Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - IF184802

Pengenalan Bahasa Isyarat pada Data Video Menggunakan Metode CNN dengan Arsitektur YOLO

STEVE DANIELS
NRP 05111640000084

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

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

Pengenalan Bahasa Isyarat pada Data Video Menggunakan Metode CNN dengan Arsitektur YOLO

STEVE DANIELS
NRP 05111640000084

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

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

SIGN LANGUAGE RECOGNITION ON VIDEO USING CNN BASED METHOD WITH YOLO ARCHITECTURE

STEVE DANIELS
NRP 05111640000084

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

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

PENGENALAN BAHASA ISYARAT PADA DATA VIDEO MENGGUNAKAN METODE CNN DENGAN ARSITEKTUR YOLO

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:
STEVE DANIELS
NRP: 05111640000084

Disetujui oleh Pembimbing Tugas Akhir:

1. Dr. Eng. Nanik Suciati, S.Kom., M.Kom. .....
   (NIP. 19710428 199412 2 001) (Pembimbing 1)

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

SURABAYA
JUNI, 2020

(Halaman ini sengaja dikosongkan)

PENGENALAN BAHASA ISYARAT PADA DATA VIDEO MENGGUNAKAN METODE CNN DENGAN ARSITEKTUR YOLO

Nama : Steve Daniels
NRP : 05111640000084
Departemen : Teknik Informatika, FTEIC-ITS
Pembimbing I : Dr.Eng. Nanik Suciati, S.Kom, M.Kom
Pembimbing II : Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

ABSTRAK

Bahasa isyarat adalah salah satu sarana penting dalam berkomunikasi. Isyarat merupakan bentuk komunikasi yang mudah dipakai oleh penyandang tunarungu, individu yang memiliki hambatan dalam berbicara, atau orang yang memiliki anggota keluarga penyandang tunarungu. Akan tetapi, kebanyakan orang normal tidak mempelajari bahasa isyarat. Karena itu, translasi dari bahasa isyarat menjadi alfabet/teks secara otomatis akan memudahkan komunikasi dengan pengguna bahasa isyarat.

Tujuan tugas akhir ini adalah mengembangkan sistem pengenalan bahasa isyarat yang dapat membaca masukan dari data video secara real-time menggunakan metode You Only Look Once (YOLO). YOLO merupakan salah satu metode deteksi objek berbasis Convolutional Neural Network (CNN) yang terkenal dengan kecepatannya. Pelatihan dilakukan dengan penyesuaian berdasarkan jumlah channel dan jumlah kelas dari dataset.

Dataset yang dipakai untuk pelatihan dan pengujian model adalah data yang diambil secara mandiri berdasarkan Bahasa Isyarat Indonesia (Bisindo). Dataset melalui tahapan praproses yaitu resizing, konversi ke grayscale, dan penambahan data/augmentasi. Sebagian data hasil praproses dipakai untuk pelatihan model, yang dilakukan dengan berbagai learning rate. Model terbaik didapatkan dari pelatihan dengan learning rate sebesar 0.001. Dari uji coba pengenalan Bahasa isyarat

menggunakan YOLO, didapatkan nilai tertinggi dari presisi, recall, akurasi, dan F1 score sebesar 100% pada uji coba data gambar. Untuk uji coba data video, didapatkan presisi sebesar 77.14%, recall sebesar 93.1%, akurasi sebesar 72.97%, F1 score sebesar 84.38%, serta tingkat Frame Per Seconds (FPS) sebesar 8 frame setiap detik.

Kata kunci: ***Pengenalan Bahasa Isyarat, Convolutional Neural Network, YOLO***.

SIGN LANGUAGE RECOGNITION ON VIDEO USING CNN BASED METHOD WITH YOLO ARCHITECTURE

Student's Name : Steve Daniels
Student's ID : 05111640000084
Department : Informatics, Faculty of ELECTICS-ITS
First Advisor : Dr.Eng. Nanik Suciati, S.Kom, M.Kom
Second Advisor : Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

ABSTRACT

Sign language is an important means of communication. Cues are used as a form of communication that is easily used by people with hearing impairment, individuals who have speech impediments, or people who have family members who are deaf. However, sign language is not learned by most normal people. Therefore, the translation of sign language into the alphabet/text automatically will facilitate the communication of the deaf with normal people.

The goal of this final project is to develop a sign language recognition system that can read input from video data in real-time using You Only Look Once (YOLO). YOLO is an object detection method based on Convolutional Neural Network (CNN) which is famous for its speed. The training is done with adjustments based on the number of channels and classes from the dataset.

The dataset used for model training and testing is data taken independently based on the Indonesian Sign Language (Bisindo). The dataset goes through preprocessing stages, namely resizing, converting to grayscale, and augmentation. Part of the preprocessed data is used for model training, which is carried out at various learning rates. The best model is obtained from training with learning rate of 0.001. From the sign language recognition test using YOLO, the results show highest score in precision, recall, accuracy, and F1 score of 100% for image testing. Video testing shows precision of 77.14%, recall of 93.1%,

accuracy of 72.97%, F1 score of 84.38%, and FPS rate of 8 frames every second.

Keywords: ***Sign Language Recognition, Convolutional Neural Network, YOLO***.

KATA PENGANTAR

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

"**PENGENALAN BAHASA ISYARAT PADA DATA VIDEO MENGGUNAKAN METODE CNN DENGAN ARSITEKTUR YOLO**"

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. Dr.Eng. Nanik Suciati, S.Kom, M.Kom dan Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. selaku pembimbing I dan II yang telah membimbing dan memberikan motivasi, nasihat dan arahan 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. Rifqi Mukti, Ariiq Firanda, Ayas Faikar, Puguh Santosa, dan Vinsensius Indra sebagai teman-teman yang hampir selalu memberikan dukungan moral, motivasi, dan hiburan kepada penulis.
5. Admin-admin Laboratorium Komputasi Cerdas & Visi (KCV) yang memberikan kesempatan penulis untuk

fokus mengerjakan Tugas Akhir ini dan menyediakan tempat di laboratorium tersebut.

6. Seluruh mahasiswa *user* TA Teknik Informatika ITS angkatan 2016 yang telah menjadi teman penulis selama pengerjaan Tugas Akhir ini.
7. Seluruh mahasiswa Teknik Informatika ITS angkatan 2016 yang telah menjadi teman penulis selama menjalani masa perkuliahan.
8. Serta seluruh pihak yang telah 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, Juni 2020

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Bahasa isyarat merupakan salah satu bentuk komunikasi manusia berupa non-verbal yang diekspresikan melalui gerakan atau bentuk tangan untuk menyampaikan suatu informasi kepada penerima secara visual. Bahasa isyarat biasa digunakan oleh penyandang tunarungu, individu yang dapat mendengar tetapi memiliki hambatan dalam berbicara, atau orang normal yang memiliki anggota keluarga penyandang tunarungu. Bahasa isyarat tidak dipelajari oleh kebanyakan orang normal, sehingga tidak semua orang memahami bahasa tersebut. Karena itu, translasi dari bahasa isyarat menjadi alfabet/teks secara otomatis akan memudahkan komunikasi penyandang tunarungu dengan orang normal.

Salah satu metode deep learning yang akhir-akhir ini banyak digunakan untuk deteksi objek dan berbasis *Convolutional Neural Network* (CNN) adalah *You Only Look Once* (YOLO). YOLO dapat menghasilkan deteksi objek yang cepat dan efektif [1] dan menjadi salah satu keunggulan YOLO dibanding metode deteksi objek lainnya. YOLO digunakan dalam berbagai penelitian yang memakai pendeteksian objek, seperti lokalisasi pelat nomor Bhutan secara real-time [2], pendeteksian pejalan kaki [3], dan pengenalan rambu lalu lintas [4].

Metode CNN pernah dipakai untuk mendeteksi *American Sign Language* (ASL) untuk ditranslasikan menjadi bentuk huruf [5]. Arsitektur CNN yang digunakan adalah GoogLeNet, yang sudah dilatih pada dataset 2012 ILSVRC. Hasil evaluasi yang diberikan yaitu akurasi sebesar 72%. Kurangnya akurasi tersebut disebabkan oleh kesalahan klasifikasi pada huruf tertentu yang memiliki bentuk tangan yang mirip (sebagai contoh: g/h dan m/n/s/t).

Di dalam tugas akhir ini, penulis melakukan implementasi CNN dengan arsitektur YOLO untuk mengenali bahasa isyarat

dari data video. Data pelatihan dan uji coba diambil dari dataset yang diambil oleh penulis sesuai dengan kode tangan abjad jari dari Bahasa Isyarat Indonesia (BISINDO) untuk 24 huruf.

1.2 Rumusan Masalah

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

1. Bagaimana cara mengimplementasikan CNN dengan arsitektur YOLO untuk pengenalan bahasa isyarat?
2. Bagaimana mengukur kinerja pengenalan bahasa isyarat menggunakan metode CNN dengan arsitektur YOLO?

1.3 Batasan Permasalahan

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

1. Implementasi tugas akhir ini menggunakan bahasa pemrograman *Python*.
2. Dataset yang digunakan adalah data yang diambil oleh penulis berdasarkan Bisindo.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah mengaplikasikan metode CNN dengan arsitektur YOLO dalam pengenalan bahasa isyarat pada video.

1.5 Manfaat

Tugas akhir ini diharapkan dapat meningkatkan kualitas dan kemudahan komunikasi manusia yang memakai bahasa isyarat sebagai sarana komunikasi.

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 *Convolutional Neural Network* dan YOLO.

1.6.3 Implementasi Perangkat Lunak

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

1.6.4 Pengujian dan Evaluasi

Tahap pengujian dan evaluasi dilakukan menggunakan data yang diambil penulis berdasarkan BISINDO untuk mengetahui hasil dan performa metode yang telah dibangun. Evaluasi dan perbaikan dilakukan untuk mendapatkan kinerja yang baik. Beberapa pengujian yang dilakukan antara lain perubahan-perubahan pada parameter pada arsitektur YOLO yang dibuat. Evaluasi akurasi akan dilakukan dengan menggunakan *Confusion*

Matrix yang memiliki empat hasil berupa: *recall*, *precision*, *accuracy*, dan *F1 Score*.

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 CNN, YOLO, Python, serta *framework* dan *library* yang digunakan.

Bab III Perancangan Sistem

Bab ini berisi pembahasan mengenai perancangan dari metode *You Only Look Once* (YOLO) yang digunakan untuk pengenalan bahasa isyarat 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.

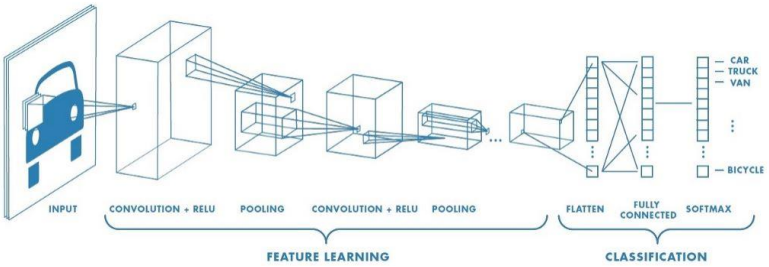
(Halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

Bab ini membahas mengenai teori-teori dasar yang digunakan dalam Tugas Akhir. Teori-teori tersebut adalah *Convolutional Neural Network, YOLO,* 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 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) adalah salah satu jenis neural network yang biasa digunakan pada data gambar. CNN terdiri dari neuron yang memiliki weight, bias, dan fungsi aktivasi. Arsitektur CNN terdiri dari 2 bagian yaitu *Feature Extraction Layer* dan *Fully Connected Layer* seperti pada Gambar 2.1.



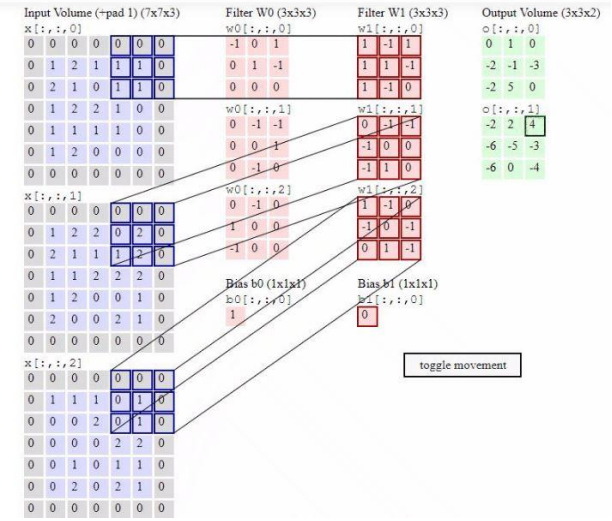
Gambar 2.1 Ilustrasi Arsitektur CNN [6]

Pada *Feature Extraction Layer*, gambar diterjemahkan menjadi fitur yang berupa angka-angka yang merepresentasikan gambar tersebut. Fitur ini kemudian dikumpulkan menjadi *feature map*. *Extraction Layer* umumnya terdiri dari 2 lapisan yaitu *Convolutional Layer* dan *Pooling Layer*.

1. *Convolutional Layer*

   C*onvolutional Layer* terdiri dari layer terdiri dari neuron yang tersusun sedemikian rupa sehingga membentuk sebuah

filter dengan piksel dengan *pengenalan* otomatis. Filter ini digeser ke seluruh bagian gambar, dilakukan operasi dot antara input dan nilai dari filter tersebut. Lalu didapatkan keluaran atau biasa disebut dengan *activation map* atau *feature map*. Penggambaran cara kerja *Convolutional Layer* dapat dilihat pada Gambar 2.2.



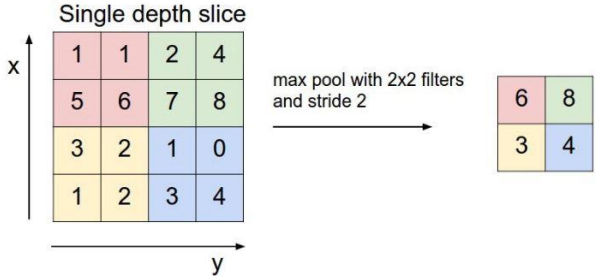
Gambar 2.2 Simulasi Convolutional Layer [6]

Sebagai contoh, gambar di atas merupakan simulasi dari *convolutional layer*. Terdapat 2 filter dengan ukuran 3×3, yang digeser ke seluruh input yang memiliki *padding* (penambahan bagian luar) = 1, dilakukan operasi dot dan menghasilkan *feature map*. Ukuran dari output dipengaruhi oleh ukuran input, ukuran filter, *padding*, dan *stride* (jarak langkah pergeseran). *Depth*/kedalaman dari output ditentukan dari jumlah filter.

2. *Pooling Layer*

Pooling Layer berada setelah *Convolutional Layer*. Lapisan ini terdiri dari sebuah filter dengan ukuran tertentu yang akan bergeser pada seluruh area feature map [6].

Pooling yang digunakan biasanya adalah *Max Pooling* dan *Average Pooling*. Sebagai contoh, jika digunakan *Max Pooling* 2×2, maka pada setiap pergeseran filter, nilai maksimum pada area 2×2 piksel tersebut akan dipilih. Sedangkan pada *Average Pooling* akan memilih nilai rata-rata. Contoh dapat dilihat pada Gambar 2.3.

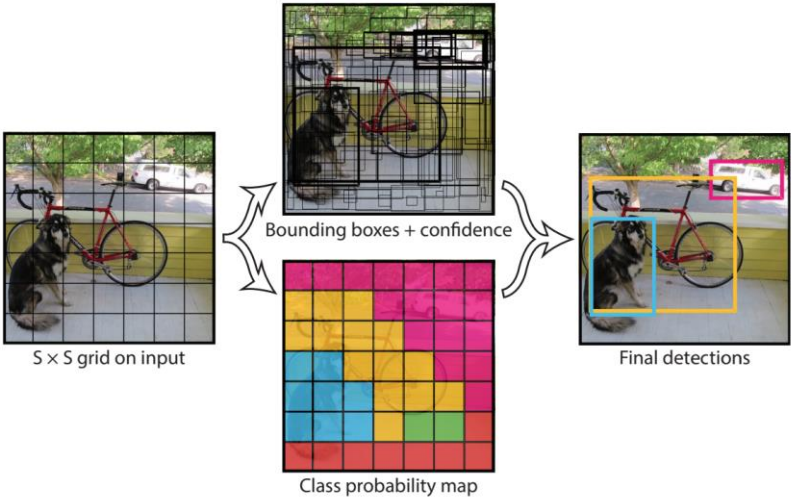


Gambar 2.3 Ilustrasi konsep *Max-Pooling* [6]

Feature map yang dihasilkan dari *feature extraction layer* masih berbentuk multidimensional array. Karena itu, dilakukan *flattening* untuk mengubah bentuk dari *feature map* menjadi vektor untuk digunakan sebagai input dari *fully connected layer*. Hasil dari layer ini adalah output dari network.

2.2 You Only Look Once (YOLO)

YOLO merupakan metode deteksi objek yang terkenal dengan kecepatannya. YOLO hanya melakukan CNN sekali saja pada seluruh gambar untuk memprediksi objek apa saja yang ada pada gambar. Hal ini bisa terjadi karena YOLO membagi gambar menjadi sel/*grid*, setiap sel bertanggung jawab untuk memprediksi sejumlah *bounding box* dan tingkat *confidence* dari setiap sel serta probabilitas kelas, diilustrasikan pada Gambar 2.4.



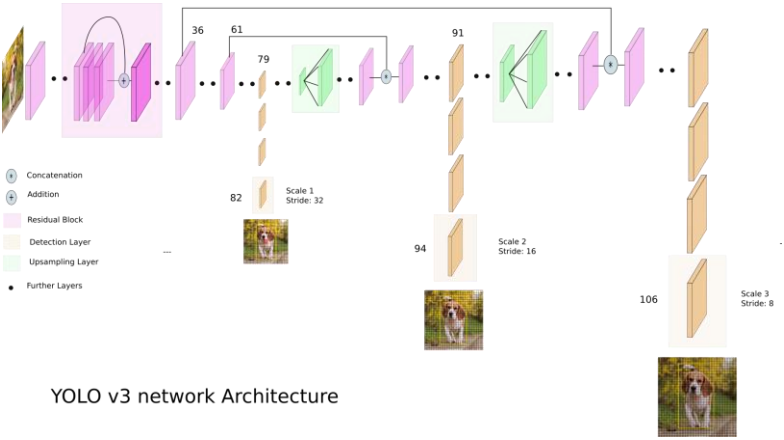
Gambar 2.4 Ilustrasi cara kerja YOLO [1]

Sebagai contoh, untuk mengevaluasi YOLO di PASCAL VOC, YOLO menggunakan 7 × 7 *grid* (S × S), 2 *bounding boxes* (B) dan 20 kelas (C). Setiap *bounding box* mempunyai 5 elemen yang diprediksi antara lain *x, y, w, h* dan *box confidence score*, dimana *x* dan *y* adalah koordinat yang merepresentasikan titik tengah atau pusat dari *bounding box*, *w* dan *h* merupakan lebar dan tinggi dari *bounding box*, dan *box confidence score* menunjukkan besar kemungkinan *bounding box* tersebut berisi objek. Setiap sel mempunyai 20 probabilitas kelas kondisional yang merupakan probabilitas kelas dari objek yang terdeteksi.

Prediksi YOLO memiliki bentuk tensor S × S × (B×5 + C). Dengan $S = 7$, $B = 2$, dan $C = 20$, prediksi final memiliki 7 × 7 × (2×5 + 20) atau 7 × 7 × 30 tensor. YOLO mempunyai 24 layer CNN untuk mengekstraksi fitur gambar dan 2 *connected layer* untuk melakukan regresi linier untuk membuat 7 × 7 × 2 prediksi *bounding box*.

2.3 YOLOv3

YOLOv3 merupakan perkembangan dari YOLO, dimana terdapat perubahan jaringan ekstraksi fitur, penggunaan *anchor box*, deteksi berbagai skala [7].

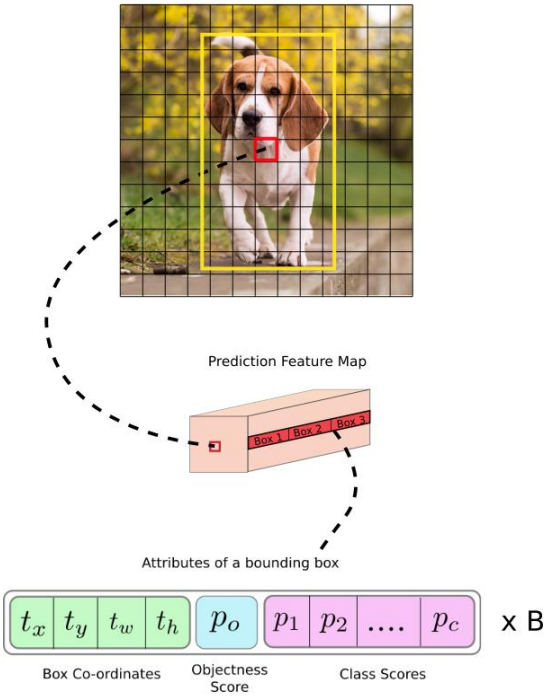


Gambar 2.5 Ilustrasi Arsitektur YOLOv3 [8]

Arsitektur YOLOv3 diilustrasikan pada Gambar 2.5. Secara garis besar, YOLO terdiri dari jaringan ekstraksi fitur dan *detection layer* dengan 3 skala yang berbeda. Arsitektur ini terdiri dari 75 *convolutional layer*, 23 *shortcut layer*, 4 *route layer*, 3 *yolo* layer, dan 2 *upsample layer*. *Shorcut layer* adalah merupakan penambahan *feature map* dari *layer* sebelumnya dengan *layer* dari beberapa lompatan sebelumnya. *Route layer* mengambil *feature map* dari *layer* beberapa lompatan sebelumnya dan menggabungkannya dengan *feature map* dari *layer* lain. *Yolo layer* berfungsi untuk mengeluarkan *output* prediksi dengan *bounding box*. *Upsample layer* berfungsi untuk memperbesar *feature map*, menggunakan *nearest neighbor interpolation*. *Layer* untuk mendeteksi terdiri dari beberapa *convolutional layer* dengan ukuran sesuai skala pendeteksian dan diakhiri dengan

layer yang memiliki jumlah filter yang disesuaikan dengan jumlah kelas.

Jaringan ekstraksi fitur yang dipakai hanya terdiri dari *Convolutional Layer*, diikuti oleh *batch normalization* dan fungsi Aktivasi *Leaky ReLU*. *Batch normalization* berguna untuk membantu proses *convergence* model saat training. Jaringan ini juga memiliki *residual block* yang memasukkan hasil dari suatu layer ke layer setelahnya dan ke suatu layer lebih dalam lagi secara langsung, sehingga tidak terjadi masalah degradasi pada jaringan yang memiliki banyak layer.



Gambar 2.6 Output YOLOv3 [9]

Masing-masing sel bertugas untuk melakukan prediksi koordinat tengah box, skor objek, dan skor kelas sesuai dengan jumlah *anchor box* seperti ilustrasi pada Gambar 2.6. Penggunaaan *anchor box* dilakukan untuk menggantikan prediksi lebar dan tinggi *bounding box* secara langsung karena melakukan prediksi lebar dan tinggi *bounding box* menyebabkan gradien yang tidak stabil saat training. Prediksi *bounding box* dilakukan dengan melakukan transformasi pada anchor box yang sudah didefinisikan dari awal. Berikut adalah rumus bagaimana output dari jaringan ditransformasikan untuk mendapat prediksi *bounding box:*

$$b_x = \sigma(t_x) + c_x \quad (2.1)$$
$$b_y = \sigma(t_y) + c_y \quad (2.2)$$
$$b_w = p_w e^{t_w} \quad (2.3)$$
$$b_h = p_h e^{t_h} \quad (2.4)$$

Persamaan (2.1) dan (2.2) digunakan untuk mendapatkan titik tengah (*x*, *y*) *bounding box* sedangkan persamaan (2.3) dan (2.4) digunakan untuk mendapatkan lebar dan tinggi *bounding box*. Dalam persamaan tersebut, b_x, b_y, b_w, b_h adalah koordinat tengah (*x*, *y*), lebar, dan tinggi dari prediksi. t_x, t_y, t_w, t_h adalah hasil output dari jaringan. Variabel c_x dan c_y adalah koordinat pojok kiri atas dari sel. Dan variabel p_w and p_h adalah dimensi dari *anchor* untuk *bounding box.*

Prediksi untuk koordinat tengah dilakukan relatif terhadap pojok kiri atas sel yang memprediksi objek. Fungsi sigmoid dipakai untuk prediksi ini agar hasil dari prediksi titik tengah tetap berada dalam sel yang memprediksi. Skor objek dan skor kelas (*confidence*) juga masuk ke dalam fungsi sigmoid sehingga hasil bernilai 0 sampai 1.

YOLOv3 melakukan prediksi dalam 3 skala yang berbeda. Layer untuk mendeteksi digunakan untuk 3 *feature map* dalam ukuran yang berbeda, masing-masing memiliki *stride* 32, 16, dan

8. Dengan input sebesar 416 × 416, deteksi dilakukan pada skala 13 × 13, 26 × 26, dan 52 × 52.

Setelah mendapatkan ouput, output tersebut lalu di filter untuk membuang prediksi *bounding box* yang mempunyai nilai deteksi dibawah *threshold* akan dibuang.

YOLOv3 sudah dipakai dalam beberapa penelitian, sehingga tersedia beberapa model yang sudah dilatih pada beberapa dataset umum, seperti ImageNet yang merupakan salah satu dataset gambar dengan jumlah gambar dan kelas terbesar, memiliki lebih dari 14 juta gambar dengan lebih dari 21 ribu kelas [10]. Contoh dataset lainnya adalah *Common Objects in Context* (COCO), salah satu dataset yang sering dipakai dalam penelitian deteksi objek, dan WIDER FACE yang digunakan untuk deteksi wajah.

2.4 Non-Maximum Suppression (NMS)

YOLO menggunakan *Non-Maximum Suppression* (NMS) untuk menghilangkan beberapa *bounding box* dengan prediksi objek yang sama, dan menyisakan satu *bounding box*. *Bounding box* dengan *confidence* tertinggi akan diambil sebagai prediksi, lalu digunakan sebagai pembanding untuk *bounding box* yang lain menggunakan *Intersection over Union* (IOU). Jika IOU melebihi *threshold* yang ditentukan, maka *bounding box* yang dibandingkan tersebut akan dihapus. Proses dilakukan secara berulang sampai tidak ada *bounding box* prediksi yang bertumpuk [11]. Ilustrasi hasil NMS dapat dilihat pada Gambar 2.7.

Gambar 2.7 NMS dengan IOU [11]

2.5 Python

Python adalah sebuah bahasa pemrograman yang multi-interpretatif dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode yang tinggi dan berorientasi objek. Python memiliki data struktur dengan level pembangunan yang tinggi. Python juga memiliki bahasa pemrograman yang simpel dan mudah dipelajari sehingga memiliki tingkat keterbacaan yang baik serta mudah untuk dilakukan perawatan [12].

2.6 Darknet

Darknet adalah *open source framework* untuk *neural network* yang ditulis dalam C dan CUDA [13]. *Framework* ini dikembangkan oleh salah satu pencipta YOLO, Joseph Redmon.

2.7 NumPy

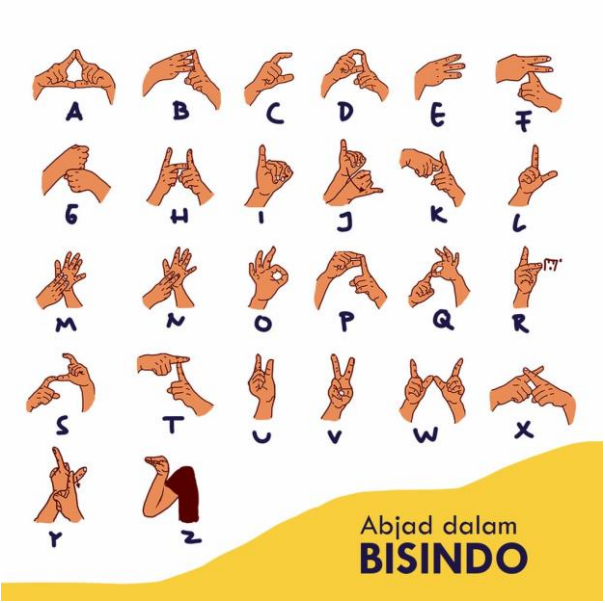
NumPy adalah *library* untuk bahasa pemrograman Python yang mendukung pengolahan data pada *array* dan matriks multidimensional yang besar. NumPy menyediakan banyak fungsi matematika yang dioperasikan pada *array* tersebut. NumPy bersifat *open source* dan memiliki banyak kontributor [14].

2.8 OpenCV

OpenCV adalah *library* yang ditujukan untuk pengolahan gambar/citra secara *real-time*. OpenCV dapat dipakai pada berbagai bahasa pemrograman seperti C++ dan python, serta dapat dipakai pada berbagai sistem operasi seperti Linux, Windows, dan MacOS [15].

2.9 Bahasa Isyarat Indonesia

Bahasa Isyarat Indonesia (BISINDO) adalah salah satu bahasa isyarat yang berlaku di Indonesia, dan merupakan bahasa yang tumbuh secara alami pada kalangan komunitas Tuli di Indonesia [16]. Isyarat huruf BISINDO berupa bentuk tangan secara statis, kecuali pada huruf J dan R yang menggunakan gerakan. Bentuk tangan isyarat huruf dasri BISINDO dapat dilihat pada Gambar 2.8.



Gambar 2.8 Isyarat huruf BISINDO [17]

2.10 Precision

Precision adalah perbandingan kebenaran prediksi suatu kelas dengan jumlah prediksi yang dilakukan pada kelas tersebut.

$$Precision_A = \frac{TP_A}{Predict_A} \qquad (2.5)$$

Cara mencari *precision* dari kelas A dapat dilihat pada persamaan (2.5), dimana TP_A adalah jumlah prediksi kelas A yang benar dan $Predict_A$ adalah jumlah prediksi kelas A. Untuk *precision* keseluruhan dapat dilakukan dengan menghitung rata-rata *precision* setiap kelas.

2.11 Recall

Recall adalah perbandingan jumlah prediksi benar yang dilakukan dengan jumlah kebenaran yang ada.

$$Recall_A = \frac{TP_A}{Truth_A} \qquad (2.6)$$

Cara untuk mencari *recall* dari kelas A dapat dilihat pada persamaan (2.6), dimana TP_A adalah jumlah prediksi kelas A yang benar dan $Truth_A$ adalah jumlah *ground truth* kelas A. . Untuk *recall* keseluruhan dapat dilakukan dengan menghitung rata-rata *recall* setiap kelas.

2.12 Accuracy

Accuracy adalah penilaian ketepatan berdasarkan seluruh kebenaran prediksi yang dilakukan dibandingkan dengan seluruh prediksi yang dilakukan.

$$Accuracy = \frac{TP}{Predict+FN} \qquad (2.7)$$

Cara untuk mencari *accuracy* dapat dilihat pada persamaan (2.7), dimana TP adalah jumlah seluruh prediksi yang benar, FN adalah jumlah kasus tidak melakukan prediksi pada data yang memiliki kelas, dan Predict adalah jumlah seluruh prediksi yang dilakukan.

2.13 F1 Score

F1 Score adalah penilaian dengan mengambil keseimbangan antara *precision* dengan *recall*.

$$F1\ score = 2 \times \frac{Prec\ X\ Rec}{Prec+Rec} \qquad (2.7)$$

Cara untuk mencari *F1 score* dapat dilihat pada persamaan (2.7), dimana Prec adalah *precision* secara keseluruhan dan Rec adalah *recall* secara keseluruhan.

BAB III
PERANCANGAN SISTEM

Bab ini menjelaskan tentang perancangan data dan sistem pengenalan bahasa isyarat menggunakan YOLO. Bab ini juga menjelaskan gambaran umum sistem dalam bentuk diagram alir.

3.1 Perancangan Data

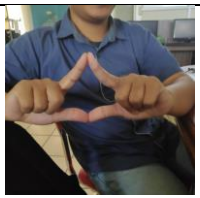
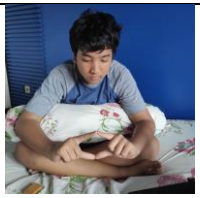
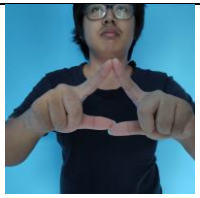
Data yang digunakan sebagai masukan dari sistem pengenalan bahasa isyarat adalah dataset yang diambil oleh penulis berupa foto dan video, terdiri dari 24 kelas di mana pada masing-masing kelas terdapat sekitar 160 sampai dengan 220 citra. Data ini merupakan sekumpulan citra karakter A sampai dengan Z kecuali J dan R.

Tabel 3.1 Spesifikasi dataset karakter

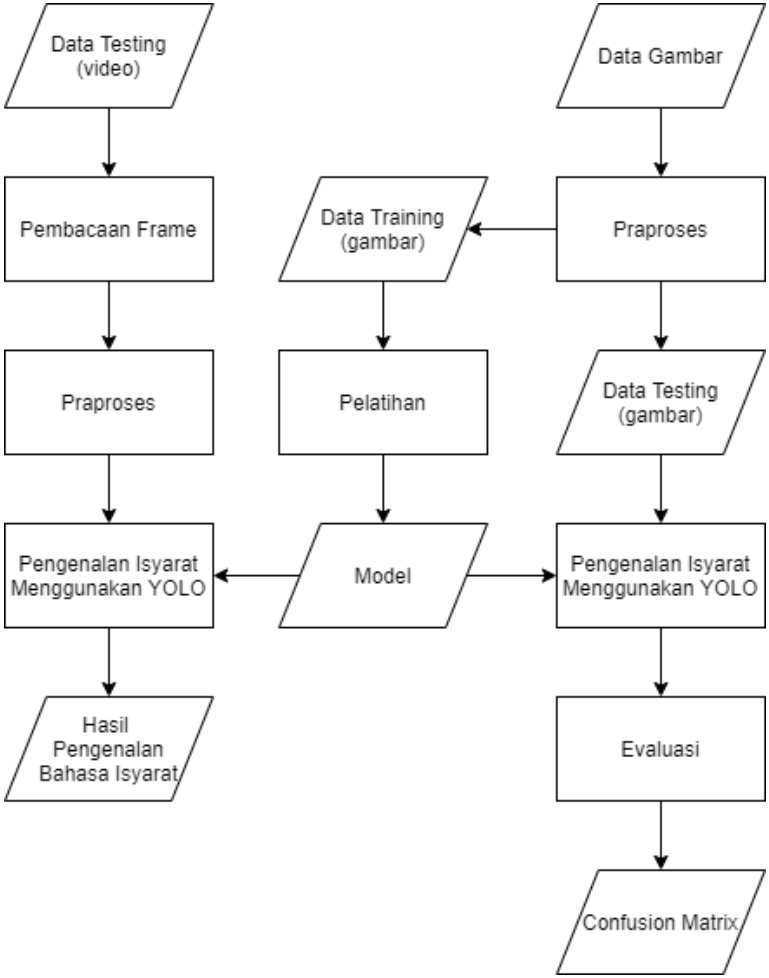
Keterangan	**Spesifikasi**
Ukuran resolusi	3024 × 3024, 640 × 480
Ekstensi	.jpg
Jumlah gambar	4,547
Jumlah kelas	24 kelas
Jumlah gambar per kelas	160-220
Ukuran file	50-500 kB
Kanal warna	3 (RGB)

Pembagian data latih dan data uji coba dengan rasio 8:2, 8 untuk data latih dan 2 untuk data uji coba. Spesifikasi dataset pengenalan karakter dapat dilihat pada Tabel 3.1. Dataset diambil dengan subjek dan lingkungan yang berbeda, sejumlah 5 kondisi yang dijelaskan pada Tabel 3.2.

Tabel 3.2 Penjelasan kondisi pada dataset

Contoh Gambar	Penjelasan Kondisi
	Kondisi 1 (K1) Jumlah gambar: 700 Kondisi lingkungan: ruangan, malam hari, pencahayaan dari lampu, agak redup. Pengambilan dilakukan dengan cara foto. Jarak pengambilan: Dekat, sekitar 20-40 cm
	Kondisi 2 (K2) Jumlah gambar: 662 Kondisi lingkungan: ruangan, siang hari, tidak memakai pencahayaan dari lampu, terang. Pengambilan dilakukan dengan cara foto. Jarak pengambilan: Dekat, sekitar 20-40 cm
	Kondisi 3 (K3) Jumlah gambar: 800 Kondisi lingkungan: ruangan, siang hari, tidak memakai pencahayaan dari lampu, terang. Pengambilan dilakukan dengan cara foto. Jarak pengambilan: Dekat-Sedang, sekitar 20-80 cm
	Kondisi 4 (K4) Jumlah gambar: 728 Kondisi lingkungan: ruangan, siang hari, tidak memakai pencahayaan dari lampu, terang. Pengambilan dilakukan dengan cara foto. Jarak pengambilan: Dekat, sekitar 20-50 cm
	Kondisi 5 (K5) Jumlah gambar: 1657 Kondisi lingkungan: ruangan, siang hari, tidak memakai pencahayaan dari lampu, terang. Pengambilan dilakukan dengan merekam video lalu menyimpan frame sebagai gambar. Jarak pengambilan: Jauh, sekitar 150cm

3.2 Desain Umum Sistem

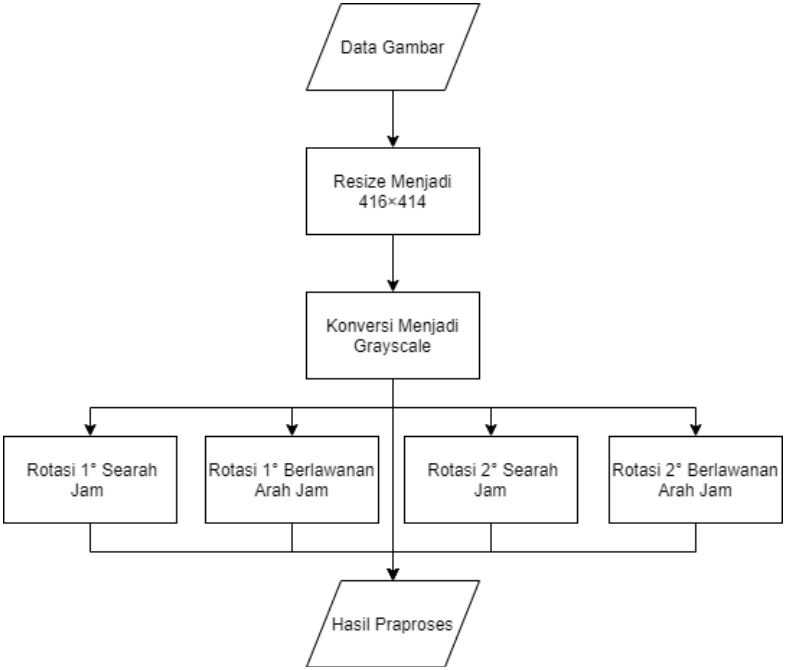


Gambar 3.1 Diagram alir sistem yang dibangun

Sistem pengenalan bahasa isyarat yang dibangun memiliki beberapa tahapan: praproses (*preprocessing*), pelatihan

model/*training*, dan Testing. Diagram alir dari sistem ditunjukkan pada Gambar 3.1.

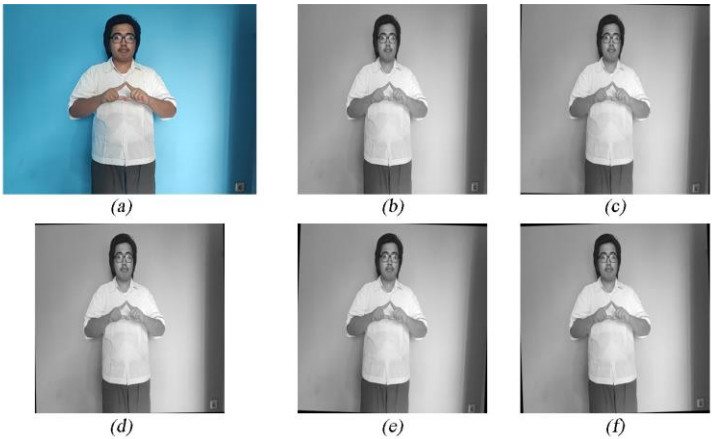
3.2.1 Tahapan Praproses



Gambar 3.2 Diagram Alir Tahapan Praproses

Pada tahap ini, akan dilakukan praproses data terlebih dahulu untuk masukan pada proses pelatihan. Praproses data yang dilakukan dapat dilihat pada Gambar 3.2, meliputi perubahan ukuran / *resize* menjadi 416 × 416, pengubahan jumlah channel dari 3 (RGB) menjadi 1 (*grayscale)*, dan penambahan data dengan rotasi dari gambar yang sudah tersedia sebesar 1 dan 2 derajat searah jam dan berlawanan arah jam. *Resizing* dilakukan untuk menyamakan ukuran data, pengubahan gambar menjadi grayscale dilakukan karena yang dicari adalah fitur bentuk, dan

penambahan data dilakukan untuk memperbanyak variasi data. Contoh hasil praproses dapat dilihat pada Gambar 3.3.



Gambar 3.3 (a) Gambar asli; (b) Hasil *resize* dan pengubahan ke *grayscale*; (c) dan (d) hasil rotasi 1 derajat berlawanan arah jam dan searah jam; (e) dan (f) hasil rotasi 2 derajat berlawanan dan searah jarum jam

3.2.2 Tahapan Pelatihan

Untuk pelatihan, data yang dipakai sebagai input untuk pelatihan adalah data yang sudah melewati tahapan praproses. 80% dari data yang sudah melewati tahapan praproses tersebut dipakai untuk pelatihan, dan siswa 20% dipakai untuk uji coba.

Penyesuaian dilakukan dari arsitektur awal YOLO, dimana perubahan terjadi pada jumlah *channel* dan jumlah *filter* dalam *convolutional layer* sebelum masuk ke *yolo layer*. Proses pelatihan memanfaatkan data latih untuk membangun model YOLO. Pelatihan menggunakan *pretrained weight darknet53* yang dilatih pada ImageNet dengan berbagai *learning rate*. Proses pelatihan akan dijalankan pada *batch size* 64. Pelatihan akan diberhentikan Ketika nilai *loss* mulai tidak mengalami penurunan.

3.2.3 Tahapan Pengujian

Tahap pengujian ini memiliki dua jenis pengujian, yaitu pengujian terhadap data gambar dan terhadap data video.

Pada pengujian gambar, dilakukan pendeteksian pada sejumlah gambar yang sudah dipersiapkan untuk testing. Pada akhir pengujian, akan dilakukan perbandingan terhadap hasil deteksi dengan *ground truth*, mendapatkan *confusion matrix* untuk mendapat nilai *akurasi*, *precision* dan *recall*.

Pada pengujian video, akan dilakukan pendeteksian pada setiap frame dari data video. Jika suatu kelas terdeteksi secara berurutan lebih dari batas yang ditentukan, kelas tersebut akan dianggap dikenal. Hasil objek yang dikenal akan dibandingkan dengan *ground truth* yang disediakan.

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 laptop ASUS X550VX. Sistem operasi yang digunakan adalah Windows 10 64-bit. PC yang digunakan memiliki spesifikasi Intel i7 6700 dengan kecepatan 2.6 GHz, *Random Access Memory* (RAM) sebesar 8 GB, dan mempunyai *Graphics Processing Unit* (GPU) yaitu NVIDIA GeForce GTX 950 sebesar 2 GB serta GPU yang disediakan oleh Google Colab.

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 serta menggunakan *Framework* Darknet.

4.2 Implementasi Praproses

Pada subbab ini akan dijelaskan proses implementasi praproses, yaitu pengubahan ukuran gambar menjadi 416 × 416, perubahan gambar RGB menjadi *grayscale*, dan penambahan data

dengan rotasi sebesar 1-2 derajat searah jam dan berlawanan arah jam.

```
def imgresize(imgfile):
  src = cv2.imread(imgfile, cv2.IMREAD_UNCHANGED)
  new_size = 416 #in pixel
  dsize = (new_size, new_size)
  output = cv2.resize(src, dsize, interpolation = cv2.INTER_AREA)
  cv2.imwrite(imgfile,output)
```

Kode Sumber 4.1 Implementasi fungsi *resize*

```
def imggray(imgfile):
  src = cv2.imread(imgfile, cv2.IMREAD_UNCHANGED)
  output = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)
  cv2.imwrite(imgfile,output)
```

Kode Sumber 4.2 Implementasi fungsi *grayscale transform*

```
def rotate_image(image, angle):
  image_center = tuple(np.array(image.shape[1::-1]) / 2)
  rot_mat = cv2.getRotationMatrix2D(image_center, angle, 1.0)
  result = cv2.warpAffine(image, rot_mat, image.shape[1::-1], flags=cv2.INTER_LINEAR)
  return result

def rotimg(imgfile, angle, addname):
  src = cv2.imread(imgfile, cv2.IMREAD_UNCHANGED)
  output = rotate_image(src, angle)
  newname = addname + imgfile
  cv2.imwrite(newname,output)
```

Kode Sumber 4.3 Implementasi fungsi rotasi

4.3 Implementasi Training

Pada subbab ini akan dijelaskan proses implementasi untuk pelatihan. Pelatihan menggunakan *framework* Darknet.

4.3.1 Persiapan Framework

Sebelum melakukan pelatihan framework yang digunakan dipersiapkan terlebih dahulu, yaitu dengan cara mengunduh framework, melakukan konfigurasi makefile, lalu melakukan compile.

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

```
%cd /content/gdrive/My\ Drive/
!git clone https://github.com/kriyeng/darknet/
%cd darknet
!ls
!git checkout feature/google-colab
```

Perintah untuk clone darknet

Hal pertama yang dilakukan untuk menyiapkan Darknet adalah mengunduhnya ke dalam Google Drive dan menggunakan Darknet di drive, karena jika hanya menggunakan local data akan hilang dalam beberapa jam. Setelah selesai mengunduh, ubah isi Makefile untuk mengatur beberapa pilihan:

- GPU = 1 untuk akselerasi menggunakan GPU
- CUDNN = 1 untuk akselerasi training menggunakan GPU
- OPENCV = 1 agar dapat mendeteksi file video
- LIBSO = 1 untuk membuat library yang dapat dipakai untuk pemanggilan fungsi

Perintah git checkout yang dilakukan adalah untuk mengubah branch darknet ke branch yang sudah dimodifikasi untuk berjalan di Google Colab, yaitu perubahan pada Makefile (GPU = 1, CUDNN = 1, OPENCV = 1) dan comment beberapa perintah print.

```
!apt install gcc-5 g++-5 -y
```

```
!ln -s /usr/bin/gcc-5 /usr/local/cuda/bin/gcc
!ln -s /usr/bin/g++-5 /usr/local/cuda/bin/g++
!make
```

Perintah untuk compile darknet

Untuk compile darknet diperlukan GCC dan G++, karena itu dilakukan instalasi GCC dan G++ terlebih dahulu. Untuk compile, jalankan perintah ‘make’ saat berada dalam folder darknet. Setelah compile, akan muncul file bernama ‘darknet’, yang akan dipakai untuk menjalankan perintah.

```
!chmod a+x ./darknet
```

Untuk menjalankan darknet, diperlukan permission. Karena itu ubah permission dari darknet terlebih dahulu.

4.3.2 Persiapan Training

Setelah framework siap digunakan, tahap selanjutnya adalah menyiapkan data yang diperlukan untuk training, yaitu file nama objek, file data training, file data testing, file data objek, file cfg, dan file pretrained weight.

File nama objek berisi list nama-nama kelas dari dataset. File data training adalah list lokasi dan nama gambar yang akan dipakai untuk training, data testing adalah list lokasi dan nama gambar yang akan dipakai untuk testing. File data objek adalah file yang menunjukkan dimana letak file nama objek, file data training, jumlah kelas, dan lokasi tempat menyimpan *weights*. File cfg adalah file yang berisi konfigurasi dari jaringan yang akan dipakai. File pretrained weight akan dipakai untuk mempercepat proses training.

Isi dari file nama objek adalah nama kelas di setiap baris, berurutan sesuai dengan urutan label. Untuk data Bisindo ini nama kelas adalah huruf A sampai Z kecuali huruf J dan R.

Untuk file data objek, diisi sebagai berikut:

- Classes = jumlah kelas (24)
- Train = file data testing (folder/test.txt)
- Names = file nama objek (folder/obj.names)

- Backup = folder lokasi penyimpanan (folder2)

Untuk file cfg sudah disediakan oleh darknet yaitu cfg/yolov3.cfg. Isi dari file cfg ini sudah sesuai untuk membangun jaringan, hanya perlu melakukan beberapa perubahan:

1. Baris 10: channels=1
2. Baris 18: learning_rate=0.001
3. Baris 603: filters=87
4. Baris 689: filters=87
5. Baris 776: filters=87

$$Filters = (C + 5) \times 3 \qquad (4.1)$$

Pengubahan channels = 3 menjadi 1 dilakukan karena data training memiliki jumlah channel 1. Perubahan jumlah filter dilakukan pada layer output didapatkan dari persamaan (4.1) dimana C adalah jumlah kelas, sehingga jumlah filter yang digunakan adalah (24 + 5) × 3 = 87. Nilai dari learning rate diubah sesuai dengan skenario yang dicoba.

```
!wget https://pjreddie.com/media/files/darknet
53.conv.74
```

File pretrained weights dapat diunduh secara langsung dari website pengembang YOLOv3, yang sudah dilatih pada Imagenet. File pretrained yang diunduh memiliki nama darknet53.conv.74.

4.3.3 Memulai Training

Training dapat dimulai dengan menjalankan darknet yang sudah dicompile dengan perintah sebagai berikut:
darknet detector train file_objek_data file_cfg file_pretrain -dont_show

```
!./darknet detector train bisindo/obj_full.dat
a yolov3_gs.cfg darknet53.conv.74 -dont_show
```

Opsi -dont_show berguna untuk tidak menampilkan window baru, dipakai karena colab tidak bisa memunculkan window baru, sehingga dapat menimbulkan error jika tidak menggunakan opsi tersebut.

Selama proses training berlangsung, akan ditampilkan nilai loss pada setiap iterasi pelatihan. Setiap 100 iterasi weight terbaru akan diupdate, dan akan disimpan file weights baru untuk setiap 1000 iterasi. Training dapat diberhentikan ketika nilai loss sudah tidak turun lagi.

Setelah beberapa waktu proses dalam colab dapat berhenti dengan sendirinya. Proses training dapat dilanjutkan dengan perintah yang sama dengan training, dan mengubah file pretrained menjadi file weights terbaru.

```
!./darknet detector train bisindo/obj_full.dat
a yolov3_gs.cfg backup/bisindo_full/yolov3_gs_
last.weights -dont_show
```

4.4 Implementasi Testing

Pengujian dibagi menjadi 2, yaitu pengujian untuk data gambar dan pengujian untuk video.

4.4.1 Pengujian Data Gambar

Untuk pengujian data gambar, Darknet akan digunakan untuk melakukan proses deteksi dan menghasilkan file .txt yang akan di*parse* untuk mendapatkan penghitungan hasil deteksi.

Untuk menjalankan proses deteksi, masukkan perintah berikut:

Darknet detector test file_objek_data file_cfg file_weights < file_data_testing > hasil.txt -thresh threshold -don't_show

```
!./darknet detector test bisindo/obj_full.data
 yolov3_gs.cfg backup/bisindo_full/yolov3_gs_l
```

```
ast.weights < bisindo/test_full.txt > result_b
isindofull_01.txt -thresh 0.01 -dont_show
```

Opsi -thresh berguna untuk menentukan threshold deteksi, dimana deteksi dengan *confidence* kurang dari threshold akan dibuang. Penulis memakai threshold sebesar 0.01 agar semua deteksi dengan *confidence* lebih dari sama dengan 0.01 akan dianggap sebagai deteksi, masuk ke proses non-max suppression, lalu mengeluarkan output.

Untuk implementasi parsing file hasil testing dan penghitungan, dapat dilihat pada kode sumber 4.4.

```
1.  alphabet = list('ABCDEFGHIKLMNOPQSTUVWXYZ')
2.
3.  def letter_to_int(letter):
4.      return alphabet.index(letter)
5.
6.  def cmatrix(filename, thresh):
7.    confmat = [[0 for i in range(24)] for j in range(
    24)]
8.    za_truth = [0 for i in range(24)]
9.    za_detect = [0 for i in range(24)]
10.   pre_val = [0 for i in range(24)]
11.   rec_val = [0 for i in range(24)]
12.   za_fn = [0 for i in range(24)]
13.   detected = []
14.   total_tp = 0
15.   total_precision = 0
16.   total_recall = 0
17.   total_ground_truth = 0
18.   total_detection = 0
19.   total_fn = 0
20.   found = True
21.   current_label = "A"
22.   current_loc = "idk"
23.   lab1 = 0 #ground truth label
24.   lab2 = 0 #detected label
25.   detecting_any = True
26.   f = open(filename)
```

```
  lines  = [line.rstrip("\n") for line in f.readlines()]

  for line in lines:
    line = line.replace('/', ' ')
    args = line.split(' ')
    if(args[0] == "Enter") :
      detected.sort(key=lambda x: x[1], reverse = True)
      if(len(detected) > 0):
        detecting_any = True
        lab2 = detected[0][0]
        total_detection += 1
        za_detect[lab2] += 1
        if(lab2==letter_to_int(current_label)):
          confmat[lab1][lab2] += 1
        else:
          print(lab2 + " - bad, " + current_loc + args[1])
          confmat[lab1][lab2] += 1
      if(detecting_any == False):
        print("no detection: " + current_loc)
      if(len(args) < 9):  #check last line
        break
      current_label = args[12]
      current_loc = args[10] + "/" + args[12] + "/" + args[13]
      lab1 = letter_to_int(current_label)
      total_ground_truth += 1
      za_truth[lab1] += 1
      detecting_any = False
      detected.clear()
    elif(args[0]!=''):
      if(args[0][1]==":" and int(args[1][:-1]) >= thresh):
        temp = letter_to_int(args[0][0]), int(args[1][:-1])
        detected.append(temp)

  for line in confmat:
    print(line)

  print("---------------------------")
```

```
64.   print("Ground Truth: " + str(total_ground_truth))
65.   print("Detected: " + str(total_detection))
66.   print("Total True Positive: " + str(total_tp))
67.
68.   for i in range(24):
69.     pre_val[i] = confmat[i][i] / za_detect[i]
70.     rec_val[i] = confmat[i][i] / za_truth[i]
71.     za_fn[i] = za_truth[i] - confmat[i][i]
72.     total_tp += confmat[i][i]
73.     total_fn += za_fn[i]
74.     total_precision += pre_val[i]
75.     total_recall += rec_val[i]
76.
77.   for i in range(24):
78.     print("class_id = " + str(i) + "\tname = " +
79.           alphabet[i] + "\t Detected = " +
80.           str(za_detect[i]) + ",TP = " +
81.           str(confmat[i][i]) + ",Truth = " +
82.           str(za_truth[i]) + ",Precision = " +
83.           format(pre_val[i],'.2f') + ",Recall = " +
84.           format(rec_val[i], '.2f'))
85.
86.   avg_pre = total_precision/24
87.   print("Average Precision = " + str(avg_pre))
88.   avg_rec = total_recall/24
89.   print("Average Recall = " + str(avg_rec))
90.   accu = total_tp/(total_detection + total_fn)
91.   print("Accuracy = " + str(accu))
92.   fone = 2 * (avg_pre * avg_rec) / (avg_pre + avg_r
    ec)
93.   print("F1 Score = " + str(fone))
```

Kode Sumber 4.4 Implementasi parsing file txt dan penghitungan kinerja

Berikut adalah contoh bagian txt yang akan diparse:

```
seen 64
Enter Image Path: /content/gdrive/My
Drive/darknet/dataset/bisindo/test/A/rot2cwIMG
_20200403_103133_1.jpg: Predicted in 16.976000
milli-seconds.
A: 99%
```

```
Enter Image Path: /content/gdrive/My
Drive/darknet/dataset/bisindo/test/A/IMG_20200
403_103134_1.jpg: Predicted in 16.913000
milli-seconds.
A: 94%
Enter Image Path: /content/gdrive/My
Drive/darknet/dataset/bisindo/test/A/rot1cwIMG
_20200217_213802_1.jpg: Predicted in 16.968000
milli-seconds.
A: 99%
Enter Image Path: /content/gdrive/My
Drive/darknet/dataset/bisindo/test/A/rot1cwIMG
_20200217_213758.jpg: Predicted in 16.963000
milli-seconds.
A: 99%
S: 4%
Enter Image Path: /content/gdrive/My
Drive/darknet/dataset/bisindo/test/A/rot1cwIMG
_20200403_103109.jpg: Predicted in 17.060000
milli-seconds.
A: 99%
Enter Image Path:
```

Pola dari fie .txt ini adalah ‘Enter Image Path’ diikuti dengan lokasi gambar yang akan dideteksi, lalu hasil prediksi berada di line baru dengan format ‘Kelas: *confidence*’. Jika disesuaikan dengan ground truth maka tiap gambar harus diikuti oleh 1 prediksi objek yang memliki *confidence* diatas threshold dan memiliki kelas yang sama dengan gambar testing. Semua prediksi disimpan dalam variabel *detected*, lalu diambil yang memiliki *confidence* tertinggi. Deteksi yang sesuai dengan ground truth dan deteksi yang salah akan dimasukkan ke dalam *confusion matrix* dengan cara *increment* nilai pada array 2 dimensi pada indeks [*ground truth*][*prediction*]. Jumlah deteksi yang dihitung adalah hasil deteksi yang memiliki nilai di atas *threshold*, dan jumlah ground truth memiliki nilai sama dengan jumlah gambar

testing. Dari hasil tersebut dapat dihitung presisi, *recall*, akurasi, dan *F1 Score*.

4.4.2 Pengujian Data Video

Implementasi untuk pengujian data video dilakukan di laptop, karena google colab tidak dapat membuka window baru untuk menampilkan hasil deteksi video. Implementasi menggunakan library hasil compile Darknet. Contoh penggunaan library sudah disediakan oleh pengembang darknet dalam file python darknet.py, dan dalam file tersebut sudah didefinisikan fungsi untuk deteksi.

```
1.  def detect_image(net, meta, im, thresh=.5, hier_thr
    esh=.5, nms=.45, debug= False):
2.      num = c_int(0)
3.      pnum = pointer(num)
4.      predict_image(net, im)
5.      letter_box = 0
6.      dets = get_network_boxes(net, im.w, im.h, thres
    h, hier_thresh, None, 0, pnum, letter_box)
7.      num = pnum[0]
8.      if nms:
9.          do_nms_sort(dets, num, meta.classes, nms)
10.     res = []
11.     for j in range(num):
12.         for i in range(meta.classes):
13.             if dets[j].prob[i] > 0:
14.                 b = dets[j].bbox
15.                 if altNames is None:
16.                     nameTag = meta.names[i]
17.                 else:
18.                     nameTag = altNames[i]
19.                 res.append((nameTag, dets[j].prob[i
    ], (b.x, b.y, b.w, b.h)))
20.     res = sorted(res, key=lambda x: -x[1])
21.     free_detections(dets, num)
22.     res2 = []
23.     if(len(res)>0):
24.         res2.append(res[0])
```

```
25.     return res2
```

Kode Sumber 4.5 Implementasi fungsi pendeteksian pada gambar

Fungsi pada kode sumber 5.4 merupakan salah satu fungsi yang terdapat pada darknet.py, yang berguna untuk melakukan pendeteksian. Hasil deteksi pertama kali dihasilkan dari fungsi `get_network_boxes`. Hasil tersebut lalu masuk ke dalam fungsi do_nms_sort yang melakukan non-max suppression. Hasil output yang dihasilkan lalu pada setiap prediksi dimasukkan ke dalam array res dengan format: kelas, *confidence*, dan *bounding box* yang berisi center x, center y, width, dan heights. Array res lalu diurutkan berdasarkan *confidence*. Untuk pengujian dilakukan pembatasan hanya satu deteksi tiap frame, dengan cara hanya mengembalikan *bounding box* dengan nilai *confidence* terbesar. Hal ini dilakukan dengan return dari fungsi detect_image ini tidak mengembalikan array res, melainkan res2 yang hanya berisi 1 *bounding box* dengan *confidence* tertinggi.

```
1.  from ctypes import *
2.  import math
3.  import random
4.  import os
5.  import cv2
6.  import numpy as np
7.  import time
8.  import darknet
9.
10. def convertBack(x, y, w, h):
11.     xmin = int(round(x - (w / 2)))
12.     xmax = int(round(x + (w / 2)))
13.     ymin = int(round(y - (h / 2)))
14.     ymax = int(round(y + (h / 2)))
15.     return xmin, ymin, xmax, ymax
16.
17. def cvDrawBoxes(detections, img):
18.     for detection in detections:
19.         x = detection[2][0]
```

```
20.         y = detection[2][1]
21.         w = detection[2][2]
22.         h = detection[2][3]
23.         xmin, ymin, xmax, ymax = convertBack(
24.                 float(x), float(y), float(w), float
    (h))
25.         pt1 = (xmin, ymin)
26.         pt2 = (xmax, ymax)
27.         cv2.rectangle(img, pt1, pt2, (0, 255, 0), 1
    )
28.         cv2.putText(img,
29.                 detection[0].decode() +
30.                 " [" + str(round(detection[1] * 100
    , 2)) + "]",
31.                 (pt1[0], pt1[1] -
     5), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
32.                 [0, 255, 0], 2)
33.     return img
34.
35. def detect_image(net, meta, im, thresh=.5, hier_thr
    esh=.5, nms=.45, debug= False):
36.     num = c_int(0)
37.     pnum = pointer(num)
38.     darknet.predict_image(net, im)
39.     letter_box = 0
40.     dets = darknet.get_network_boxes(net, im.w, im.
    h, thresh, hier_thresh, None, 0, pnum, letter_box)
41.     num = pnum[0]
42.     if nms:
43.         darknet.do_nms_sort(dets, num, meta.classes
    , nms)
44.     res = []
45.     for j in range(num):
46.         for i in range(meta.classes):
47.             if dets[j].prob[i] > 0:
48.                 b = dets[j].bbox
49.                 nameTag = meta.names[i]
50.                 res.append((nameTag, dets[j].prob[i
    ], (b.x, b.y, b.w, b.h)))
51.     res = sorted(res, key=lambda x: -x[1])
52.     darknet.free_detections(dets, num)
53.     res2 = []
54.     if(len(res)>0):
```

```
55.         res2.append(res[0])
56.     return res2
57.
58. def YOLO():
59.
60.     configPath = "./yolov3_gs.cfg"
61.     weightPath = "./yolov3_gs_last.weights"
62.     metaPath = "./bisindo/obj.data"
63.     netMain = darknet.load_net_custom(configPath.en
    code("ascii"), weightPath.encode("ascii"), 0, 1)
64.     metaMain = darknet.load_meta(metaPath.encode("a
    scii"))
65.
66.     usewebcam = False
67.     skipframe = 1
68.     if(usewebcam):
69.         #cap = cv2.VideoCapture(-1)  #webcam
70.         cap = cv2.VideoCapture("http://192.168.100.
    25:8080/video") #android ip cam
71.     else:
72.         cap = cv2.VideoCapture("tests/wax.mp4")
73.     print("Starting the YOLO loop...")
74.
75.     darknet_image = darknet.make_image(darknet.netw
    ork_width(netMain),darknet.network_height(netMain),
    1)
76.     fcount = 0
77.     streak = 0
78.     streak_limit = 7
79.     lastchar = ''
80.     last_streak_char = ''
81.     detected_char = []
82.     prev_time = time.time()
83.     while(True):
84.         ret, frame_read = cap.read()
85.         if(ret == False):
86.             print("video ends")
87.             break
88.         if(fcount%skipframe == 0):
89.             frame_resized = cv2.resize(frame_read,(
    darknet.network_width(netMain), darknet.network_hei
    ght(netMain)),
90.                     interpolation=cv2.INTER_LINEAR)
```

```
91.             frame_gray = cv2.cvtColor(frame_resized
    , cv2.COLOR_BGR2GRAY)
92.
93.             darknet.copy_image_from_bytes(darknet_i
    mage,frame_gray.tobytes())
94.
95.             detections = detect_image(netMain, meta
    Main, darknet_image, thresh=0.15)
96.             if(len(detections) > 0):
97.                 char = str(detections[0][0].decode(
    ))
98.                 #print("detect " + char)
99.                 print(char)
100.                    else:
101.                        char = ''
102.                        #print("no detection")
103.                        print('0')
104.                    image = cvDrawBoxes(detections,
    frame_resized)
105.                    if(lastchar == char):
106.                        streak += 1
107.                        if(streak == streak_limit an
    d char != last_streak_char):
108.                            last_streak_char = char
109.                            if(char != ''):
110.                                detected_char.append
    (char)
111.                    else:
112.                        streak = 1
113.                    lastchar = char
114.                    cv2.imshow('Demo', image)
115.                    #print(1/(time.time()-
    prev_time))  #print fps
116.                    prev_time = time.time()
117.                    if cv2.waitKey(1) & 0xFF == ord(
    'q'):
118.                        print("video ends")
119.                        break
120.                fcount += 1
121.            cap.release()
122.            print(detected_char)
```

Kode Sumber 4.6 Implementasi pengenalan isyarat pada video

Implementasi pengenalan isyarat pada data video dapat dilihat pada kode sumber 4.6. Fungsi `convertBack` berguna untuk mengubah nilai *bounding box* dari center x, center, y, width, height menjadi x-min, y-min, x-max, dan y-max.

Fungsi `cvDrawBoxes` berguna untuk menggambarkan *bounding box* untuk output image.

Fungsi YOLO menjalankan pembacaan file cfg, file weight, dan membaca file data objek. File cfg akan dipakai untuk membangun jaringan sesuai dengan konfigurasi dalam file cfg tersebut.

Pembacaan video dapat melalui berbagai sumber, yaitu file video, webcam, dan stream video. Pembacaan sumber video dilakukan dengan fungsi `cv2.VideoCapture`. Jika sumber video adalah file, ambil jumlah frame untuk menjadi penanda bahwa video telah selesai, sedangakan untuk webcam hanya dapat dihentikan dengan trigger dari pengguna, dalam implementasi ini adalah tombol ‘q’ pada keyboard.

Fungsi dari penggunaan skipframe adalah agar deteksi dan penampilan hasil dilakukan pada setiap kelipatan angka yang ditentukan. Tujuan penggunaaan skipframe ini untuk webcam, karena kecepatan input masuk lebih cepat dari kecepatan membaca, mendeteksi, dan menampilkan frame, sehingga perlu ada frame yang tidak dideteksi dan ditampilkan agar tidak terjadi penumpukan frame yang perlu dibaca, menyebabkan lag/ jeda yang semakin besar antara input kamera dengan output hasil deteksi.

Pada setiap frame yang dibaca dan akan dideteksi, dibuat sebuah gambar dari frame tersebut lalu di*resize* dan diubah ke *grayscale*. Gambar tersebut lalu dimasukkan sebagai input ke dalam fungsi deteksi, yang akan mengeluarkan output yang berisi nama kelas, *confidence*, dan *bounding box*. Setiap hasil deteksi dimasukkan sebagai input untuk fungsi `cvDrawBoxes`, yang akan mengembalikan gambar frame ditambah dengan *bounding box* hasil prediksi. Hasil dari fungsi tersebut lalu ditampilkan lewat fungsi `cv2.imShow`.

Pada kode ini juga dilakukan pengecekan deteksi. Apabila deteksi pada objek yang sama terjadi berurutan mencapai batas `streaklimit` yang ditentukan, maka objek tersebut akan ditambahkan ke dalam array yang dianggap sebagai isyarat yang dikenal dalam video.

(Halaman ini sengaja dikosongkan)

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 laptop ASUS X550VX. Sistem operasi yang digunakan adalah Windows 10 64-bit. PC yang digunakan memiliki spesifikasi Intel i7 6700 dengan kecepatan 2.6 GHz, *Random Access Memory* (RAM) sebesar 8 GB, dan mempunyai *Graphics Processing Unit* (GPU) yaitu NVIDIA GeForce GTX 950 sebesar 2 GB serta GPU yang disediakan oleh Google Colab. 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 serta menggunakan *Framework* Darknet.

5.2 Deskripsi Dataset

Dataset yang dipakai dalam pengujian sistem pengenalan isyarat adalah dataset yang diambil oleh penulis berupa foto dan video, terdiri dari 24 kelas di mana pada masing-masing kelas terdapat sekitar 160 sampai dengan 220 citra. Data ini merupakan sekumpulan citra karakter A sampai dengan Z kecuali J dan R. Spesifikasi Data uji coba gambar dapat dilihat pada Tabel 5.1.

Tabel 5.1 Spesifikasi data uji coba gambar

Keterangan	**Spesifikasi**
Ukuran resolusi	416 × 416
Ekstensi	.jpg
Jumlah gambar	4,547
Jumlah kelas	24 kelas
Jumlah gambar per kelas	160-220

Keterangan	**Spesifikasi**
Ukuran file	50-500 kB
Kanal warna	1 (*Grayscale*)

5.3 Skenario Uji Coba

Pada subbab ini, terdapat beberapa skenario uji coba:

1. Hasil Pelatihan
   Uji coba dilakukan dalam proses pembentukan model, dimana pelatihan dilakukan menggunakan variasi pada parameter *learning rate*. Penilaian dilakukan dengan perbandingan nilai terendah dari *training loss* yang didapat saat pelatihan.
2. Hasil Uji Coba pada data gambar
   Uji coba pada data gambar dilakukan dengan cara melakukan pengenalan pada seluruh gambar dari data testing. Hasil prediksi akan dibandingkan dengan *ground truth* yang sudah disediakan penulis. Penilaian akan dilakukan dengan menggunakan *precision, recall, accuracy,* dan *F1 score*.
3. Hasil Uji Coba pada data video
   Uji coba pada data video dilakukan dengan cara melakukan pengenalan pada data video yang diambil oleh penulis. Hasil prediksi akan dibandingkan dengan *ground truth* yang disediakan penulis. Penilaian yang dilakukan adalah kecepatan deteksi diukur dengan *FPS*, dan ketepatan prediksi.

5.3.1 Hasil Pelatihan

Pelatihan dilakukan dengan membentuk tiga model, dengan parameter *learning rate* yang berbeda, yaitu 0.1, 0.01, dan 0.001. Pelatihan model dilakukan sampai nilai *loss* tidak mengalami penurunan. Setiap model dibandingkan berdasarkan nilai *loss* terendah yang didapat. Hasil pelatihan dapat dilihat pada Tabel 5.2.1

Tabel 5.2 Hasil training dengan variasi *learning rate*

Learning Rate	Loss
0.1	0.942662
0.01	0.15379
0.001	**0.143697**

Dari Tabel 5.2 didapatkan bahwa pelatihan model menggunakan learning rate sebesar 0.001 memiliki nilai *loss* paling rendah. Model ini akan digunakan untuk skenario uji coba lainnya.

5.3.2 Hasil Uji Coba pada Data Gambar

Uji coba pada data gambar dilakukan dengan cara melakukan pengenalan pada seluruh gambar dari data testing. Hasil prediksi akan dibandingkan dengan *ground truth* yang sudah disediakan penulis. Pada satu gambar akan dibatasi 1 deteksi dengan *confidence* tertinggi. Penilaian akan dilakukan dengan menggunakan *precision*, *recall*, *accuracy*, dan *F1 score*. Hasil uji coba dapat dilihat pada Tabel 5.3.

Tabel 5.3 Hasil uji coba pengenalan pada data gambar

Threshold (%)	Precision (%)	Recall (%)	Accuracy (%)	F1 Score (%)
100	100	41.67	42.20	58.83
90	100	76.53	76.97	86.71
80	100	83	83.33	90.71
70	100	87.72	87.90	93.46
60	100	91.56	91.62	95.59
50	100	94.53	94.57	97.19
40	100	96.70	96.72	98.32
30	100	98.27	98.28	99.13

Threshold (%)	Precision (%)	Recall (%)	Accuracy (%)	F1 Score (%)
20	100	99.51	99.52	99.75
10	100	99.98	99.98	99.99
9	100	100	100	100
8	100	100	100	100
7	100	100	100	100
6	100	100	100	100
5	100	100	100	100
4	100	100	100	100
3	100	100	100	100
2	100	100	100	100
1	100	100	100	100

Dapat dilihat dari nilai presisi pada Tabel 5.3, semua deteksi yang dilakukan menghasilkan kelas yang tepat. Tetapi, terdapat beberapa gambar yang memiliki *confidence* rendah sehingga tidak terdeteksi pada threshold tertentu.

5.3.3 Hasil Uji Coba pada Data Video

Uji coba video dilakukan pada Laptop penulis, agar dapat menampilkan hasil prediksi. Threshold yang dipakai adalah 15%. Penghitungan FPS (*Frame Per Second*) dilakukan dengan menghitung waktu yang diperlukan sebelum proses gambar untuk deteksi sampai sebelum menampilkan gambar prediksi.

Jumlah objek yang akan keluar sebagai deteksi dibatasi menjadi 1 dengan confindence tertinggi. Deteksi isyarat secara berurut mencapai batas yang ditentukan akan dianggap sebagai isyarat yang terdeteksi, dan akan dibandingkan dengan ground truth yang disediakan.

Uji coba dilakukan terhadap sejumlah video yang diambil oleh penulis, terpisah dengan data yang diambil dengan training

dengan kondisi yang mirip dengan kondisi 5 (pengambilan jarak jauh). Spesifikasi video testing dapat dilihat pada Tabel 5.4.

Tabel 5.4 Spesifikasi video untuk uji coba

Video	Target Huruf	Jumlah Frame
1	GLEAM	291
2	IVY	109
3	NICE	103
4	OK	103
5	SHIZA	233
6	WAX	130
7	BDFPUV	392
8	QUEST	233

Setiap video testing memiliki dimensi yang sama, yaitu 640 × 480. Video terdiri dari dua atau lebih isyarat yang ditampilkan secara bersambung, membentuk rangkaian huruf sesuai dengan target huruf dari video tersebut.

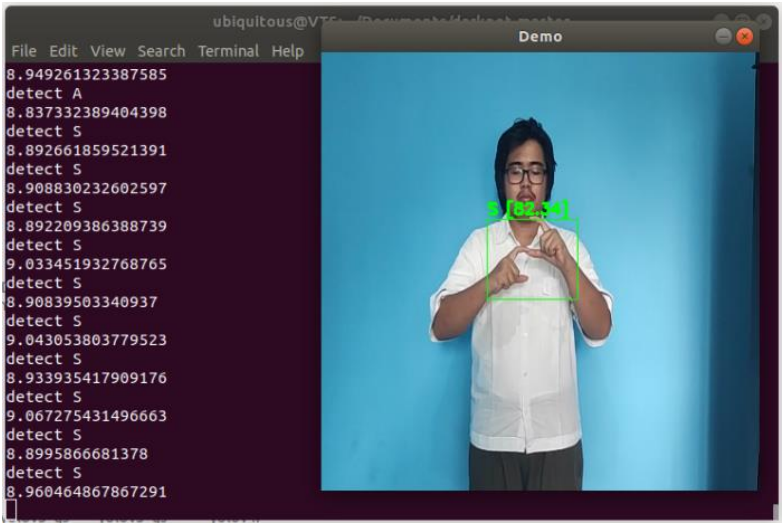
Hasil dari uji coba pada data video menunjukkan nilai presisi sebesar 77.14%, *recall* sebesar 93.1%, akurasi sebesar 72.97%, dan *F1 score* sebesar 84.38% pada *streak limit* sebesar 7. Detil dari hasil uji coba dapat dilihat pada Tabel 5.5.

Tabel 5.5 Hasil uji coba pengenalan pada data video

Streak Limit	Target Huruf	Hasil Deteksi	Keterangan
6	GLEAM	GLEAM	-
	IVY	OIOY	kesalahan prediksi awal I menjadi O, dan V menjadi O
	NICE	MICE	kesalahan prediksi M menjadi N

Streak Limit	Target Huruf	Hasil Deteksi	Keterangan
	SHIZA	SLZA	pergantian prediksi D antara D dan H tanpa deteksi berurut sebanyak 6, kesalahan prediksi I menjadi L
	WAX	WADA	kesalahan prediksi awal huruf X menjadi D, lalu menjadi A
	BDFPUV	BPDBPD UVUV	Kesalahan prediksi D menjadi P, F menjadi B, P menjadi D, U menjadi V
	QUEST	QUEIST	deteksi I secara berurut mencapai 6 saat perubahan isyarat dari E ke S
7	WAX	WA	Kesalahan deteksi X menjadi A
	BDFPUV	BPDBPD VUV	Kesalahan prediksi D menjadi P, F menjadi B, P menjadi D, U menjadi V
	QUEST	QUEST	jumlah deteksi I secara berurut berkurang, di bawah 7

Dari hasil uji coba yang ditampilkan pada Tabel 5.5, diapatkan dua kesalahan yang terjadi. Kesalahan pertama adalah kesalahan prediksi kelas, seperti pada huruf ‘N’ pada terget ‘NICE’ dikenal sebagai ‘M’, juga huruf ‘H’ pada terget ‘SHIZA’ tidak dikenal karena hasil prediksi yang didapatkan terus berganti antara ‘H’ dengan ‘D’ tanpa adanya prediksi yang berurutan mencapai *streak limit* yang ditentukan. Kesalahan kedua adalah pengenalan isyarat yang terjadi saat transisi dari satu isyarat ke isyarat yang lain, dengan contoh adanya pengenalan isyarat ‘I’ pada target ‘QUEST’ pada saat transisi antara isyarat ‘E’ menuju ‘S’.



Gambar 5.1 Salah satu frame hasil deteksi dan print fps

Salah satu frame hasil deteksi dapat dilihat pada Gambar 5.11, yang menampilkan hasil prediksi isyarat ‘S’ dengan *confidence* sebesar 82.34%. Angka pada gambar tersebut menunjukkan FPS, yang berkisar 8-9 frame per detik.

5.4 Hasil dan Evaluasi

Pada hasil uji coba data gambar, setelah diteliti didapatkan kelompok kelas yang memiliki *confidenece* di bawah threshold, dapat dilihat dari Tabel 5.6.

Tabel 5.6 Detil pada kasus tidak terdeteksi

	Threshold								
Class	80%	70%	60%	50%	40%	30%	20%	10%	1%
A	3	1	0	0	0	0	0	0	0
B	12	1	0	0	0	0	0	0	0
C	58	43	21	11	4	1	0	0	0
D	47	26	9	8	6	4	1	0	0
E	51	29	15	3	2	1	0	0	0
F	23	12	7	0	0	0	0	0	0
G	61	37	17	7	4	3	0	0	0
H	30	16	8	7	2	0	0	0	0
I	67	53	33	23	7	1	0	0	0
K	2	0	0	0	0	0	0	0	0
L	60	38	27	15	7	4	0	0	0
M	43	32	20	9	4	1	0	0	0
N	64	53	32	10	1	0	0	0	0
O	**70**	**70**	**70**	**69**	**64**	**48**	**21**	**1**	0
P	63	63	63	55	37	13	0	0	0
Q	69	66	56	29	11	2	0	0	0
S	9	1	0	0	0	0	0	0	0
T	5	0	0	0	0	0	0	0	0
U	9	2	0	0	0	0	0	0	0
V	4	3	2	1	0	0	0	0	0
W	0	0	0	0	0	0	0	0	0
X	3	2	0	0	0	0	0	0	0
Y	5	2	1	0	0	0	0	0	0
Z	0	0	0	0	0	0	0	0	0
Total	758	550	381	247	149	78	22	1	0

Didapati bahwa kelas O memiliki jumlah terbesar dalam deteksi dengan *confidence* di bawah threshold 80% sampai 10%. Hasil deteksi yang memiliki *confidence* terkecil adalah pada kelas O dengan *confidence* sebesar 9%.

Dari total 247 kasus gambar tidak terdeteksi pada threshold 50%, dapat dibagi berdasarkan kondisi gambar. Detil kasus tidak terdeteksi ditunjukkan pada Tabel 5.7.

Tabel 5.7 Detil kasus tidak terdeteksi pada threshold 50%

Kelas	K5	K4	K3	K2	K1	TOTAL
A	0	0	0	0	0	0
B	0	0	0	0	0	0
C	11	0	0	0	0	11
D	8	0	0	0	0	8
E	3	0	0	0	0	3
F	0	0	0	0	0	0
G	6	0	1	0	0	7
H	7	0	0	0	0	7
I	23	0	0	0	0	23
K	0	0	0	0	0	0
L	15	0	0	0	0	15
M	9	0	0	0	0	9
N	10	0	0	0	0	10
O	**69**	0	0	0	0	69
P	55	0	0	0	0	55
Q	29	0	0	0	0	29
S	0	0	0	0	0	0
T	0	0	0	0	0	0
U	0	0	0	0	0	0
V	1	0	0	0	0	1

Kelas	K5	K4	K3	K2	K1	TOTAL
W	0	0	0	0	0	0
X	0	0	0	0	0	0
Y	0	0	0	0	0	0
Z	0	0	0	0	0	0
TOTAL	246	0	1	0	0	247

Dapat dilihat dari Tabel 5.7 bahwa sebagian besar dari kasus tidak terdeteksi terjadi pada data kondisi 5, yaitu dimana jarak pengambilan yang terbesar, dan objek yang diambil kecil dibandingkan dengan data kondisi lain. Kelas O yang memiliki jumlah kasus tidak terdeteksi paling banyak, semua kasus berasal dari gambar yang memiliki kondisi 5.

Untuk deteksi video, kecepatan berkisar antara 8-9 FPS untuk input video dengan dimensi 640 × 480. Kecepatan 8 FPS tidak dapat memproses seluruh frame input *live* yang umumnya berjumlah 20-30 frame setiap detiknya secara *real-time*, sehingga terjadi penumpukan frame yang perlu diproses. Karena itu, untuk penggunaan secara real-time, perlu adanya pemilihan frame yang dilakukan pengenalan agar frame yang dikenali oleh sistem adalah frame terbaru yang baru didapat dari input. Sebagai contoh, jika sistem memiliki kecepatan 8 FPS dan input sebesar 24 FPS, maka frame yang akan dideteksi adalah frame kelipatan 3. Dengan deteksi pada 8 frame tiap detik dan batas deteksi berurutan adalah 7, pendeteksian dapat dilakukan secara langsung/*live* (contoh: menggunakan webcam) dengan penggunaan isyarat mendekati kecepatan natural dimana isyarat dilakukan sekitar 1 detik untuk setiap huruf.

Kesalahan prediksi pada terjadi diperkirakan pada kelas N dan M karena mempunyai bentuk isarat yang mirip, begitu juga P dengan D, U dengan V, I dengan L pada jarak jauh, dan W dengan A pada jarak jauh.

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas tentang kesimpulan yang didasari oleh hasil uji coba yang telah dilakukan terhadap *YOLO* 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. Pengenalan bahasa isyarat dengan arsitektur YOLO berhasil diimplementasikan menggunakan Darknet dan didapatkan model terbaik memiliki nilai *loss* terendah sebesar 0.143697 pada parameter *learning rate* sebesar 0.001.
2. Berdasarkan uji coba pengenalan Bahasa isyarat pada data gambar, didapatkan nilai tertinggi untuk presisi, recall, akurasi, dan *F1 score* sebesar 100% pada threshold 9%.
3. Berdasarkan uji coba pengenalan Bahasa isyarat pada data video, didapatkan nilai presisi sebesar 77.14%, *recall* sebesar 93.1%, akurasi sebesar 72.97%, dan *F1 score* sebesar 84.38%.
4. Berdasarkan uji coba pengenalan Bahasa isyarat pada data video, deteksi secara *real-time* dapat dilakukan dengan cara menyesuaikan kecepatan deteksi dengan jumlah pendeteksian frame setiap detik yang dilakukan.
5. Hasil uji coba pengenalan bahasa isyarat masih memunculkan kesalahan pada deteksi, yaitu saat isyarat yang terdeteksi berukuran kecil atau isyarat memiliki kemiripan dengan isyarat lain.

6.2 Saran

Saran yang diberikan untuk pengembangan sistem pengenalan bahasa isyarat pada data video, yaitu:

1. Mencoba Input network dengan dimensi yang lebih tinggi, seperti 608 × 608 untuk pendeteksian objek yang berukuran kecil dan mendapatkan detil dari bahasa isyarat lebih baik.
2. Menambah variasi data dalam kondisi yang berbeda agar dapat menangani kondisi saat penggunaan sehari-hari.
3. Melakukan percobaan dalam mengubah parameter saat training seperti *momentum* dan *decay* untuk pembuatan model yang lebih baik.

DAFTAR PUSTAKA

[1] J. Redmon, S. Divvala, R. Girshick dan A. Fahradi, "You only look once: Unified, real-time object detection," *Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 779-788, 2016.

[2] Hendry dan R.-C. Chen, "Automatic License Plate Recognition via sliding-window darknet-YOLO deep learning," *Image and Vision Computing,* vol. 87, pp. 47-56, 2019.

[3] H. Y. T. S. Z. a. Z. Y. Qu, "A Pedestrian Detection Method Based on YOLOv3 Model and Image Enhanced by Retinex," *11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI),* pp. 1-5, 2018.

[4] P. S. M. M. W. B. K. S. K. G. A. K. K. a. M. E.-M. Zaki, "Traffic Signs Detection and Recognition System using Deep Learning," *arXiv preprint arXiv:2003.03256,* 2020.

[5] B. Garcia dan S. A. Viesca, "Real-time American Sign Language Recognition with Convolutional Neural Networks," *Convolutional Neural Networks for Visual Recognition,* vol. 2, 2016.

[6] S. Sena, "Medium," 13 Novermber 2017. [Online]. Available: https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94. [Diakses 18 December 2019].

[7] J. a. F. A. Redmon, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767.,* 2018.

[8] A. Kathuria, "What's new in YOLO v3?," 23 April 2018. [Online]. Available: https://towardsdatascience.com/yolo-v3-object-

detection-53fb7d3bfe6b. [Diakses 22 June 2020].

[9] R. Balsys, "YOLO v3 theory explained," 17 July 2019. [Online]. Available: https://pylessons.com/YOLOv3-introduction. [Diakses 21 June 2020].

[10] A. Padmanabhan, "ImageNet," 25 October 2019. [Online]. Available: https://devopedia.org/imagenet. [Diakses 11 August 2020].

[11] K. Sambasivarao, "Medium," 1 October 2019. [Online]. Available: https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c. [Diakses 18 December 2019].

[12] S. Yegulalp, "InfoWorld," 1 June 2018. [Online]. Available: https://www.infoworld.com/article/3204016/what-is-python.html. [Diakses 18 December 2019].

[13] J. Redmon, "Darknet: Open Source Neural Networks in C," [Online]. Available: https://pjreddie.com/darknet/. [Diakses 22 June 2020].

[14] Wikipedia, "NumPy," [Online]. Available: http://www.numpy.org. [Diakses 22 June 2020].

[15] "OpenCV," [Online]. Available: https://opencv.org/. [Diakses 22 June 2020].

[16] Wikipedia, "Bahasa Isyarat Indonesia," [Online]. Available: https://id.wikipedia.org/wiki/Bahasa_Isyarat_Indonesia. [Diakses 21 June 2020].

[17] P. K. ABK, "ypedulikasihabk," 9 November 2018. [Online]. Available: https://www.ypedulikasihabk.org/2018/11/09/mengenal-bahasa-isyarat/. [Diakses 18 December 2019].

LAMPIRAN

L.1. Hasil uji coba gambar pada threshold 100%

Class	Predicted	Truth	TP	FP	FN
A	78	187	78	0	109
B	29	176	29	0	147
C	20	169	20	0	149
D	91	187	91	0	96
E	32	173	32	0	141
F	60	187	60	0	127
G	38	174	38	0	136
H	104	191	104	0	87
I	75	177	75	0	102
K	56	183	56	0	127
L	72	189	72	0	117
M	128	207	128	0	79
N	107	190	107	0	83
O	71	187	71	0	116
P	76	192	76	0	116
Q	110	202	110	0	92
S	118	194	118	0	76
T	88	193	88	0	105
U	89	187	89	0	98
V	82	191	82	0	109
W	137	192	137	0	55
X	79	198	79	0	119
Y	79	213	79	0	134
Z	100	208	100	0	108

L.2. Hasil uji coba gambar pada threshold 90%

Class	Predicted	Truth	TP	FP	FN
A	158	187	158	0	29
B	154	176	154	0	22
C	95	169	95	0	74
D	121	187	121	0	66
E	96	173	96	0	77
F	132	187	132	0	55
G	92	174	92	0	82
H	151	191	151	0	40
I	106	177	106	0	71
K	175	183	175	0	8
L	118	189	118	0	71
M	145	207	145	0	62
N	120	190	120	0	70
O	117	187	117	0	70
P	127	192	127	0	65
Q	132	202	132	0	70
S	178	194	178	0	16
T	167	193	167	0	26
U	158	187	158	0	29
V	171	191	171	0	20
W	192	192	192	0	0
X	186	198	186	0	12
Y	202	213	202	0	11
Z	207	208	207	0	1

L.3. Hasil uji coba gambar pada threshold 80%

Class	Detected	Truth	TP	FP	FN
A	184	187	184	0	3
B	164	176	164	0	12
C	111	169	111	0	58
D	140	187	140	0	47
E	122	173	122	0	51
F	164	187	164	0	23
G	113	174	113	0	61
H	161	191	161	0	30
I	110	177	110	0	67
K	181	183	181	0	2
L	129	189	129	0	60
M	164	207	164	0	43
N	126	190	126	0	64
O	117	187	117	0	70
P	129	192	129	0	63
Q	133	202	133	0	69
S	185	194	185	0	9
T	188	193	188	0	5
U	178	187	178	0	9
V	187	191	187	0	4
W	192	192	192	0	0
X	195	198	195	0	3
Y	208	213	208	0	5
Z	208	208	208	0	0

L.4. Hasil uji coba gambar pada threshold 70%

Class	Detected	Truth	TP	FP	FN
A	186	187	186	0	1
B	175	176	175	0	1
C	126	169	126	0	43
D	161	187	161	0	26
E	144	173	144	0	29
F	175	187	175	0	12
G	137	174	137	0	37
H	175	191	175	0	16
I	124	177	124	0	53
K	183	183	183	0	0
L	151	189	151	0	38
M	175	207	175	0	32
N	137	190	137	0	53
O	117	187	117	0	70
P	129	192	129	0	63
Q	136	202	136	0	66
S	193	194	193	0	1
T	193	193	193	0	0
U	185	187	185	0	2
V	188	191	188	0	3
W	192	192	192	0	0
X	196	198	196	0	2
Y	211	213	211	0	2
Z	208	208	208	0	0

L.5. Hasil uji coba gambar pada threshold 60%

Class	Detected	Truth	TP	FP	FN
A	187	187	187	0	0
B	176	176	176	0	0
C	148	169	148	0	21
D	178	187	178	0	9
E	158	173	158	0	15
F	180	187	180	0	7
G	157	174	157	0	17
H	183	191	183	0	8
I	144	177	144	0	33
K	183	183	183	0	0
L	162	189	162	0	27
M	187	207	187	0	20
N	158	190	158	0	32
O	117	187	117	0	70
P	129	192	129	0	63
Q	146	202	146	0	56
S	194	194	194	0	0
T	193	193	193	0	0
U	187	187	187	0	0
V	189	191	189	0	2
W	192	192	192	0	0
X	198	198	198	0	0
Y	212	213	212	0	1
Z	208	208	208	0	0

L.6. Hasil uji coba gambar pada threshold 50%

Class	Detected	Truth	TP	FP	FN
A	187	187	187	0	0
B	176	176	176	0	0
C	158	169	158	0	11
D	179	187	179	0	8
E	170	173	170	0	3
F	187	187	187	0	0
G	167	174	167	0	7
H	184	191	184	0	7
I	154	177	154	0	23
K	183	183	183	0	0
L	174	189	174	0	15
M	198	207	198	0	9
N	180	190	180	0	10
O	118	187	118	0	69
P	137	192	137	0	55
Q	173	202	173	0	29
S	194	194	194	0	0
T	193	193	193	0	0
U	187	187	187	0	0
V	190	191	190	0	1
W	192	192	192	0	0
X	198	198	198	0	0
Y	213	213	213	0	0
Z	208	208	208	0	0

L.7. Hasil uji coba gambar pada threshold 40%

Class	Detected	Truth	TP	FP	FN
A	187	187	187	0	0
B	176	176	176	0	0
C	165	169	165	0	4
D	181	187	181	0	6
E	171	173	171	0	2
F	187	187	187	0	0
G	170	174	170	0	4
H	189	191	189	0	2
I	170	177	170	0	7
K	183	183	183	0	0
L	182	189	182	0	7
M	203	207	203	0	4
N	189	190	189	0	1
O	123	187	123	0	64
P	155	192	155	0	37
Q	191	202	191	0	11
S	194	194	194	0	0
T	193	193	193	0	0
U	187	187	187	0	0
V	191	191	191	0	0
W	192	192	192	0	0
X	198	198	198	0	0
Y	213	213	213	0	0
Z	208	208	208	0	0

L.8. Hasil uji coba gambar pada threshold 30%

Class	Detected	Truth	TP	FP	FN
A	187	187	187	0	0
B	176	176	176	0	0
C	168	169	168	0	1
D	183	187	183	0	4
E	172	173	172	0	1
F	187	187	187	0	0
G	171	174	171	0	3
H	191	191	191	0	0
I	176	177	176	0	1
K	183	183	183	0	0
L	185	189	185	0	4
M	206	207	206	0	1
N	190	190	190	0	0
O	139	187	139	0	48
P	179	192	179	0	13
Q	200	202	200	0	2
S	194	194	194	0	0
T	193	193	193	0	0
U	187	187	187	0	0
V	191	191	191	0	0
W	192	192	192	0	0
X	198	198	198	0	0
Y	213	213	213	0	0
Z	208	208	208	0	0

L.9. Hasil uji coba gambar pada threshold 20%

Class	Detected	Truth	TP	FP	FN
A	187	187	187	0	0
B	176	176	176	0	0
C	169	169	169	0	0
D	186	187	186	0	1
E	173	173	173	0	0
F	187	187	187	0	0
G	174	174	174	0	0
H	191	191	191	0	0
I	177	177	177	0	0
K	183	183	183	0	0
L	189	189	189	0	0
M	207	207	207	0	0
N	190	190	190	0	0
O	166	187	166	0	21
P	192	192	192	0	0
Q	202	202	202	0	0
S	194	194	194	0	0
T	193	193	193	0	0
U	187	187	187	0	0
V	191	191	191	0	0
W	192	192	192	0	0
X	198	198	198	0	0
Y	213	213	213	0	0
Z	208	208	208	0	0

L.10. Hasil uji coba gambar pada threshold 10%

Class	Detected	Truth	TP	FP	FN
A	187	187	187	0	0
B	176	176	176	0	0
C	169	169	169	0	0
D	187	187	187	0	0
E	173	173	173	0	0
F	187	187	187	0	0
G	174	174	174	0	0
H	191	191	191	0	0
I	177	177	177	0	0
K	183	183	183	0	0
L	189	189	189	0	0
M	207	207	207	0	0
N	190	190	190	0	0
O	186	187	186	0	1
P	192	192	192	0	0
Q	202	202	202	0	0
S	194	194	194	0	0
T	193	193	193	0	0
U	187	187	187	0	0
V	191	191	191	0	0
W	192	192	192	0	0
X	198	198	198	0	0
Y	213	213	213	0	0
Z	208	208	208	0	0

L.11. Hasil uji coba gambar pada threshold 9%

Class	Detected	Truth	TP	FP	FN
A	187	187	187	0	0
B	176	176	176	0	0
C	169	169	169	0	0
D	187	187	187	0	0
E	173	173	173	0	0
F	187	187	187	0	0
G	174	174	174	0	0
H	191	191	191	0	0
I	177	177	177	0	0
K	183	183	183	0	0
L	189	189	189	0	0
M	207	207	207	0	0
N	190	190	190	0	0
O	187	187	187	0	0
P	192	192	192	0	0
Q	202	202	202	0	0
S	194	194	194	0	0
T	193	193	193	0	0
U	187	187	187	0	0
V	191	191	191	0	0
W	192	192	192	0	0
X	198	198	198	0	0
Y	213	213	213	0	0
Z	208	208	208	0	0

L.12. Pembagian data training

Kelas	K1	K2	K3	K4	K5	Total
A	78	73	141	148	308	935
B	75	97	128	120	284	880
C	82	74	124	120	276	845
D	123	82	127	120	296	935
E	84	89	143	120	256	865
F	120	110	138	120	260	935
G	78	76	134	120	288	870
H	131	96	133	120	284	955
I	58	93	161	116	280	885
K	111	123	118	120	260	915
L	94	111	147	120	284	945
M	164	111	181	120	252	1035
N	128	92	144	116	280	950
O	124	107	117	120	280	935
P	127	140	129	120	252	960
Q	156	134	122	120	276	1010
S	133	123	124	120	276	970
T	124	99	129	120	300	965
U	125	110	121	120	272	935
V	129	122	133	124	256	955
W	116	115	129	120	288	960
X	116	159	137	116	264	990
Y	150	162	136	132	272	1065
Z	146	144	138	120	284	1040
Total	2772	2642	3234	2912	6628	22735

L.13. Pembagian data testing

Kelas	K1	K2	K3	K4	K5	Total
A	17	17	39	37	77	935
B	25	18	32	30	71	880
C	18	26	26	30	69	845
D	27	23	33	30	74	935
E	21	21	37	30	64	865
F	30	30	32	30	65	935
G	22	19	31	30	72	870
H	39	19	32	30	71	955
I	17	22	39	29	70	885
K	29	32	27	30	65	915
L	36	24	28	30	71	945
M	41	29	44	30	63	1035
N	37	28	26	29	70	950
O	31	23	33	30	70	935
P	38	35	26	30	63	960
Q	39	31	33	30	69	1010
S	37	32	26	30	69	970
T	31	26	31	30	75	965
U	40	15	34	30	68	935
V	31	33	32	31	64	955
W	34	30	26	30	72	960
X	29	51	23	29	66	990
Y	35	43	34	33	68	1065
Z	24	41	42	30	71	1040
Total	728	668	766	728	1657	22735

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Steve Daniels, lahir di Bekasi pada tanggal 29 Oktober 1998. Penulis menempuh pendidikan mulai dari TK Bonafide (2002 – 2004), SD Mutiara 17 Agusutus (2004 – 2007), SD Strada Budi Luhur I (2007 – 2010), SMP Strada Budi Luhur (2010 – 2013), SMAK Penabur Summarecon Bekasi (2013 – 2016), dan sekarang sedang menjalani pendidikan sarjana di Departemen Teknik Informatika, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember Surabaya.

Penulis aktif dalam berbagai kepanitiaan. Di antaranya adalah menjadi staf dana dan usaha Schematics 2017, dan staf ahli dana dan usaha Schematics 2018. Komunikasi dengan penulis dapat melalui telepon: +6282213065755 dan *email*: **dan.steve2910@gmail.com**.