Frame* 13	7	7	100.00%
Video 3	*Frame* 14	7	7	100.00%
Video 3	*Frame* 15	7	7	100.00%
Video 3	*Frame* 16	7	7	100.00%
Video 3	*Frame* 17	7	7	100.00%
Video 3	*Frame* 18	7	7	100.00%
Video 3	*Frame* 19	7	7	100.00%
Video 3	*Frame* 20	7	7	100.00%
Video 4	*Frame* 1	8	8	100.00%
Video 4	*Frame* 2	8	8	100.00%
Video 4	*Frame* 3	8	8	100.00%
Video 4	*Frame* 4	8	8	100.00%
Video 4	*Frame* 5	8	8	100.00%
Video 4	*Frame* 6	8	8	100.00%
Video 4	*Frame* 7	8	8	100.00%
Video 4	*Frame* 8	8	8	100.00%
Video 4	*Frame* 9	8	8	100.00%
Video 4	*Frame* 10	8	8	100.00%
Video 4	*Frame* 11	8	8	100.00%
Video 4	*Frame* 12	8	8	100.00%

Video	*Frame*	Cakupan	Total	Akurasi
Video 4	*Frame* 13	8	8	100.00%
Video 4	*Frame* 14	8	8	100.00%
Video 4	*Frame* 15	8	8	100.00%
Video 4	*Frame* 16	8	8	100.00%
Video 4	*Frame* 17	8	8	100.00%
Video 4	*Frame* 18	8	8	100.00%
Video 4	*Frame* 19	8	8	100.00%
Video 4	*Frame* 20	8	8	100.00%
Video 5	*Frame* 1	6	7	85.71%
Video 5	*Frame* 2	6	7	85.71%
Video 5	*Frame* 3	6	7	85.71%
Video 5	*Frame* 4	6	7	85.71%
Video 5	*Frame* 5	6	7	85.71%
Video 5	*Frame* 6	6	7	85.71%
Video 5	*Frame* 7	6	7	85.71%
Video 5	*Frame* 8	6	7	85.71%
Video 5	*Frame* 9	6	7	85.71%
Video 5	*Frame* 10	6	7	85.71%
Video 5	*Frame* 11	6	7	85.71%
Video 5	*Frame* 12	6	7	85.71%
Video 5	*Frame* 13	6	7	85.71%
Video 5	*Frame* 14	6	7	85.71%
Video 5	*Frame* 15	6	7	85.71%
Video 5	*Frame* 16	6	7	85.71%
Video 5	*Frame* 17	6	7	85.71%
Video 5	*Frame* 18	6	7	85.71%
Video 5	*Frame* 19	6	7	85.71%
Video 5	*Frame* 20	6	7	85.71%

Video	*Frame*	Cakupan	Total	Akurasi
Video 6	*Frame* 1	0	9	0.00%
Video 6	*Frame* 2	0	9	0.00%
Video 6	*Frame* 3	0	9	0.00%
Video 6	*Frame* 4	0	9	0.00%
Video 6	*Frame* 5	0	9	0.00%
Video 6	*Frame* 6	0	9	0.00%
Video 6	*Frame* 7	9	9	100.00%
Video 6	*Frame* 8	9	9	100.00%
Video 6	*Frame* 9	9	9	100.00%
Video 6	*Frame* 10	9	9	100.00%
Video 6	*Frame* 11	9	9	100.00%
Video 6	*Frame* 12	9	9	100.00%
Video 6	*Frame* 13	9	9	100.00%
Video 6	*Frame* 14	9	9	100.00%
Video 6	*Frame* 15	9	9	100.00%
Video 6	*Frame* 16	9	9	100.00%
Video 6	*Frame* 17	9	9	100.00%
Video 6	*Frame* 18	9	9	100.00%
Video 6	*Frame* 19	9	9	100.00%
Video 6	*Frame* 20	9	9	100.00%
Video 7	*Frame* 1	8	8	100.00%
Video 7	*Frame* 2	7	8	87.50%
Video 7	*Frame* 3	7	8	87.50%
Video 7	*Frame* 4	7	8	87.50%
Video 7	*Frame* 5	7	8	87.50%
Video 7	*Frame* 6	7	8	87.50%
Video 7	*Frame* 7	8	8	100.00%
Video 7	*Frame* 8	8	8	100.00%

Video	*Frame*	Cakupan	Total	Akurasi
Video 7	*Frame* 9	8	8	100.00%
Video 7	*Frame* 10	8	8	100.00%
Video 7	*Frame* 11	8	8	100.00%
Video 7	*Frame* 12	8	8	100.00%
Video 7	*Frame* 13	8	8	100.00%
Video 7	*Frame* 14	7	8	87.50%
Video 7	*Frame* 15	8	8	100.00%
Video 7	*Frame* 16	7	8	87.50%
Video 7	*Frame* 17	7	8	87.50%
Video 7	*Frame* 18	7	8	87.50%
Video 7	*Frame* 19	7	8	87.50%
Video 7	*Frame* 20	7	8	87.50%
Video 8	*Frame* 1	0	7	0.00%
Video 8	*Frame* 2	6	7	85.71%
Video 8	*Frame* 3	0	7	0.00%
Video 8	*Frame* 4	6	7	85.71%
Video 8	*Frame* 5	7	7	100.00%
Video 8	*Frame* 6	7	7	100.00%
Video 8	*Frame* 7	6	7	85.71%
Video 8	*Frame* 8	7	7	100.00%
Video 8	*Frame* 9	7	7	100.00%
Video 8	*Frame* 10	7	7	100.00%
Video 8	*Frame* 11	7	7	100.00%
Video 8	*Frame* 12	7	7	100.00%
Video 8	*Frame* 13	7	7	100.00%
Video 9	*Frame* 1	7	7	100.00%
Video 9	*Frame* 2	7	7	100.00%
Video 9	*Frame* 3	7	7	100.00%

Video	*Frame*	Cakupan	Total	Akurasi
Video 9	*Frame* 4	7	7	100.00%
Video 9	*Frame* 5	7	7	100.00%
Video 9	*Frame* 6	7	7	100.00%
Video 9	*Frame* 7	7	7	100.00%
Video 9	*Frame* 8	7	7	100.00%
Video 9	*Frame* 9	7	7	100.00%
Video 9	*Frame* 10	7	7	100.00%
Video 9	*Frame* 11	7	7	100.00%
Video 9	*Frame* 12	7	7	100.00%
Video 9	*Frame* 13	7	7	100.00%
Video 9	*Frame* 14	7	7	100.00%
Video 9	*Frame* 15	7	7	100.00%
Video 9	*Frame* 16	7	7	100.00%
Video 9	*Frame* 17	7	7	100.00%
Video 9	*Frame* 18	7	7	100.00%
Video 9	*Frame* 19	7	7	100.00%
Video 9	*Frame* 20	7	7	100.00%
Video 10	*Frame* 1	7	7	100.00%
Video 10	*Frame* 2	7	7	100.00%
Video 10	*Frame* 3	7	7	100.00%
Video 10	*Frame* 4	7	7	100.00%
Video 10	*Frame* 5	7	7	100.00%
Video 10	*Frame* 6	7	7	100.00%
Video 10	*Frame* 7	7	7	100.00%
Video 10	*Frame* 8	7	7	100.00%
Video 10	*Frame* 9	7	7	100.00%
Video 10	*Frame* 10	7	7	100.00%
Video 10	*Frame* 11	7	7	100.00%

Video	*Frame*	Cakupan	Total	Akurasi
Video 10	*Frame* 12	7	7	100.00%
Video 10	*Frame* 13	7	7	100.00%
Video 10	*Frame* 14	7	7	100.00%
Video 10	*Frame* 15	7	7	100.00%
Video 10	*Frame* 16	7	7	100.00%
Video 10	*Frame* 17	7	7	100.00%
Video 10	*Frame* 18	7	7	100.00%
Video 10	*Frame* 19	7	7	100.00%
Video 10	*Frame* 20	7	7	100.00%

L.30 Hasil Uji Coba Segmentasi pada Video Malam

Video	*Frame*	TP	FP	FN	TN	Akurasi	Prec.	*Recall*
V1	F1	1	1	5	0	14.29%	50.00%	16.67%
V1	F2	1	2	5	0	12.50%	33.33%	16.67%
V1	F3	3	1	3	0	42.86%	75.00%	50.00%
V1	F4	4	1	2	0	57.14%	80.00%	66.67%
V1	F5	4	2	2	0	50.00%	66.67%	66.67%
V1	F6	1	0	5	0	16.67%	100.00%	16.67%
V1	F7	1	1	5	0	14.29%	50.00%	16.67%
V1	F8	1	0	4	0	20.00%	100.00%	20.00%
V1	F9	3	0	2	0	60.00%	100.00%	60.00%
V1	F10	1	0	4	0	20.00%	100.00%	20.00%
V1	F11	1	1	4	0	16.67%	50.00%	20.00%
V1	F12	1	1	4	0	16.67%	50.00%	20.00%
V1	F13	1	1	5	0	14.29%	50.00%	16.67%
V1	F14	3	1	3	0	42.86%	75.00%	50.00%
V2	F1	8	1	0	0	88.89%	88.89%	100.00%

Video	*Frame*	TP	FP	FN	TN	Akurasi	Prec.	*Recall*
V2	F2	8	0	0	0	100.00%	100.00%	100.00%
V2	F3	8	0	0	0	100.00%	100.00%	100.00%
V2	F4	8	0	0	0	100.00%	100.00%	100.00%
V2	F5	6	0	2	0	75.00%	100.00%	75.00%
V2	F6	6	0	2	0	75.00%	100.00%	75.00%
V2	F7	8	1	0	0	88.89%	88.89%	100.00%
V2	F8	6	1	2	0	66.67%	85.71%	75.00%
V2	F9	6	1	2	0	66.67%	85.71%	75.00%
V2	F10	6	0	2	0	75.00%	100.00%	75.00%
V2	F11	6	0	2	0	75.00%	100.00%	75.00%
V2	F12	8	0	0	0	100.00%	100.00%	100.00%
V2	F13	8	0	0	0	100.00%	100.00%	100.00%
V2	F14	8	0	0	0	100.00%	100.00%	100.00%
V2	F15	8	1	0	0	88.89%	88.89%	100.00%
V2	F16	8	0	0	0	100.00%	100.00%	100.00%
V2	F17	6	0	2	0	75.00%	100.00%	75.00%
V2	F18	8	0	0	0	100.00%	100.00%	100.00%
V2	F19	8	0	0	0	100.00%	100.00%	100.00%
V2	F20	8	0	0	0	100.00%	100.00%	100.00%
V3	F1	7	0	0	0	100.00%	100.00%	100.00%
V3	F2	7	0	0	0	100.00%	100.00%	100.00%
V3	F3	7	0	0	0	100.00%	100.00%	100.00%
V3	F4	7	0	0	0	100.00%	100.00%	100.00%
V3	F5	7	0	0	0	100.00%	100.00%	100.00%
V3	F6	7	0	0	0	100.00%	100.00%	100.00%
V3	F7	7	0	0	0	100.00%	100.00%	100.00%
V3	F8	7	0	0	0	100.00%	100.00%	100.00%
V3	F9	7	0	0	0	100.00%	100.00%	100.00%

Video	*Frame*	TP	FP	FN	TN	Akurasi	Prec.	*Recall*
V3	F10	7	0	0	0	100.00%	100.00%	100.00%
V3	F11	7	0	0	0	100.00%	100.00%	100.00%
V3	F12	7	0	0	0	100.00%	100.00%	100.00%
V3	F13	7	0	0	0	100.00%	100.00%	100.00%
V3	F14	7	0	0	0	100.00%	100.00%	100.00%
V3	F15	7	0	0	0	100.00%	100.00%	100.00%
V3	F16	7	0	0	0	100.00%	100.00%	100.00%
V3	F17	7	0	0	0	100.00%	100.00%	100.00%
V3	F18	7	0	0	0	100.00%	100.00%	100.00%
V3	F19	7	0	0	0	100.00%	100.00%	100.00%
V4	F1	7	0	1	0	87.50%	100.00%	87.50%
V4	F2	8	0	0	0	100.00%	100.00%	100.00%
V4	F3	8	0	0	0	100.00%	100.00%	100.00%
V4	F4	7	0	0	0	100.00%	100.00%	100.00%
V4	F5	8	0	0	0	100.00%	100.00%	100.00%
V4	F6	8	0	0	0	100.00%	100.00%	100.00%
V4	F7	8	0	0	0	100.00%	100.00%	100.00%
V4	F8	8	0	0	0	100.00%	100.00%	100.00%
V4	F9	8	0	0	0	100.00%	100.00%	100.00%
V4	F10	8	0	0	0	100.00%	100.00%	100.00%
V4	F11	8	0	0	0	100.00%	100.00%	100.00%
V4	F12	8	0	0	0	100.00%	100.00%	100.00%
V4	F13	8	0	0	0	100.00%	100.00%	100.00%
V4	F14	8	0	0	0	100.00%	100.00%	100.00%
V4	F15	8	0	0	0	100.00%	100.00%	100.00%
V4	F16	8	0	0	0	100.00%	100.00%	100.00%
V4	F17	8	0	0	0	100.00%	100.00%	100.00%
V4	F18	8	0	0	0	100.00%	100.00%	100.00%

Video	*Frame*	TP	FP	FN	TN	Akurasi	Prec.	*Recall*
V4	F19	8	0	0	0	100.00%	100.00%	100.00%
V4	F20	8	0	0	0	100.00%	100.00%	100.00%
V5	F1	6	1	1	0	75.00%	85.71%	85.71%
V5	F2	6	0	0	0	100.00%	100.00%	100.00%
V5	F3	6	0	0	0	100.00%	100.00%	100.00%
V5	F4	6	0	0	0	100.00%	100.00%	100.00%
V5	F5	6	1	0	0	85.71%	85.71%	100.00%
V5	F6	6	1	0	0	85.71%	85.71%	100.00%
V5	F7	6	1	0	0	85.71%	85.71%	100.00%
V5	F8	6	0	0	0	100.00%	100.00%	100.00%
V5	F9	6	0	0	0	100.00%	100.00%	100.00%
V5	F10	6	0	0	0	100.00%	100.00%	100.00%
V5	F11	6	0	0	0	100.00%	100.00%	100.00%
V5	F12	6	1	0	0	85.71%	85.71%	100.00%
V5	F13	6	0	0	0	100.00%	100.00%	100.00%
V5	F14	6	0	0	0	100.00%	100.00%	100.00%
V5	F15	6	0	0	0	100.00%	100.00%	100.00%
V5	F16	6	0	0	0	100.00%	100.00%	100.00%
V5	F17	6	0	0	0	100.00%	100.00%	100.00%
V5	F18	6	0	0	0	100.00%	100.00%	100.00%
V5	F19	6	0	0	0	100.00%	100.00%	100.00%
V5	F20	6	0	0	0	100.00%	100.00%	100.00%
V6	F1	8	4	1	0	61.54%	66.67%	88.89%
V6	F2	8	3	1	0	66.67%	72.73%	88.89%
V6	F3	9	3	0	0	75.00%	75.00%	100.00%
V6	F4	9	2	0	0	81.82%	81.82%	100.00%
V6	F5	7	3	2	0	58.33%	70.00%	77.78%
V6	F6	8	3	1	0	66.67%	72.73%	88.89%

Video	*Frame*	TP	FP	FN	TN	Akurasi	Prec.	*Recall*
V6	F7	9	2	0	0	81.82%	81.82%	100.00%
V6	F8	9	3	0	0	75.00%	75.00%	100.00%
V6	F9	9	1	0	0	90.00%	90.00%	100.00%
V6	F10	9	2	0	0	81.82%	81.82%	100.00%
V6	F11	9	3	0	0	75.00%	75.00%	100.00%
V6	F12	9	4	0	0	69.23%	69.23%	100.00%
V6	F13	7	3	2	0	58.33%	70.00%	77.78%
V6	F14	9	3	0	0	75.00%	75.00%	100.00%
V7	F1	8	0	0	0	100.00%	100.00%	100.00%
V7	F2	7	0	1	0	87.50%	100.00%	87.50%
V7	F3	7	1	1	0	77.78%	87.50%	87.50%
V7	F4	6	0	1	0	85.71%	100.00%	85.71%
V7	F5	6	0	1	0	85.71%	100.00%	85.71%
V7	F6	7	0	0	0	100.00%	100.00%	100.00%
V7	F7	7	0	1	0	87.50%	100.00%	87.50%
V7	F8	7	0	1	0	87.50%	100.00%	87.50%
V7	F9	6	0	2	0	75.00%	100.00%	75.00%
V7	F10	6	0	2	0	75.00%	100.00%	75.00%
V7	F11	7	1	1	0	77.78%	87.50%	87.50%
V7	F12	7	1	1	0	77.78%	87.50%	87.50%
V7	F13	7	1	1	0	77.78%	87.50%	87.50%
V7	F14	5	1	3	0	55.56%	83.33%	62.50%
V7	F15	7	0	1	0	87.50%	100.00%	87.50%
V7	F16	6	1	1	0	75.00%	85.71%	85.71%
V7	F17	6	0	1	0	85.71%	100.00%	85.71%
V7	F18	6	1	1	0	75.00%	85.71%	85.71%
V7	F19	6	0	1	0	85.71%	100.00%	85.71%
V7	F20	6	1	1	0	75.00%	85.71%	85.71%

Video	*Frame*	TP	FP	FN	TN	Akurasi	Prec.	*Recall*
V8	F1	4	0	2	0	66.67%	100.00%	66.67%
V8	F2	0	0	7	0	0.00%	100.00%	0.00%
V8	F3	1	0	6	0	14.29%	100.00%	14.29%
V8	F4	3	0	3	0	50.00%	100.00%	50.00%
V8	F5	1	0	6	0	14.29%	100.00%	14.29%
V8	F6	1	0	6	0	14.29%	100.00%	14.29%
V8	F7	0	0	7	0	0.00%	100.00%	0.00%
V8	F8	5	0	1	0	83.33%	100.00%	83.33%
V8	F9	1	0	6	0	14.29%	100.00%	14.29%
V8	F10	1	0	6	0	14.29%	100.00%	14.29%
V8	F11	3	0	4	0	42.86%	100.00%	42.86%
V9	F1	6	0	1	0	85.71%	100.00%	85.71%
V9	F2	7	0	0	0	100.00%	100.00%	100.00%
V9	F3	7	0	0	0	100.00%	100.00%	100.00%
V9	F4	7	0	0	0	100.00%	100.00%	100.00%
V9	F5	7	0	0	0	100.00%	100.00%	100.00%
V9	F6	7	0	0	0	100.00%	100.00%	100.00%
V9	F7	7	1	0	0	87.50%	87.50%	100.00%
V9	F8	7	0	0	0	100.00%	100.00%	100.00%
V9	F9	7	0	0	0	100.00%	100.00%	100.00%
V9	F10	7	0	0	0	100.00%	100.00%	100.00%
V9	F11	7	0	0	0	100.00%	100.00%	100.00%
V9	F12	7	1	0	0	87.50%	87.50%	100.00%
V9	F13	7	1	0	0	87.50%	87.50%	100.00%
V9	F14	7	1	0	0	87.50%	87.50%	100.00%
V9	F15	7	0	0	0	100.00%	100.00%	100.00%
V9	F16	7	1	0	0	87.50%	87.50%	100.00%
V9	F17	7	0	0	0	100.00%	100.00%	100.00%

Video	*Frame*	TP	FP	FN	TN	Akurasi	Prec.	*Recall*
V9	F18	7	1	0	0	87.50%	87.50%	100.00%
V9	F19	7	1	0	0	87.50%	87.50%	100.00%
V9	F20	7	0	0	0	100.00%	100.00%	100.00%
V10	F1	7	1	0	0	87.50%	87.50%	100.00%
V10	F2	7	1	0	0	87.50%	87.50%	100.00%
V10	F3	7	0	0	0	100.00%	100.00%	100.00%
V10	F4	5	0	2	0	71.43%	100.00%	71.43%
V10	F5	5	0	2	0	71.43%	100.00%	71.43%
V10	F6	5	0	2	0	71.43%	100.00%	71.43%
V10	F7	7	1	0	0	87.50%	87.50%	100.00%
V10	F8	7	1	0	0	87.50%	87.50%	100.00%
V10	F9	7	0	0	0	100.00%	100.00%	100.00%
V10	F10	7	1	0	0	87.50%	87.50%	100.00%
V10	F11	7	2	0	0	77.78%	77.78%	100.00%
V10	F12	7	1	0	0	87.50%	87.50%	100.00%
V10	F13	7	1	0	0	87.50%	87.50%	100.00%
V10	F14	7	0	0	0	100.00%	100.00%	100.00%
V10	F15	7	2	0	0	77.78%	77.78%	100.00%
V10	F16	7	3	0	0	70.00%	70.00%	100.00%
V10	F17	7	0	0	0	100.00%	100.00%	100.00%
V10	F18	7	0	0	0	100.00%	100.00%	100.00%
V10	F19	7	1	0	0	87.50%	87.50%	100.00%
V10	F20	7	2	0	0	77.78%	77.78%	100.00%

L.31 Hasil Uji Coba Majority Vote pada Video Malam

Vid.	*Majority Vote*	TP	FP	FN	TN	Akurasi	Prec.	*Recall*
V1	6	1	0	6	0	14.29%	100.00%	14.29%

Vid.	*Majority Vote*	TP	FP	FN	TN	Akurasi	Prec.	*Recall*
V2	AG 2351 FL	8	0	0	0	100.00%	100.00%	100.00%
V3	L 3673 DX	7	0	0	0	100.00%	100.00%	100.00%
V4	N 3526 NBB	8	0	0	0	100.00%	100.00%	100.00%
V5	0 6042 S	5	1	1	0	71.43%	83.33%	83.33%
V6	WAG 3192 WKABL	8	4	0	0	66.67%	66.67%	100.00%
V7	A 4427 M	6	0	2	0	75.00%	100.00%	75.00%
V8	W	1	0	6	0	14.29%	100.00%	14.29%
V9	S 3271 DU	7	0	0	0	100.00%	100.00%	100.00%
V10	L 3669 TT	7	0	0	0	100.00%	100.00%	100.00%

L.32 Hasil Uji Coba Pengenalan Karakter pada Video Malam

Video	*Frame*	Benar	Total	Akurasi
V1	F1	0	1	0.00%
V1	F2	1	1	100.00%
V1	F3	2	3	66.67%
V1	F4	2	4	50.00%
V1	F5	4	4	100.00%
V1	F6	1	1	100.00%
V1	F7	1	1	100.00%
V1	F8	1	1	100.00%
V1	F9	3	3	100.00%
V1	F10	1	1	100.00%
V1	F11	1	1	100.00%
V1	F12	1	1	100.00%
V1	F13	0	1	0.00%
V1	F14	2	3	66.67%

Video	*Frame*	Benar	Total	Akurasi
V2	F1	6	8	75.00%
V2	F2	8	8	100.00%
V2	F3	8	8	100.00%
V2	F4	8	8	100.00%
V2	F5	6	6	100.00%
V2	F6	6	6	100.00%
V2	F7	7	8	87.50%
V2	F8	5	6	83.33%
V2	F9	6	6	100.00%
V2	F10	6	6	100.00%
V2	F11	5	6	83.33%
V2	F12	8	8	100.00%
V2	F13	8	8	100.00%
V2	F14	8	8	100.00%
V2	F15	7	8	87.50%
V2	F16	8	8	100.00%
V2	F17	6	6	100.00%
V2	F18	8	8	100.00%
V2	F19	8	8	100.00%
V2	F20	8	8	100.00%
V3	F1	6	7	85.71%
V3	F2	6	7	85.71%
V3	F3	6	7	85.71%
V3	F4	6	7	85.71%
V3	F5	6	7	85.71%
V3	F6	7	7	100.00%
V3	F7	7	7	100.00%
V3	F8	7	7	100.00%

Video	*Frame*	Benar	Total	Akurasi
V3	F9	6	7	85.71%
V3	F10	7	7	100.00%
V3	F11	7	7	100.00%
V3	F12	7	7	100.00%
V3	F13	6	7	85.71%
V3	F14	7	7	100.00%
V3	F15	6	7	85.71%
V3	F16	7	7	100.00%
V3	F17	6	7	85.71%
V3	F18	6	7	85.71%
V3	F19	7	7	100.00%
V4	F1	7	7	100.00%
V4	F2	8	8	100.00%
V4	F3	7	8	87.50%
V4	F4	7	7	100.00%
V4	F5	8	8	100.00%
V4	F6	7	8	87.50%
V4	F7	8	8	100.00%
V4	F8	8	8	100.00%
V4	F9	8	8	100.00%
V4	F10	8	8	100.00%
V4	F11	8	8	100.00%
V4	F12	8	8	100.00%
V4	F13	8	8	100.00%
V4	F14	8	8	100.00%
V4	F15	7	8	87.50%
V4	F16	8	8	100.00%
V4	F17	8	8	100.00%

Video	*Frame*	Benar	Total	Akurasi
V4	F18	8	8	100.00%
V4	F19	8	8	100.00%
V4	F20	8	8	100.00%
V5	F1	5	6	83.33%
V5	F2	5	6	83.33%
V5	F3	5	6	83.33%
V5	F4	5	6	83.33%
V5	F5	5	6	83.33%
V5	F6	5	6	83.33%
V5	F7	5	6	83.33%
V5	F8	5	6	83.33%
V5	F9	5	6	83.33%
V5	F10	5	6	83.33%
V5	F11	5	6	83.33%
V5	F12	5	6	83.33%
V5	F13	5	6	83.33%
V5	F14	5	6	83.33%
V5	F15	5	6	83.33%
V5	F16	5	6	83.33%
V5	F17	5	6	83.33%
V5	F18	5	6	83.33%
V5	F19	5	6	83.33%
V5	F20	5	6	83.33%
V6	F1	8	8	100.00%
V6	F2	8	8	100.00%
V6	F3	7	9	77.78%
V6	F4	9	9	100.00%
V6	F5	6	7	85.71%

Video	*Frame*	Benar	Total	Akurasi
V6	F6	8	8	100.00%
V6	F7	9	9	100.00%
V6	F8	9	9	100.00%
V6	F9	9	9	100.00%
V6	F10	8	9	88.89%
V6	F11	8	9	88.89%
V6	F12	8	9	88.89%
V6	F13	7	7	100.00%
V6	F14	8	9	88.89%
V7	F1	7	8	87.50%
V7	F2	6	7	85.71%
V7	F3	6	7	85.71%
V7	F4	6	6	100.00%
V7	F5	6	6	100.00%
V7	F6	7	7	100.00%
V7	F7	6	7	85.71%
V7	F8	5	7	71.43%
V7	F9	5	6	83.33%
V7	F10	6	6	100.00%
V7	F11	7	7	100.00%
V7	F12	7	7	100.00%
V7	F13	6	7	85.71%
V7	F14	4	5	80.00%
V7	F15	6	7	85.71%
V7	F16	6	6	100.00%
V7	F17	6	6	100.00%
V7	F18	6	6	100.00%
V7	F19	6	6	100.00%

Video	*Frame*	Benar	Total	Akurasi
V7	F20	6	6	100.00%
V8	F1	3	4	75.00%
V8	F2	0	0	100.00%
V8	F3	1	1	100.00%
V8	F4	3	3	100.00%
V8	F5	1	1	100.00%
V8	F6	0	1	0.00%
V8	F7	0	0	100.00%
V8	F8	4	5	80.00%
V8	F9	1	1	100.00%
V8	F10	1	1	100.00%
V8	F11	3	3	100.00%
V9	F1	6	6	100.00%
V9	F2	7	7	100.00%
V9	F3	7	7	100.00%
V9	F4	6	7	85.71%
V9	F5	6	7	85.71%
V9	F6	6	7	85.71%
V9	F7	7	7	100.00%
V9	F8	7	7	100.00%
V9	F9	7	7	100.00%
V9	F10	7	7	100.00%
V9	F11	7	7	100.00%
V9	F12	7	7	100.00%
V9	F13	7	7	100.00%
V9	F14	7	7	100.00%
V9	F15	7	7	100.00%
V9	F16	7	7	100.00%

Video	*Frame*	Benar	Total	Akurasi
V9	F17	7	7	100.00%
V9	F18	7	7	100.00%
V9	F19	6	7	85.71%
V9	F20	7	7	100.00%
V10	F1	7	7	100.00%
V10	F2	7	7	100.00%
V10	F3	6	7	85.71%
V10	F4	4	5	80.00%
V10	F5	5	5	100.00%
V10	F6	5	5	100.00%
V10	F7	7	7	100.00%
V10	F8	7	7	100.00%
V10	F9	7	7	100.00%
V10	F10	7	7	100.00%
V10	F11	7	7	100.00%
V10	F12	7	7	100.00%
V10	F13	7	7	100.00%
V10	F14	7	7	100.00%
V10	F15	7	7	100.00%
V10	F16	7	7	100.00%
V10	F17	7	7	100.00%
V10	F18	7	7	100.00%
V10	F19	7	7	100.00%
V10	F20	7	7	100.00%

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Nuzha Musyafira, lahir di Bangil pada tanggal 15 Maret 2000. Penulis menempuh pendidikan mulai dari TK ABA IV (2004 – 2006), SDN Jember Lor 1 (2006 – 2012), SMP Negeri 2 Jember (2012 – 2014), SMA Negeri 1 Jember (2014 – 2016), dan sekarang sedang menjalani pendidikan S1 Teknik Informatika di ITS. Penulis aktif dalam organisasi dan kepanitiaan UKM *Maritime Challenge* (MC) ITS, *Indonesia Maritime Challenge* (IMC), dan Schematics. Di antaranya adalah menjadi *staff* Divisi Produksi, Perbaikan, dan Perawatan (P3) Kapal UKM MC ITS 2017-2018, Sekretaris 1 UKM MC ITS 2018-2019, *staff* Departemen Acara IMC 2017, *staff* ahli Departemen Acara IMC 2018, *staff* Departemen *National Logic Competition* Schematics ITS 2017, dan *staff* ahli Departemen *National Logic Competition* Schematics ITS 2018. Komunikasi dengan penulis dapat melalui telepon: +6289608555029 dan *email*: **nuzhamusyafira@gmail.com**.