TUGAS AKHIR - IF184802

PENGENALAN EKSPRESI WAJAH PADA DATA VIDEO MENGGUNAKAN PENDEKATAN BERBASIS CONVOLUTIONAL NEURAL NETWORK

AYAS FAIKAR NAFIS
NRP 05111640000138

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

Dosen Pembimbing II
Anny Yuniarti, S.Kom., M.Comp.Sc.

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

TUGAS AKHIR – IF184802

PENGENALAN EKSPRESI WAJAH PADA DATA VIDEO MENGGUNAKAN PENDEKATAN BERBASIS CONVOLUTIONAL NEURAL NETWORK

AYAS FAIKAR NAFIS
NRP 05111640000138

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

Dosen Pembimbing II
Anny Yuniarti, S.Kom., M.Comp.Sc.

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

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESIS – IF184802

FACIAL EXPRESSIONS RECOGNITION ON VIDEO DATA USING A CONVOLUTIONAL NEURAL NETWORK BASED APPROACH

AYAS FAIKAR NAFIS
NRP 05111640000138

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

Second Advisor
Anny Yuniarti, S.Kom., M.Comp.Sc.

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

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

PENGENALAN EKSPRESI WAJAH PADA DATA VIDEO MENGGUNAKAN PENDEKATAN BERBASIS CONVOLUTIONAL NEURAL NETWORK

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:
AYAS FAIKAR NAFIS
NRP: 05111640000138

Disetujui oleh Pembimbing Tugas Akhir:

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

2. Anny Yuniarti, S.Kom., M.Comp.Sc. .....
   (NIP. 19810622 200501 2 002) (Pembimbing 2)

SURABAYA
Juni, 2020

(Halaman ini sengaja dikosongkan)

PENGENALAN EKSPRESI WAJAH PADA DATA VIDEO MENGGUNAKAN PENDEKATAN BERBASIS CONVOLUTIONAL NEURAL NETWORK

Nama : Ayas Faikar Nafis
NRP : 05111640000138
Departemen : Teknik Informatika, FTEIC-ITS
Pembimbing I : Dini Adni Navastara, S.Kom., M.Sc.
Pembimbing II : Anny Yuniarti, S.Kom., M.Comp.Sc.

ABSTRAK

Ekspresi wajah pada manusia merupakan salah satu bentuk komunikasi secara non-verbal untuk menyampaikan keadaan emosi dalam diri manusia, sehingga berperan penting dalam interaksi sosial antar manusia. Pengenalan ekspresi wajah memainkan peran penting dalam interaksi manusia-komputer, pengawasan perilaku manusia, teknik pendidikan, psikologis, hingga robot sociable. Pengenalan ekspresi wajah dengan komputer pada saat ini dapat dilakukan dengan metode deteksi objek yang berbasis deep learning untuk mendeteksi area wajah dan memprediksi ekspresi wajah, namun metode deteksi objek dengan dua tahap yaitu tahap deteksi dan tahap klasifikasi memiliki waktu deteksi dan klasifikasi yang lama. Salah satu solusi untuk mengatasi masalah ini adalah dengan menggunakan metode deteksi objek yang hanya melakukan satu tahap untuk deteksi dan klasifikasi, salah satunya adalah metode You Only Look Once (YOLO).

Dalam penelitian ini, dilakukan pengembangan pengenalan ekspresi wajah manusia dengan menggunakan metode You Only Look Once (YOLO) yang berbasis Convolutional Neural Network (CNN) dan mempunyai waktu deteksi yang cepat. Terdapat 7 kelas ekspresi wajah yang dapat dikenali yaitu marah, jijik, takut, senang, sedih, terkejut dan netral. Dataset dilakukan preprocessing, lalu dilakukan training dengan YOLO, setelah itu model diuji coba dengan data testing untuk evaluasi.

Dataset yang digunakan pada penelitian ini adalah dataset ekspresi wajah berbasis video seperti The Extended Cohn-Kanade AU-Coded Facial Expression Database (CK+), The Indonesian Mixed Emotion Dataset (IMED), dan data video yang dikumpulkan dari 8 subjek mahasiswa Teknik Informatika ITS dengan berbagai macam posisi wajah. Berdasarkan hasil pengujian, akurasi terbaik pada pengenalan ekspresi wajah sebesar 68% dan waktu deteksi sebesar 63,56 detik dengan parameter channel 3, learning rate 0,01, momentum 0,90, decay 0,0005, dan default anchor box.

Kata kunci: ***CNN, Pengenalan Ekspresi Wajah, YOLO.***

FACIAL EXPRESSIONS RECOGNITION ON VIDEO DATA USING A CONVOLUTIONAL NEURAL NETWORK BASED APPROACH

Student's Name : Ayas Faikar Nafis
Student's ID : 05111640000138
Department : Informatics, Faculty of ELECTICS-ITS
First Advisor : Dini Adni Navastara, S.Kom., M.Sc.
Second Advisor : Anny Yuniarti, S.Kom., M.Comp.Sc.

ABSTRACT

Facial expression in humans is a form of non-verbal communication to convey the emotional state in humans, so it plays an important role in social interaction between humans. The introduction of facial expressions plays an important role in human-computer interaction, monitoring human behavior, educational techniques, psychological, to sociable robots. Facial expressions recognition with a computer at this time can be done with object detection methods based on deep learning to detect facial areas and predict facial expressions, but the object detection method with two stages, namely the detection phase and the classification stage have a long detection and classification time. One solution to overcome this problem is to use an object detection method that only takes one step for detection and classification, one of which is the You Only Look Once (YOLO) method.

In this study, the development of human facial expression recognition using the You Only Look Once (YOLO) method based on Convolutional Neural Network (CNN) and has a fast detection time. There are 7 classes of facial expressions that can be recognized, namely anger, disgust, fear, happy, sadness, surprise and neutral. The dataset is preprocessed, then training with YOLO, after which the model is tested with data testing for evaluation.

The dataset used in this study is a video-based facial expression dataset such as The Extended Cohn-Kanade AU Coded Facial Expression Database (CK +), The Indonesian Mixed

Emotion Dataset (IMED), and video data collected from 8 ITS Informatics Engineering students various face positions. Based on the test results, the best accuracy on facial expression recognition is 68% and the detection time is 63.56 seconds with channel 3 parameters, learning rate 0.01, momentum 0.90, decay 0.0005, and default anchor box.
Keywords: ***CNN, Face Expression Recognition, YOLO.***

KATA PENGANTAR

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

"**PENGENALAN EKSPRESI WAJAH PADA DATA VIDEO MENGGUNAKAN PENDEKATAN BERBASIS CONVOLUTIONAL NEURAL NETWORK**"

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. Abi, Ummi, Alma, Ara, dan keluarga besar 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 Anny Yuniarti, S.Kom., M.Comp.Sc. 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, 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. Sukarelawan yang telah mengirimkan video ekspresi wajah dengan berbagai posisi. Video tersebut sangat membantu dalam pengerjaan Tugas Akhir ini.
5. Akbar, Abyan, dan Zahrah yang menemani penulis di grup kelompok biomedik serta membagikan pengalaman dan motivasi selama pengerjaan Tugas Akhir ini.

6. Rifqi Mukti, Steve, Iyus, Puguh, dan Ariiq yang telah menemani dan menghibur penulis selama kuliah di Teknik Informatika ITS.
7. Teman-teman BrainWash Syndicate yang telah banyak membantu dan menemani penulis selama kuliah di Teknik Informatika ITS.
8. Admin-admin Laboratorium *Mobile Innovation Studio* (MIS) Muhajir, Fasma, Zevi, Fino, Anggar, Yuda, Andre, Naja, Shania, Hisam, Kana, Syubban, dan Ela yang menemani penulis selama menjadi admin MIS.
9. Admin-admin Laboratorium Komputasi Cerdas & Visi (KCV) yang memberikan kesempatan penulis untuk fokus mengerjakan Tugas Akhir ini dan menyediakan tempat di laboratorium tersebut.
10. Seluruh mahasiswa *user* TA Teknik Informatika ITS angkatan 2016 yang telah menjadi teman penulis selama pengerjaan Tugas Akhir ini.
11. Seluruh mahasiswa Teknik Informatika ITS angkatan 2016 yang telah menjadi teman penulis selama menjalani masa kuliah di Teknik Informatika ITS.
12. 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, Juni 2020

DAFTAR ISI

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Manusia merupakan makhluk sosial yang dapat berkomunikasi dengan manusia lainnya dengan cara verbal maupun dengan cara non-verbal. Ekspresi wajah merupakan salah satu bentuk komunikasi manusia berupa non-verbal yang diekspresikan melalui otot-otot pada wajah untuk menyampaikan keadaan emosi dalam diri manusia, sehingga berperan penting dalam interaksi sosial antar manusia [1].

Pengenalan ekspresi wajah telah lama menjadi bidang penelitian yang menarik. Penelitian mengenai analisis ekspresi wajah yang dilakukan oleh psikolog menemukan 6 ekspresi dasar manusia yaitu marah, jijik, takut, senang, sedih, dan terkejut. Analisis ekspresi wajah merupakan bidang penelitian utama psikologi, dan banyak karya dan literatur diterbitkan di bidang ini [2].

Namun, seiring dengan perkembangan jaman, penelitian mengenai analisis ekspresi wajah merambah kepada pengenalan ekspresi wajah secara otomatis oleh komputer. Pengenalan ekspresi wajah memainkan peran penting dalam interaksi manusia-komputer, pengawasan perilaku manusia, teknik pendidikan, psikologis, dan *robot sociable* [3]. Komputer dapat belajar layaknya manusia dan medeteksi pola-pola ekspresi manusia dengan metode *deep learning* yang merupakan bagian dari *machine learning* [4].

Pengenalan ekspresi wajah dengan komputer pada saat ini dapat dilakukan dengan metode deteksi objek yang berbasis *deep learning* untuk mendeteksi area wajah dan memprediksi ekspresi wajah, namun metode deteksi objek dengan dua tahap yaitu tahap deteksi dan tahap klasifikasi mempunyai kelemahan yaitu waktu deteksi dan klasifikasi yang lama [4].

Salah satu solusi untuk mengatasi masalah ini adalah dengan menggunakan metode deteksi objek yang hanya melakukan satu

tahap untuk deteksi dan klasifikasi. Salah satu metode *deep learning* yang hanya menggunakan satu tahap untuk deteksi dan klasifikasi objek yang akhir-akhir ini banyak digunakan adalah *You Only Look Once* (YOLO). YOLO merupakan metode deteksi objek berbasis *Convolutional Neural Network* (CNN) yang dapat menghasilkan deteksi objek yang cepat dan efektif dan menjadi salah satu keunggulan YOLO dari metode deteksi objek lainnya [4].

Metode YOLO pernah digunakan pada penelitian untuk membuat *robot sociable* yang mempunyai kemampuan untuk mengenali ekspresi wajah pengguna robot tersebut sehingga robot dapat mengetahui emosi pengguna dan dapat melakukan respon yang tepat kepada pengguna robot [3]. Tetapi metode YOLO yang digunakan merupakan YOLO yang telah dilakukan *training* hanya untuk mendeteksi wajah yang bernama YOLO Faced, sedangkan untuk pengenalan ekspresi wajah menggunakan Ensemble CNN. Sehingga penelitian tersebut masih menggunakan dua tahap untuk mengenali ekspresi wajah manusia.

Di dalam tugas akhir ini, metode YOLO digunakan untuk mendeteksi wajah dan mengenali ekspresi wajah berbasis data video. Terdapat 7 kelas ekspresi wajah yang dapat dikenali yaitu marah, jijik, takut, senang, sedih, terkejut dan netral. Dataset yang digunakan adalah dataset ekspresi wajah berbasis video seperti *The Extended Cohn-Kanade AU-Coded Facial Expression Database* (CK+) [5], *The Indonesian Mixed Emotion Dataset* (IMED) [6], dan data video dari 8 subjek mahasiswa Teknik Informatika ITS dengan berbagai macam posisi wajah, sehingga YOLO dapat mengenali ekspresi wajah dengan posisi wajah yang berbeda. Pertama, dataset dilakukan *preprocesing*, dan dilakukan *training* dengan menggunakan YOLO. Pengujian dilakukan dengan menggunakan data *testing* yang akan menghasilkan hasil berupa *bounding box*, prediksi kelas ekspresi wajah dan *confidence score*. Evaluasi dilakukan dengan membandingkan kelas hasil prediksi dengan kelas *ground truth* menggunakan pengukuran akurasi, *precision*, *recall* dan *F1-Score*.

1.2 Rumusan Masalah

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

1. Bagaimana proses *preprocessing* pada dataset berbasis video?
2. Bagaimana cara mengimplementasikan YOLO untuk pengenalan ekspresi wajah?
3. Bagaimana cara mengukur kinerja pengenalan ekspresi wajah menggunakan metode YOLO?

1.3 Batasan Permasalahan

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

1. Dataset yang dipakai berasal dari dataset *The Extended Cohn-Kanade AU-Coded Facial Expression Database* (CK+), *The Indonesian Mixed Emotion Dataset* (IMED) dan data video dari 8 subjek mahasiswa Teknik Informatik ITS dengan berbagai macam posisi wajah.
2. Terdapat 7 kelas ekspresi yang dapat dikenali yaitu senang, sedih, marah, jijik, takut, terkejut dan netral.
3. Menggunakan bahasa Python 3.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah mengaplikasikan metode YOLO untuk pengenalan ekspresi wajah manusia pada data berbasis video.

1.5 Manfaat

Tugas akhir ini diharapkan dapat meningkatkan kualitas interaksi manusia dan komputer dan dapat diterapkan pada sistem yang membutuhkan pengenalan ekspresi wajah seperti bidang keamanan, kesehatan, dan 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 subbab 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*, *darknet*, dan TensorFlow.

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 dan TensorFlow sebagai *framework*, serta *library* pendukung lainya.

1.6.4 Pengujian dan Evaluasi

Tahap pengujian dan evaluasi dilakukan menggunakan dataset *The Extended Cohn-Kanade AU-Coded Facial Expression Database* (CK+), *The Indonesian Mixed Emotion Dataset* (IMED), dan data video untuk mengetahui hasil dan performa metode yang telah dibangun. Evaluasi dan perbaikan akan dilakukan hingga

perangkat lunak yang diuji menghasilkan hasil performa yang baik. Beberapa pengujian yang dilakukan antara lain perubahan-perubahan pada parameter *channel*, *learning rate*, *momentum*, *decay*, dan *anchor box* pada arsitektur YOLO yang dibuat. Evaluasi akurasi akan dilakukan dengan menggunakan akurasi, presisi, *recall*, 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 *Neural Network*, YOLO, dan *library* yang digunakan.

Bab III Perancangan Sistem

Bab ini berisi pembahasan mengenai perancangan dari metode YOLO yang digunakan untuk pengenalan ekspresi wajah 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

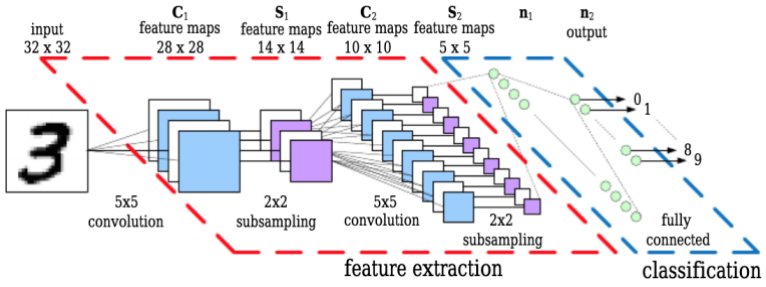
Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses dan tertulis saat pengerjaan tugas akhir, dan saran untuk pengembangan solusi ke depannya.

BAB II
TINJAUAN PUSTAKA

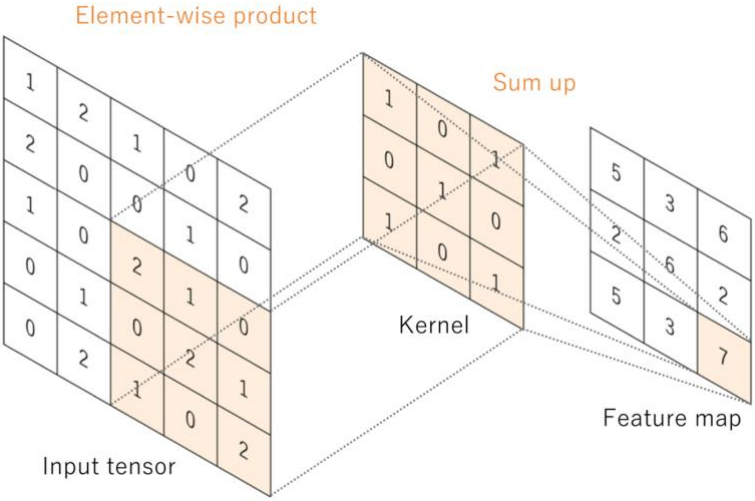
Bab ini membahas mengenai teori-teori dasar yang digunakan dalam tugas akhir. Teori-teori tersebut adalah *Convolution Neural Network (CNN),* YOLO, *Intersection Over Union (IOU), Non-Maximum Suppression* dan beberapa teori lain yang mendukung pembuatan tugas akhir. Penjelasan ini bertujuan untuk memberikan gambaran umum dan diharapkan dapat mendukung sistem yang dibangun.

2.1 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) adalah salah satu jenis model *deep learning* untuk memproses data yang memiliki pola kisi, seperti gambar dan dirancang untuk memperlajari fitur secara otomatis dan adaptif. CNN biasanya terdiri dari tiga jenis *layer* antara lain *convolutional layer*, *pooling layer*, dan *fully connected layer*. Pada *convolutional layer* dan *pooling layer* melakukan proses ekstraksi fitur, sedangkan *fully connected layer*, memetakan fitur yang diekstraksi ke dalam hasil akhir, seperti klasifikasi [7]. Gambar 2.1 merupakan contoh arsitektur CNN untuk pengenalan digit angka yang ditulis dengan tangan. *Subsampling Layer* pada Gambar 2.1 mempunyai arti yang sama dengan *Pooling Layer*.



Gambar 2.1 Contoh Arsitektur CNN [22]



Gambar 2.2 Operasi Konvolusi pada Convolutional Layer [7]

2.1.1 Convolutional Layer

Convolutional Layer terdiri dari tumpukan hasil proses konvolusi pada gambar input. Konvolusi merupakan operasi matematika yang dilakukan pada gambar input secara berulang untuk setiap posisi piksel pada gambar. Proses konvolusi ini dilakukan pada gambar input dengan *grid* yang lebih kecil yang disebut *kernel* [7]. Operasi konvolusi pada gambar input berukuran 5×5 dengan *kernel* berukuran 3×3 dapat dilihat pada Gambar 2.2.

2.1.2 Pooling Layer

Pooling Layer atau *Subsampling Layer* digunakan untuk mereduksi ukuran dari data. Metode *Pooling* dalam yang biasa digunakan dalam CNN adalah *Max Pooling* dan *Average Pooling*. *Max Pooling* membagi *output* dari *Convolution Layer* menjadi beberapa matriks kecil lalu mengambil nilai maksimal dari tiap

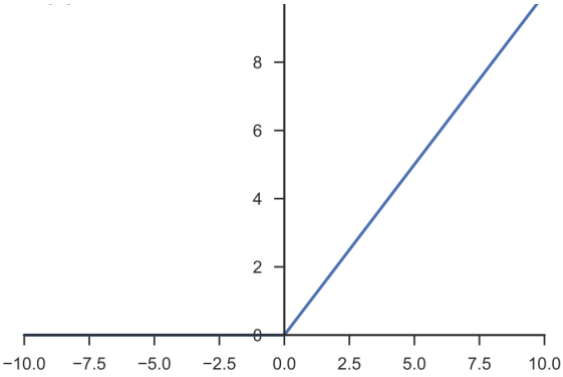
matriks untuk menyusun matriks citra yang telah direduksi, sedangkan *Average Pooling* akan memilih nilai rata-ratanya [8].

2.1.3 Fully Connected Layer

Feature Map keluaran dari konvolusi akhir atau *Pooling Layer* biasanya akan diubah menjadi *array* satu dimensi (1D) angka atau vektor, dan terhubung ke satu atau lebih *Fully Connected Layer*, di mana setiap input terhubung ke setiap *output* dengan bobot yang bisa dipelajari. Setelah fitur diekstraksi oleh *Convolutional Layer* dan direduksi oleh *Pooling Layer*, fitur dipetakan oleh subset dari *Fully Connencted Layer* ke hasil akhir *neural network*, seperti probabilitas untuk setiap kelas dalam tugas klasifikasi. *Fully Connencted Layer* akhir biasanya memiliki jumlah *node* keluaran yang sama dengan jumlah kelas. Setiap lapisan yang terhubung sepenuhnya diikuti oleh fungsi *non-linear*, seperti ReLU [7].

2.1.4 Fungsi Aktivasi ReLU

Fungsi aktivasi adalah unit yang menentukan informasi mana yang harus dikirim ke *neuron* berikutnya. Setiap neuron dalam *neural network* menerima nilai *output neuron* dari *layer* sebelumnya sebagai input dan meneruskan nilai yang diproses ke



Gambar 2.3 Fungsi Aktivasi ReLU [7]

layer berikutnya. Dalam *multilayer neural network*, terdapat fungsi antara dua lapisan. Fungsi ini disebut fungsi aktivasi. Salah satu fungsi aktivasi adalah ReLU (*Rectified Linear Unit*) yang mempunyai nilai fitur masukkan x. Jika nilai x kurang dari nol, maka keluaran dari ReLU dijadikan nol. Sedangkan jika x lebih dari nol, maka keluaran dari ReLU adalah x. Gambar dai fungsi aktivasi ReLU dapat dilihat pada Gambar 2.3 dan persamaan ReLU dapat dilihat pada Persamaan (2.1) [8].

$$f(x) = \max(0, x) \tag{2.1}$$

2.1.5 Fungsi Softmax

Fungsi aktivasi yang diterapkan ke *Fully Connected Layer* biasanya berbeda dari yang lain. Fungsi aktivasi yang sesuai perlu dipilih sesuai dengan setiap tugas. Fungsi aktivasi yang diterapkan pada tugas klasifikasi multi-kelas adalah fungsi *softmax* yang menormalkan nilai nyata *output* dari *Fully Connected Layer* ke probabilitas kelas target, di mana setiap nilai berkisar antara 0 dan 1 dan semua nilai berjumlah 1. Fungsi ini akan menghasilkan nilai probabilitas. Label atau kelas dari data masukan akan ditentukan berdasarkan kelas dengan nilai probabilitas tertinggi. Fungsi softmax S dengan masukkan x pada index ke-i dan index j dapat dirumuskan pada Persamaan (2.2) [7].

$$S(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{2.2}$$

2.1.6 Cross Entropy Loss

Loss function merupakan fungsi yang menggambarkan kerugian yang dihasilkan oleh model CNN. Model CNN dikatakan baik ketika menghasilkan *loss* yang paling rendah. *Loss function* yang sering digunakan pada permasalahan klasifikasi banyak kelas adalah *Cross Entropy Loss*.

Cross Entropy Loss digunakan untuk mengevaluasi perbedaan antara distribusi probabilitas yang diperoleh dari

pelatihan saat ini dan distribusi aktual. Fungsi ini membandingkan probabilitas yang diprediksi dengan nilai *output* aktual (0 atau 1) di setiap kelas dan menghitung nilai penalti berdasarkan jarak dari mereka. Fungsi ini bersifat logaritmik, sehingga fungsinya memberikan skor yang lebih kecil untuk perbedaan yang lebih kecil dan skor yang lebih besar untuk perbedaan yang lebih besar. *Cross Entropy Loss* juga disebut kerugian *softmax*, yang mengindikasikan selalu digunakan di CNN dengan lapisan *softmax*. Fungsi *Cross Entropy Loss D* akan menghitung *error* antara nilai prediksi *S* dengan nilai sebenarnya *T*, seperti pada Persamaan (2.3). Selanjutnya, nilai *error* akhir *J* dengan *weight W* dan bias *b* diambil dari rata-rata hasil *cross entropy*, seperti pada Persamaan (2.4) [8].

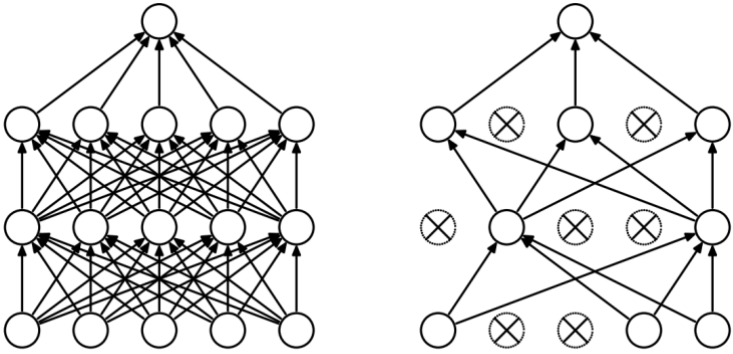
$$D(S_i, T_i) = -\sum_j T_{ij} \log S_{ij} \tag{2.3}$$

$$J(W, b) = \frac{1}{n}\sum_i D(S_i, T_i) \tag{2.4}$$

2.1.7 Dropout

Dropout adalah teknik yang dapat mencegah *overfitting*, mempercepat *training* dan menyediakan cara untuk menggabungkan secara eksponensial banyak arsitektur *neural network* yang berbeda secara efisien. Istilah *Dropout* mengacu pada menghapus sementara *neuron* dari jaringan, bersama dengan semua koneksi masuk dan keluarnya. pada *hidden layer* maupun *visible layer* dalam *neural network*. *Neuron* yang akan dihilangkan akan dipilih adalah acak.

Ilustrasi neural network sebelum dan sesudah menggunakan *Dropout* dapat dilihat pada Gambar 2.4. Pada bagian (a) Semua neuron dipakai pada *neural network* yang tidak memakai *Dropout*, dan pada bagian (b) sebagian dari *neuron* dalam *neural network* tidak digunakan setelah menggunakan *Dropout* [9].



Gambar 2.4 Ilustrasi Neural Network Sebelum (a) dan Sesudah (b) Menggunakan Dropout [9]

2.1.8 Batch Normalization

Batch Normalization adalah jenis *layer* suplemental yang secara adaptif menormalkan nilai input dari *layer*, mengurangi risiko *overfitting*, serta meningkatkan *gradient flow* pada *neural network*, memungkinkan tingkat pembelajaran yang lebih tinggi, dan mengurangi ketergantungan pada inisialisasi [7].

2.2 YOLO (You Only Look Once)

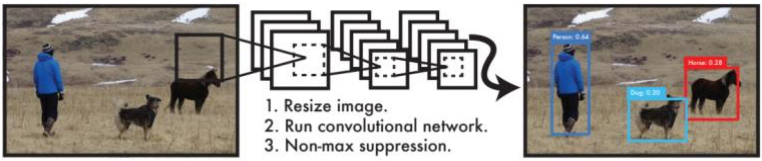
YOLO (*You Only Look Once*) merupakan arsitektur *one-shot* pertama yang berhasil diusulkan untuk deteksi dan klasifikasi objek secara bersamaan. YOLOv2 atau YOLO9000, adalah versi modifikasi dari jaringan YOLO yang asli. YOLOv2 dapat berjalan dalam berbagai ukuran dan menawarkan *trade-off* antara kecepatan dan akurasi dikarenankan metode pelatihan multi-skala. Selanjutnya versi terakhir YOLOv3 dirilis pada tahun 2018. YOLOv3 memiliki arsitektur baru untuk ekstraksi fitur bernama Darknet-53. Tabel perbandingan antara YOLOv1, YOLOv2, dan YOLOv3 dapat dilihat pada Tabel 2.1.

Tabel 2.1 Perbandingan YOLOv1, YOLOv2, dan YOLOv3

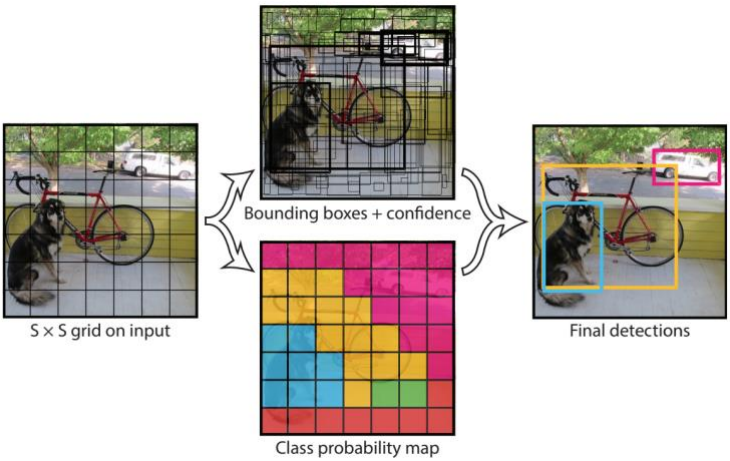
Keterangan	YOLOv1	YOLOv2	YOLOv3
Feature Extractor	Darknet	Darknet-19	Darknet-53
Batch Normalization	Tidak ada	Ada	Ada
Jumlah Anchor Box	Tidak ada	5	9
Loss Function	Sum Squared Error	Sum Squared Error	Cross Entropy Loss
Multi-Scale Training	Tidak	Ya	Ya

2.2.1 YOLOv1

Kebanyakan metode *object detection* selain YOLO seperti R-CNN, pertama akan menghasilkan *bounding box* yang memperkirakan ada atau tidaknya objek, lalu dilakukan klasifikasi dengan CNN untuk menentukan objek apa yang terdapat pada *bounding box* tersebut. YOLO menggunakan pendekatan yang berbeda, YOLO hanya melakukan sekali CNN untuk memprediksi objek-objek yang terdapat pada gambar seperti yang dapat dilihat pada Gambar 2.5. Hal ini bisa terjadi karena YOLO membagi gambar menjadi *grid* sebesar $S \times S$. Setiap sel bertanggung jawab untuk memprediksi sebanyak B *bounding box* dan tingkat



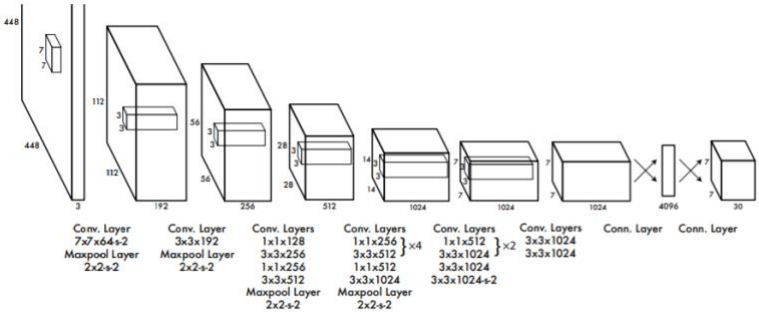
Gambar 2.5 Sistem Deteksi YOLO [4]



Gambar 2.6 Model YOLO [4]

confidence dari setiap sel serta C probabilitas kelas seperti yang dapat dilihat pada Gambar 2.6.

Sebagai contoh, untuk mengevaluasi YOLO pada dataset PASCAL VOC, YOLO menggunakan 7×7 *grid* $(S \times S)$, 2 *bounding box* (B) dan 20 kelas (C). Setiap *bounding box* mempunyai 5 elemen yang diprediksi antara lain *x*, *y*, *w*, *h* dan *box confidence score*, di mana *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 seberapa yakin kemungkinan *bounding box* tersebut berisi objek dan seberapa akurat *bounding box* prediksi dengan *ground truth* yang dapat diukur dengan *Intersection Over Union* (IOU). Jika tidak ada objek ditemukan pada sel maka *box confidence score* akan bernilai nol, sebaliknya jika ditemukan objek pada sel maka box confidence score akan sama dengan *Intersection Over Union* (IOU) antara *bounding box* prediksi dan *ground truth.* Setiap sel mempunyai 20 probabilitas



Gambar 2.7 Arsitektur YOLOv1 [4]

kelas kondisional yang merupakan probabilitas kelas dari objek yang terdeteksi.

Prediksi YOLO memiliki bentuk tensor $S \times S \times (B \times 5 + C)$ sehingga dengan $S = 7$, $B = 2$, dan $C = 20$, prediksi final memiliki tensor $7 \times 7 \times (2 \times 5 + 20)$ atau $7 \times 7 \times 30$ tensor. YOLO mempunyai 24 layer CNN untuk mengekstraksi fitur gambar dan 2 *connected layer* untuk melakukan regresi linier untuk membuat tensor $7 \times 7 \times 2$ prediksi *bounding box*. Akhirnya, *bounding box* yang ditampilkan hanya yang mempunyai *confidence score* melebihi dari *threshold* yang ditentukan [4]. Arsitektur YOLOv1 dapat diliihat pada Gambar 2.7.

2.2.2 YOLOv2

YOLO versi awal atau YOLOv1 mempunyai bebrapa kekurangan yaitu tidak bisa mendeteksi objek yang kecil, hanya dapat mendeteksi 49 objek saja, dan mempunyai *localization error* yang tinggi. YOLOv2 merupakan pengembangan dari YOLOv1 yang berfokus pada peningkatan daya ingat dan lokalisasi kesalahan dari versi YOLO sebelumnya, YOLOv2 adalah metode deteksi yang lebih akurat dan lebih cepat. Itu memiliki banyak perbaikan dibandingkan dengan YOLO, termasuk *batch normalization, high resolution classifier, Anchor boxes, dimension clusters*, *direct location prediction*, *multi-scale training*, *fine grained features*, dan *high resolution detector*. Selain itu, *pass-*

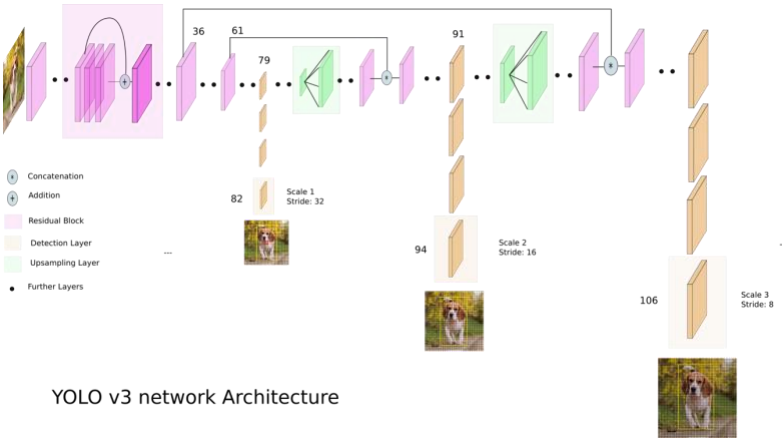
through layer ditambahkan untuk meningkatkan kemampuan untuk dapat mendeteksi objek yang lebih kecil. Selain itu, YOLOv2 menggunakan arsitektur *neural network* lain yang disebut Darknet-19 sedangkan YOLOv1 menggunakan arsitektur Darknet yang didasarkan pada arsitektur GoogleNet [10].

2.2.3 YOLOv3

YOLOv3 merupakan pengembangan dari YOLOv2. Berbeda dari YOLOv2, arsitektur YOLOv3 didasarkan pada Darknet-53 yang menggunakan 53 *convolutional layer* untuk ekstraksi fitur seperti pada Gambar 2.8. YOLOv3 banyak perbaikan dibandingkan dengan YOLOv2. Pertama, YOLOv3 memprediksi *bounding box* pada 3 skala yang berbeda dan skala diberikan dengan mengambil sampel dimensi gambar input masing-masing sebesar 32, 16, dan 8. Skala ini digunakan untuk mendeteksi objek dengan ukuran besar, sedang, dan kecil.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Gambar 2.8 Arsitektur Darknet-53 [11]



Gambar 2.9 Arsitektur YOLOv3 [23]

Akibatnya, YOLOv3 memprediksi 10 kali jumlah *bounding box* yang diprediksi oleh YOLOv2. Kedua, *loss function* diubah dari *squared error* menjadi *cross entropy loss*. Dengan kata lain, *confidence* objek untuk setiap *bounding box* diprediksi menggunakan *logistic regression*. Akhirnya, YOLOv3 melakukan klasifikasi *multi-label* untuk mendeteksi objek. Oleh karena itu, fungsi *softmax* diganti dengan pengklasifikasi logistik independen.

YOLOv3 mempunyai 75 *Convolutional Layer*, 23 *Shortcut Layer*, 3 *YOLO Layer*, 4 *Route Layer*, dan 2 *Upsample Layer* yang membuat YOLOv3 memiliki 107 *Layer* seperti pada Gambar 2.9. *Shortcut layer* merupakan layer yang menambahkan *feature map* layer sebelumnya dengan layer ke-n sebelumnya. YOLO layer merupakan layer yang mengeluarkan prediksi objek dengan *bounding box*. *Route layer* merupakan layer yang mengambil *feature map* dari layer ke-n sebelumnya. Sedangkan *Upsample layer* merupakan layer yang dapat memperbesar *feature map* pada layer sebelumnya.

Cara kerja YOLOv3 hampir sama dengan YOLO sebelumnya, yaitu membagi gambar menjadi $S \times S$ *grid*, lalu setiap *grid* akan mendeteksi objek disekitarnya. Tetapi YOLOv3

menggunakan *Anchor Box* atau *Prior Box* yang merupakan *bounding box* yang sudah didefinisikan di awal dengan berbagai macam ukuran. YOLOv3 menggunakan 3 *Anchor Box* untuk setiap skala untuk mendeteksi objek pada setiap sel, sehingga YOLOv3 mempunyai total 9 *Anchor Box* [11].

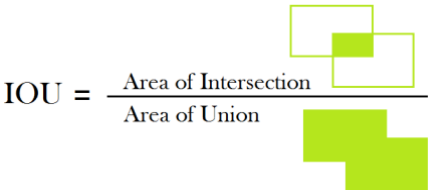
2.3 Intersection Over Union (IOU)

Intersection Over Union (IOU) merupakan teknik perhitungan yang dapat menentukan seberapa baik letak dan ukuran *bounding box* hasil prediksi terhadap *bounding box ground truth*. Semakin besar IOU yang dihasilkan maka semakin baik pula *bounding box* yang diprediksi. Rumus IOU dengan area *bounding box prediksi A* dan area *bounding box ground truth B* dapat dilihat pada Persamaan (2.5) dan Gambar 2.10 [12].

$$IOU = \frac{|A \cap B|}{|A \cup B|} \tag{2.5}$$

2.4 Non-Maximum Suppression (NMS)

YOLO menggunakan *Non-Maximum Suppression* (NMS) untuk menentukan *bounding box* terbaik. Langkah pertama dalam NMS adalah membuang semua *bounding box* yang telah diprediksi yang mempunyai probabilitas deteksi atau *confidence* objek yang kurang dari *threshold* yang telah ditentukan sebelumnya, sebagai contoh jika *threshold* diatur menjadi 0,6, maka *bounding box* yang



Gambar 2.10 Intersection Over Union [24]

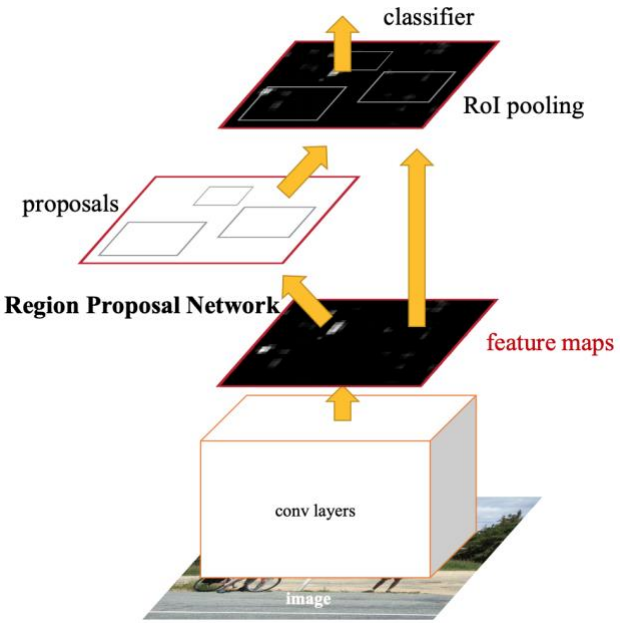
Gambar 2.11 Non-Maximum Suppression [25]

mempunyai probabilitas deteksi objek kurang dari 0,6 akan dibuang.

Setelah membuang *bounding box* yang mempunyai probabilitas deteksi kurang dari *threshold*, *bounding box* yang tersisa mungkin saling bertumpuk atau *overlapping* satu sama lain dengan deteksi kelas yang sama, oleh karena itu, *bounding box* yang saling bertumpuk harus dibuang dan menyisakan satu *bounding box* terbaik menggunakan *Intersection Over Union* (IOU), semakin besar IOU maka semakin besar pula kemungkinan *bounding box* tersebut menjadi *bounding box* terbaik. Hasil dari *Non-Maximum Suppression* dapat dilihat pada Gambar 2.11 [4].

2.5 Faster R-CNN

Faster R-CNN merupakan metode deteksi objek dan merupakan pengembangan dari R-CNN dan Fast R-CNN. R-CNN pada dasarnya mengusulkan *region* yang dianggap sebagai objek dengan algoritma pencarian selektif, lalu dilakukan klasifikasi setiap *region* yang diusulkan dengan CNN. Salah satu masalah dari R-CNN adalah waktu deteksi yang lama. Fast R-CNN merupakan pengembangan dari R-CNN yang bertujuan untuk memangkas waktu deteksi dari R-CNN dengan melakukan CNN pada citra yang akan menghasilkan *feature map*. Setelah itu, *region* diusulkan dari *feature map* dengan algoritma pencarian selektif dan diklasifikasi pada *Fully Connected Layer*. Tetapi Fast R-CNN



Gambar 2.12 Faster R-CNN [13]

masih memiliki waktu deteksi yang lama walaupun lebih cepat dibandingkan dengan R-CNN.

Faster R-CNN diusulkan untuk mengurangi waktu deteksi pada Fast R-CNN. Cara kerja Faster R-CNN hampir sama dengan Fast R-CNN. Pertama, CNN dilakukan pada citra yang akan menghasilkan *feature map*. Setelah itu, *region* diusulkan dari *feature map* dengan dengan menggunakan *Region Proposal Network* (RPN) yang menggantikan algoritma pencarian selektif untuk mengusulkan *region*. Usulan *region* yang diprediksi kemudian dilakukan *reshape* menggunakan ROI Pooling Layer yang kemudian digunakan untuk mengklasifikasikan citra dalam *region* yang diusulkan dan memprediksi nilai *offset* untuk *bounding box*. Proses deteksi dengan menggunakan Faster R-CNN dapar dilihat pada Gambar 2.12 [13].

2.6 Akurasi, Precision, Recall, dan F1-Score

Confusion Matrix adalah pengukuran kinerja *machine learning* untuk masalah klasifikasi yang dapat digunakan pada dua atau lebih kelas [14]. Tabel 2.2 merupakan tabel *confusion matrix* dengan 4 kombinasi nilai prediksi dan aktual yang berbeda.

TP (*True Positive*) adalah ketika kelas yang diprediksi positif dan kenyataannya juga positif. Contohnya, orang yang diprediksi sakit dan kenyataannya benar sakit. *TN* (*True Negative*) adalah ketika kelas yang diprediksi negatif dan kenyataannya juga negatif. Contohnya, orang yang diprediksi tidak sakit dan pada kenyataanya tidak sakit. *FP* (*False Positive*) adalah ketika kelas yang diprediksi positif namun kenyataannya negatif. Contohnya, orang yang diprediksi sakit namun sesungguhnya tidak sakit. *FN* (*False Negative*) adalah ketika kelas yang diprediksi negatif namun kenyataannya positif. Contohnya, orang yang diprediksi tidak sakit namun sesungguhnya sakit.

Nilai-nilai pada tabel *confusion matrix* dapat digunakan untuk mengukur tingkat validasi data. Beberapa jenis teknik validasi yang umum digunakan antara lain *accuracy*, *precision*, *recall*, dan *F1-Score*.

Tabel 2.2 Confusion Matrix

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

Akurasi atau *Accuracy* merupakan salah satu ukuran yang paling umum digunakan untuk kinerja klasifikasi, dan didefinisikan sebagai rasio antara sampel yang diklasifikasikan dengan benar dengan jumlah total sampel dan dapat dirumuskan pada Persamaan (2.6).

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.6}$$

Precision mewakili proporsi sampel positif yang diklasifikasikan dengan benar terhadap jumlah total sampel yang diprediksi positif dan dapat dirumuskan pada (2.7).

$$Precision = \frac{TP}{TP + FP} \tag{2.7}$$

Recall mewakili sampel positif yang diklasifikasikan dengan benar ke total jumlah sampel positif dan dapat dirumuskan pada Persamaan (2.8).

$$Recall = \frac{TP}{TP + FN} \tag{2.8}$$

F1-Score mewakili rata-rata harmonik dari *precision* dan *recall* dan dapat dirumuskan pada Persamaan (2.9). Nilai F1-Score berkisar antara nol hingga satu, dan nilai tinggi F1-Score menunjukkan kinerja klasifikasi yang tinggi [14].

$$F1 - Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{2.9}$$

2.7 Python

Python adalah salah satu bahasa pemograman yang populer saat ini. Python sering dimanfaatkan dalam pengembangan web, perangkat lunak, penelitian, dan *system scripting*. Python dapat digunakan untuk menangani data besar dan melakukan operasi

Gambar 2.13 The Extended Cohn-Kanade AU-Coded Facial Expression Database (CK+) [5]

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 [15].

2.8 The Extended Cohn-Kanade AU-Coded Facial Expression Database (CK+)

The Extended Cohn-Kanade AU-Coded Facial Expression Database (CK+) merupakan sekumpulan citra wajah manusia dengan total 593 citra berurutan dari 123 subjek dengan 8 kelas ekspresi antara lain netral dengan 123 subjek, marah dengan 45 subjek, senang dengan 69 subjek, sedih dengan 28 subjek, jijik dengan 59 subjek, terkejut dengan 83 subjek, takut dengan 25 subjek, dan menghina dengan 18 subjek. Citra pada dataset ini memiliki ukuran citra 640×490 atau 640×480 piksel. Contoh citra untuk setiap kelas ekspresi wajah pada dataset CK+ dapat dilihat pada Gambar 2.13 [5].

2.9 The Indonesian Mixed Emotion Dataset (IMED)

The Indonesian Mixed Emotion Dataset (IMED) merupakan dataset ekspresi wajah berbasis video dari Universitas Indonesia. IMED memiliki 19 kategori emosi yang dilakukan oleh 15 subjek, semuanya adalah orang Indonesia dengan berbagai etnis antara lain Jawa, Sunda, Melayu, Batak, Minang, dan Manado. Subjek adalah

60% perempuan dan 40% laki-laki dengan usia mulai dari 17 hingga 32 tahun yang menunjukkan kelas emosi dasar dan campuran dalam video.

Dataset IMED memiliki kategori emosi atau kelas antara lain netral, senang, sedih, jijik, marah, takut, terkejut, senang-terkejut, senang-jijik, sedih-terkejut, sedih-takut, sedih-jijik, takut-terkejut, jijik-terkejut, marah-terkejut, takut-marah, takut-jijik, dan marah-jijik. Contoh citra untuk setiap kelas ekspresi wajah dataset IMED dapat dilihat pada Gambar 2.14 [6].

Gambar 2.14 The Indonesian Mixed Emotion Dataset (IMED) [6]

2.10 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, MTCNN, TensorFlow, OpenCV, Numpy, Scikit-learn, dan Matplotlib.

2.10.1 MTCNN

MTCNN merupakan *library open source* yang digunakan untuk mendeteksi wajah pada citra. MTCNN sendiri merupakan implementasi dari paper *Multi-task Cascaded Convolutional Networks* yang dibangun menggunakan *library* Keras dan OpenCV. *Output* dari MTCNN adalah *bounding box* wajah yang terdeteksi, titik mata, hidung, dan mulut, dan *confidence score* [16].

2.10.2 TensorFlow

TensorFlow merupakan *library open source* yang dikembangkan oleh tim Google Brain untuk pembuatan program yang membutuhkan komputasi numerik berkinerja tinggi. TensorFlow menyediakan berbagai macam fungsi *machine learning* dan *deep learning* yang dapat dijalankan dalam CPU atau GPU [17].

2.10.3 OpenCV

OpenCV (*Open Source Computer Vision*) merupakan *library* yang digunakan pada pengolahan citra secara dinamis dan *real-time*. OpenCV dapat digunakan pada berbagai bahasa pemograman seperti Python, C++, Java, dan MATLAB. OpenCV memiliki fitur seperti *Feature and Object Detection, Motion Analysis and Object Tracking, Image Filtering,* dan *Image Processing* [18].

2.10.4 Numpy

Numpy merupakan *library* untuk bahasa pemrograman Python yang bersifat *open source* dan mendukung pengolahan data pada *array* dan matriks multidimensi yang besar sehingga banyak digunakan dalam berbagai macam penelitian. Numpy menyediakan berbagai macam fungsi matematika, seperti aljabar linear, transformasi Fourier, dan pembuatan angka acak [19].

2.10.5 Scikit-learn

Scikit-learn merupakan *machine learning library* untuk bahasa pemrograman Python yang bersifat *open source.* Scikit-learn menyediakan fitur seperti *classification*, *regression*, *clustering*, termasuk algoritma *support vector machines*, *random forest*, dan *gradient boosting* [20].

2.10.6 Matplotlib

Matplotlib merupakan *library* untuk bahasa pemrograman 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 dan plot pencar [21].

BAB III
PERANCANGAN SISTEM

Bab ini menjelaskan tentang perancangan data dan sistem pengenalan ekspresi wajah menggunakan YOLO. Bab ini juga akan menjelaskan gambaran umum sistem dalam bentuk diagram alir.

3.1 Perancangan Data

Dataset yang digunakan untuk melakukan *training* dan *testing* pada YOLO adalah *The Extended Cohn-Kanade Dataset* (CK+) dan *The Indonesian Mixed Emotion Dataset* (IMED) yang telah dijelaskan pada subbab 2.8 dan 2.9. Tabel spesifikasi mengenai dataset CK+ dapat dilihat pada Tabel 3.1 dan untuk dataset IMED dapat dilihat pada Tabel 3.2. Terdapat 7 kategori emosi atau kelas yang digunakan pada tugas akhir ini antara lain senang, sedih, marah, jijik, takut, terkejut dan netral.

Tabel 3.1 Spesifikasi Dataset CK+

Keterangan	**Spesifikasi**
Ukuran resolusi asli	640×480 atau 640×490
Ekstensi	.png
Jumlah kelas	8 kelas
Ukuran file	100 - 200 kB
Kanal warna	1 (Grayscale) atau 3 (RGB)

Tabel 3.2 Spesifikasi Dataset IMED

Keterangan	**Spesifikasi**
Ukuran resolusi asli	720×480
Ekstensi	.jpg
Jumlah kelas	19 kelas
Ukuran file	100 - 200 kB
Kanal warna	3 (RGB)

Gambar 3.1 Contoh Posisi Video Ekspresi Wajah Senang

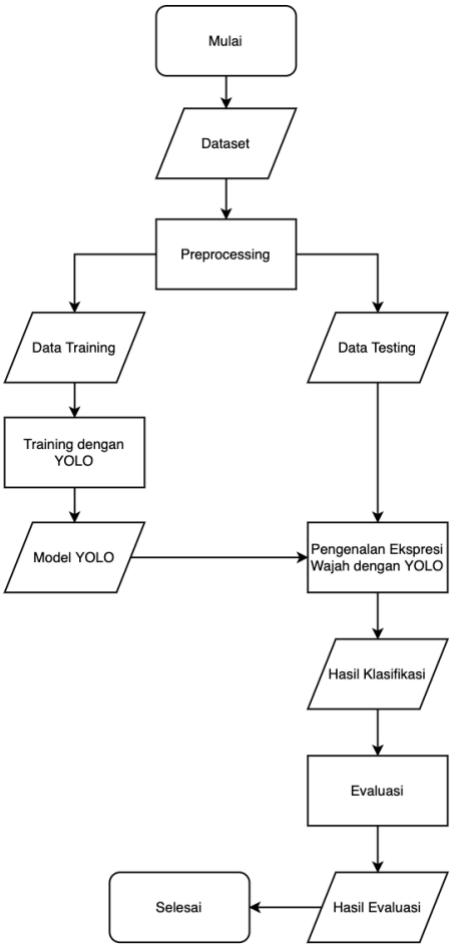
Selain dataset CK+ dan IMED, penulis juga mengumpulkan video ekspresi wajah dari 8 subjek yang merupakan mahasiswa Teknik Informatika ITS angkatan tahun 2016 dan 2017 sebagai data *training* dan *testing* untuk evaluasi YOLO. Setiap subjek video ekspresi ini memiliki 7 video yang setiap videonya memiliki satu ekspresi wajah. Video ekspresi wajah ini memiliki berbagai macam posisi wajah seperti menghadap kamera, menengok ke kanan sedikit, menegok ke kiri sedikit, mematahkan kepala ke kanan, dan mematahkan kepala ke kiri. Contoh posisi dari salah satu video dengan kelas ekspresi wajah senang dapat dilihat pada Gambar 3.1. Untuk spesifikasi video dapat dilihat pada Tabel 3.3.

Tabel 3.3 Spesifikasi Data Video Ekspresi Wajah

Keterangan	**Spesifikasi**
Ukuran resolusi asli	640×480, 960×540, 1280×720, 1920×1080
Ekstensi	.mp4
Jumlah kelas	7 kelas
Ukuran file	2 MB - 35 MB
Kanal warna	3 (RGB)

3.2 Desain Umum Sistem

Sistem pengenalan ekspresi wajah yang dibangun memiliki proses utama diantaranya *preprocessing*, *training* dengan YOLO,



Gambar 3.2 Diagram Alir Sistem yang Dibangun

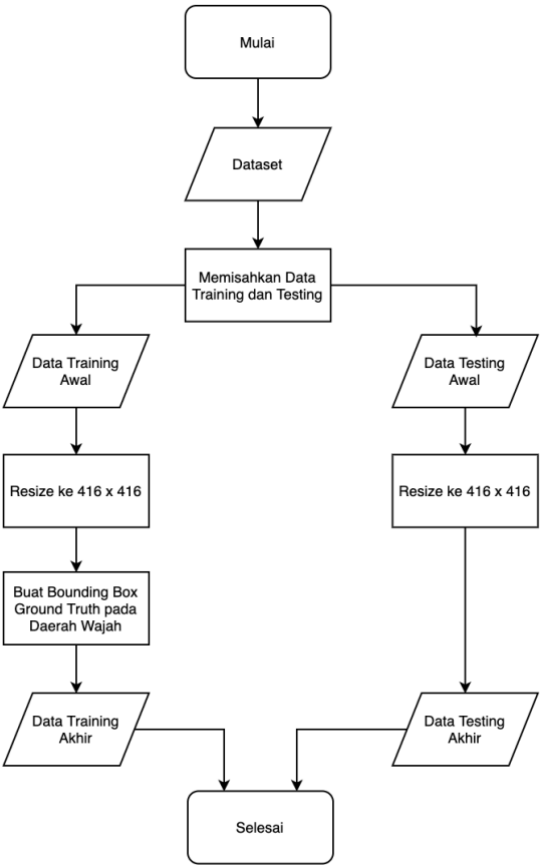
pengenalan ekspresi wajah dengan YOLO, dan evaluasi. Diagram alir dari sistem ditunjukkan pada Gambar 3.2.

3.2.1 Preprocessing

Tahap *preprocessing* dilakukan dengan memisahkan dataset CK+, IMED dan data video menjadi data *training* dan data *testing* dengan jumlah subjek yang berbeda. Jumlah subjek untuk *training* dan *testing* untuk setiap kelas ekspresi ditentukan agar jumlah data antar setiap kelas ekspresi menjadi rata. Untuk dataset CK+, 25 subjek masuk ke data *training* dan 10 subjek lainnya masuk ke data *testing*. Sedangkan untuk IMED, 10 subjek masuk ke data *training* dan 5 lainnya subjek masuk ke data *testing*. Untuk data video, 5 subjek masuk ke data *training* dan 3 subjek lainnya masuk ke data *testing*.

Dataset CK+ dan IMED merupakan dataset yang berbasis video, maka setiap kelas ekspresi pada setiap subjek terdiri dari *sequence* citra wajah dengan ekspresi. Pada dataset CK+ mempunyai *sequence* citra wajah dimulai dari ekspresi netral hingga ekspresi puncak. Untuk kelas ekspresi netral, diambil 3 citra pertama dari kelas ekspresi pada setiap subjek. Untuk ekspresi lainnya diambil 3 citra terakhir dari kelas ekspresi pada setiap subjek. Sedangkan pada dataset IMED mempunyai *sequence* citra wajah dimulai dari ekspresi netral, ekspresi puncak, dan kembali ke ekspresi netral, sehingga untuk setiap kelas diambil 5 citra dibagian puncak ekspresi. Pada data video, 3 citra diambil untuk setiap posisi wajah untuk data *training*, dan 2 citra untuk setiap posisi wajah untuk data testing.

Setelah itu, citra pada data *training* dan *testing* dilakukan *resize* menjadi 416×416 piksel. Setiap citra pada data *training* dibuat *bounding box ground truth* pada daerah wajah menggunakan *Multi-task Cascaded Convolutional Networks* (MTCNN) untuk mempermudah pembuatan *bounding box ground truth*. Setelah itu, Jika ada *bounding box ground truth* yang kurang sesuai atau wajah citra tidak terdeteksi, maka dilakukan perbaikan atau pembuatan *bounding box ground truth* secara manual.



Gambar 3.3 Diagram Alir Tahap Preprocessing

Bounding box ground truth digunakan untuk melakukan *training* dan *testing* pada YOLO. Diagram alir untuk tahap *preprocessing* dapat dilihat pada Gambar 3.3.

3.2.2 Training dengan YOLO

Tahap *training* dilakukan dengan menggunakan *pretrained* model dari YOLOv3 yang dilakukan pada dataset ImageNet dan menggunakan data *training* yang telah melalui tahap *preprocessing*. Tahap *training* dilakukan dengan menggunakan *hyperparameter* bawaan dengan *batch size* 64, *subdivision* 16, dan *max batches* 6000. *Max batches* berfungsi seperti iterasi pada tahap *training*. YOLO akan melakukan *training* dengan *batch* yang berisi 64 citra dan dibagi menjadi 16 *subdivision* atau *mini batch* untuk dilakukan *training*. Setelah satu *batch* selesai dilakukan *training* maka dilanjutkan dengan melakukan *training* batch setelahnya pada iterasi selanjutnya. Pada tahap *training*, *Hyperparameter channel, learning rate, momentum, decay*, dan *anchor box* dapat diubah menyesuaikan skenario uji coba. Pada setiap akhir iterasi terdapat proses pengujian model untuk mengetahui seberapa baik model yang dilatih.

3.2.3 Pengenalan Ekspresi Wajah dengan YOLO

Setelah tahap *training* selesai, YOLO akan menghasilkan model yang telah dilakukan *training* dan siap untuk dilakukan pengujian pengenalan ekspresi wajah. YOLO diuji menggunakan data *testing* yang telah dilakukan tahap *preprocessing*. *Hyperparameter* yang dapat diubah pada tahap ini adalah parameter *channel, momentum,* dan *anchor box* yang nilainya sama pada saat *training*. YOLO akan mendeteksi dan mengklasifikasi kelas ekspresi wajah pada setiap citra *testing* menggunakan model YOLO yang telah dilakukan *training* dan selanjutnya akan menghasilkan *bounding box* yang berisi posisi, hasil klasifikasi dan *confidence score*.

3.2.4 Evaluasi

Hasil klasifikasi yang telah didapatkan melalui tahap deteksi dan klasifikasi akan digunakan untuk melakukan evaluasi pada model YOLO yang telah di lakukan *training*. Tahap evaluasi berfungsi untuk mengukur seberapa baik model dapat memprediksi

kelas ekspresi wajah dengan benar pada data yang belum pernah dilihat oleh model. Pada akhir pengujian, akan dilakukan perhitungan untuk mendapatkan nilai *accuracy*, *precision*, *recall* dan *F1-Score*.

(Halaman ini sengaja dikosongkan)

BAB IV
IMPLEMENTASI

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

4.1 Lingkungan Implementasi

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

4.1.1 Perangkat Keras

Implementasi tugas akhir ini menggunakan aplikasi web Google Colaboratory yang memiliki spesifikasi 2 Intel(R) Xeon(R) CPU @ 2.20GHz, *Random Access Memory* (RAM) sebesar 13 GB, dan mempunyai *Graphics Processing Unit* (GPU) yaitu NVIDIA Tesla K80 dengan RAM sebesar 12 GB.

4.1.2 Perangkat Lunak

Perangkat lunak yang digunakan antara lain menggunakan bahasa pemograman Python 3.6, dilengkapi dengan *library* antara lain MTCNN, OpenCV, TensorFlow-GPU, Numpy, Matplotlib, dan Scikit-learn. Perangkat lunak berupa darknet juga digunakan untuk melakukan *training* pada YOLO.

4.2 Implementasi Tahap Preprocessing

Tahap *preprocessing* dimulai dari memisahkan dataset menjadi data *training* dan data *testing*. Pada dataset CK+, 25 subjek masuk ke dalam data *training* dan 10 subjek lainnya masuk ke dalam data *testing*. Implementasi pemisahan dataset pada CK+ dapat dilihat pada Kode Sumber 4.1 untuk kelas netral dan Kode Sumber 4.2 untuk kelas yang lain.

Pada Kode Sumber 4.1, Baris 4 hingga 6 berfungsi untuk mendapatkan *path* atau alamat *folder* dari setiap subjek. Dataset CK+ tidak memiliki kelas netral secara eksplisit atau terpisah dengan kelas ekpresi wajah lainnya, tetapi setiap kelas memiliki *sequence* citra yang dimulai dari ekspresi netral, sehingga 3 citra pertama dipilih untuk masuk ke dalam data *training* dan *testing* untuk kelas netral seperti dengan implemetasi pada baris 8 hingga 18.

```
1. # Mendapatkan dataset dengan kelas NETRAL
2. for path in subject_folder_paths:
3.     # Dapatkan semua subpath dari folder subjek
4.     subpath = sorted(glob.glob(path+"/*/"))
5.     # Dapatkan path pertama dari folder subjek
6.     first_face_paths = sorted(glob.glob(subpath[0]+'/*.p
   ng'))
7.     # Jika counter kelas netral kurang dari jumlah subje
   k training
8.     if counter[0] < train_subject:
9.             # Dapatkan tiga gambar pertama
10.            for face_file_path in first_face_paths[:3]:

11.                # Lakukan preprocessing untuk setiap ga
   mbar
12.                extract_dataset(face_file_path,os.path.
   join(full_train, 'neutral'),0, 'train')
13.        # Jika counter kelas netral lebih dari jumlah s
   ubjek training
14.        elif counter[0] < total_subject:
15.            # Dapatkan tiga gambar pertama
16.            for face_file_path in first_face_paths[:3]:

17.                # Lakukan preprocessing untuk setiap ga
   mbar
18.                extract_dataset(face_file_path,os.path.
   join(full_test, 'neutral'),0, 'test')
19.        # Update counter untuk kelas NETRAL
20.        counter[0] += 1
```

Kode Sumber 4.1 Preprocessing Dataset CK+ untuk Kelas Netral

Pada Kode Sumber 4.2, Baris 4 berfungsi untuk mendapatkan *path* atau alamat *folder* dari setiap subjek. Baris 6 hingga 7 berfungsi untuk mendapatkan file yang berisi label kelas ekspresi wajah *ground truth*. Dataset CK+ yang akan masuk ke dalam data *training* dan *testing* hanya sequence citra yang memiliki *file* yang berisi label kelas ekspresi wajah *ground truth* dan tidak memiliki label contempt atau menghina, selain itu maka diabaikan, seperti yang ditunjukkan pada baris 9 hingga 15. Setiap kelas selain netral akan diambil 3 gambar terakhir untuk masuk ke dalam data *training* dan *testing*, seperti yang ditunjukkan pada baris 18 hingga 28.

Kelas *fear* atau takut memiliki 25 subjek dan karena jumlah subjek yang masuk ke dalam data *training* adalah 25 subjek, maka semua subjek pada kelas takut akan masuk ke dalam data *training*. Untuk data *testing* pada kelas takut akan diambil 3 gambar sebelum 3 gambar terakhir dari 10 subjek, seperti yang ditunjukkan pada baris 31 hingga 34.

Kelas *sadness* atau sedih memiliki 28 subjek dan karena jumlah subjek yang masuk ke dalam data *training* adalah 25 subjek, maka 25 subjek pada kelas sedih akan masuk ke dalam data *training* dan menyisakan 3 subjek untuk data *testing*. Untuk data *testing* selain 3 subjek sisa pada kelas takut akan diambil 3 gambar sebelum 3 gambar terakhir dari subjek yang lain selain 3 subjek sisa, seperti yang ditunjukkan pada baris 37 hingga 41.

```
1. # Mendapatkan dataset dengan kelas selain NETRAL
2. for path in face_folder_paths:
3.     # Dapatkan path gambar dari setiap folder dalam seti
   ap subjek
4.     faces_path = sorted(glob.glob(str(path+'/*.png')))
5.     # Dapatkan path emotion yang berisi data kelas groun
   d truth
6.     emotion_folder_paths = path.replace(dataset_folder,e
   motion_folder)
7.     emotions_path = sorted(glob.glob(str(emotion_folder_
   paths+'/*.txt')))
```

```
8.      # Jika file emosi ground
   truth ditemukan, maka lanjut
9.      if (emotions_path):
10.                     # Dapatkan label dari file emosi gr
   ound truth
11.                     label = get_label_from_file(str(emo
   tions_path[0]))
12.                     # Abaikan label = 2 atau label cont
   empt/menghina
13.                     if (label != 2):
14.                         if (label > 2):
15.                             label = label - 1
16.
17.                         # Jika counter kelas label kura
   ng dari jumlah subjek training
18.                         if counter[label] < train_subje
   ct:
19.                             # Dapatkan tiga gambar tera
   khir
20.                             for face_file_path in faces
   _path[-3:]:
21.                                 # Lakukan preprocessing
    untuk setiap gambar
22.                                 extract_dataset(face_fi
   le_path,os.path.join(full_train, labels[label]),label,
   'train')
23.                         # Jika counter label lebih dari
    jumlah subjek training
24.                         elif counter[label] < total_sub
   ject:
25.                             # Dapatkan tiga gambar pert
   ama
26.                             for face_file_path in faces
   _path[-3:]:
27.                                 # Lakukan preprocessing
    untuk setiap gambar
28.                                 extract_dataset(face_fi
   le_path,os.path.join(full_test, labels[label]),label,
   'test')
29.
30.                         ## Dapatkan data testing untuk
   label fear/takut
```

```
31.                         if label == 3 and counter[label
   ] < test_subject:
32.                             # Dapatkan tiga gambar sebe
   lum tiga gambar terakhir
33.                             for face_file_path in faces
   _path[-6:-3]:
34.                                 extract_dataset(face_fi
   le_path,os.path.join(full_test, labels[label]),label,
   'test')
35.
36.                         ## Dapatkan data testing yang l
   ain untuk label sad/sedih
37.                         if label == 5 and counter[label
   ] < (test_subject-3):
38.                             # Dapatkan tiga gambar sebe
   lum tiga gambar terakhir
39.                             for face_file_path in faces
   _path[-6:-3]:
40.                                 # Lakukan preprocessing
    untuk setiap gambar
41.                                 extract_dataset(face_fi
   le_path,os.path.join(full_test, labels[label]),label,
   'test')
42.                         # Update counter untuk label ya
   ng bersangkutan
43.                         counter[label] += 1
```

Kode Sumber 4.2 Preprocesing Dataset CK+ untuk Kelas Selain Netral

Pada dataset IMED, 10 subjek masuk ke dalam data *training* dan 5 subjek lainnya masuk ke dalam data *testing*. Implementasi pemisahan dataset pada dataset IMED dapat dilihat pada Kode Sumber 4.3. Baris 2 hingga 4 berfungsi untuk mendapatkan *path* gambar dan mengurutkannya secara alfabetik. Baris 7 berfungsi untuk mendapatkan nilai tengah dari jumlah gambar pada *folder*. Baris 5 hingga 19 berfungsi untuk mengambil 5 gambar yang berada di urutan tengah *folder* dan memasukkannya ke dalam data *training* dan data *testing*. Gambar dibagian tengah diambil karena merupakan ekspresi wajah puncak.

```
1. # Dapatkan path gambar
2. faces_path = glob.glob(str(path+'/*.jpg'))
3. # Melakukan sort nama file gambar
4. faces_path.sort(key=lambda s: [int(t) if t.isdigit() els
   e t.lower() for t in re.split('(\d+)', s)])
5.
6. # Dapatkan nilai tengah dari jumlah gambar pada folder
7. mid = int(len(faces_path)/2)
8. # Jika counter label kurang dari jumlah subjek training

9. if counter[i] < train_subject:
10.         # Dapatkan 5 gambar pertengahan
11.         for face_file_path in faces_path[mid-
   3:mid+2]:
12.             # Lakukan preprocessing untuk setiap gambar

13.             extract_dataset(face_file_path,os.path.join
   (full_train, labels[i]),i, 'train')
14.     # Jika counter label lebih dari jumlah subjek train
   ing
15.     elif counter[i] < (train_subject + test_subject):
16.         # Dapatkan 5 gambar pertengahan
17.         for face_file_path in faces_path[mid-
   3:mid+2]:
18.             # Lakukan preprocessing untuk setiap gambar

19.             extract_dataset(face_file_path,os.path.join
   (full_test, labels[i]),i, 'test')
20.     # Update counter untuk label yang bersangkutan
21.     counter[i] += 1
```

Kode Sumber 4.3 Preprocessing Dataset IMED

Pada data video, 3 gambar dipilih dari *sequence* citra data video untuk setiap posisi wajah untuk setiap ekspresi wajah pada setiap subjek secara manual. Implementasi pemisahan dataset pada data video dapat dilihat pada Kode Sumber 4.4. Pada baris 3, 5 subjek dipilih secara acak untuk data *training* dengan 3 gambar untuk setiap posisi wajah. Pada baris 5, 3 subjek lainnya masuk ke dalam data *testing* dengan 2 gambar untuk setiap posisi wajah. Pada baris 9 hingga 11 berfungsi untuk mendapatkan path dari file gambar.

Pada baris 16 berfungsi untuk mengambil 3 gambar untuk data *training* dan melakukan *preprocesssing* lanjutan. Pada baris 20 hingga 22 berfungsi untuk mengambil 2 gambar untuk data *testing* dan melakukan *preprocesssing* lanjutan.

```
1. for i,label in enumerate(labels):
2.     # pilih 5 subjek training secara acak
3.     train_subjects = random.sample(subjects, train_subje
   ct_num)
4.     # 3 subjek lain masuk ke data testing
5.     test_subjects = list(set(subjects) - set(train_subje
   cts))
6.
7.     for num,subject in enumerate(subjects):
8.         # Dapatkan file path gambar
9.         face_folder_paths = os.path.join(subject_paths[n
   um],label)
10.            faces_path = glob.glob(str(face_folder_path
   s+'/*.jpg'))
11.            faces_path.sort(key=lambda s: [int(t) if t.
   isdigit() else t.lower() for t in re.split('(\d+)', s)])

12.
13.            if subject in train_subjects:
14.                # lakukan preprocessing lanjutan pada d
   ata training
15.                for face_file_path in faces_path:
16.                    extract_dataset(face_file_path,os.p
   ath.join(full_train, labels[i]),i,'train')
17.            else:
18.                # lakukan preprocessing lanjutan pada d
   ata testing
19.                # ambil 2 gambar untuk setiap posisi
20.                faces_path = [l for num,l in enumerate(
   faces_path) if (num+1)%3 != 0]
21.                for face_file_path in faces_path:
22.                    extract_dataset(face_file_path,os.p
   ath.join(full_test, labels[i]),i,'test')
```

Kode Sumber 4.4 Preprocessing Data Video

Pada fungsi *extract_dataset* yang ditemukan pada Kode Sumber 4.1, Kode Sumber 4.2, dan Kode Sumber 4.3, melakukan *preprocessing* lanjutan dengan melakukan *resize* citra menjadi 416×416 piksel pada baris 19, setiap citra diambil daerah wajahnya menggunakan *Multi-task Cascaded Convolutional Networks* (MTCNN) pada baris 23. Daerah wajah hasil MTCNN yang berupa *bounding box* ini disimpan pada sebuah file dengan nama yang sama dengan nama file gambar yang ditunjukkan pada baris 25 hingga 33. Daftar file data *training* dan *testing* juga disimpan ada sebuah file teks yang ditunjukkan pada baris 38 hingga 43. Implementasi dari fungsi *extract_dataset* dapat dilihat pada Kode Sumber 4.5.

```
1.  # Import library MTCNN
2.  from mtcnn import MTCNN
3.  detector = MTCNN()
4.
5.  def extract_dataset(file_path,label_path,label,mode):
6.      size = 416
7.      file_name = os.path.basename(file_path)
8.      file_name = file_name.replace('png','jpg')
9.      label_fn = file_name.replace('jpg','txt')
10.     file_name = file_name.replace(' ','_')
11.
12.     path = Path(label_path)
13.     path.mkdir(parents=True, exist_ok=True)
14.
15.     rel_path = '/'.join(label_path.split('/')[-2:])
16.     rel_path = os.path.join('data/obj/', rel_path)
17.
18.     img = cv2.imread(file_path)
19.     img = cv2.resize(img, (size,size), interpolation =
    cv2.INTER_AREA)
20.     rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
21.     if mode == 'train':
22.         # Deteksi wajah dengan MTCNN
23.         detect = detector.detect_faces(rgb)
24.         # Jika wajah terdeteksi
25.         if detect:
26.             # Ambil koordinat bounding box
```

```
27.             x, y, w, h = detect[0]['box']
28.             xc = int(x+w/2)
29.             yc = int(y+h/2)
30.             print(x,y,w,h)
31.
32.             with open(os.path.join(label_path,label_fn)
    ,"w") as f:
33.                 f.write("{0} {1} {2} {3} {4}\n".format(
    label,xc/size,yc/size,w/size,h/size))
34.     # menyimpan file gambar yang telah dipreprocessing

35.     cv2.imwrite(os.path.join(label_path, file_name), im
    g)
36.
37.     # menyimpan daftar nama file dengan kelas
38.     with open(os.path.join(path.parent,'path_'+mode+'_w
    ith_class.txt'),"a+") as f:
39.         f.write("{0},{1}\n".format(os.path.join(rel_pat
    h, file_name),label))
40.
41.     # menyimpan daftar nama file tanpa kelas
42.     with open(os.path.join(path.parent,'path_'+mode+'.t
    xt'),"a+") as f:
43.         f.write("{0}\n".format(os.path.join(rel_path, f
    ile_name)))
```

Kode Sumber 4.5 Implementasi Fungsi extract_dataset

Setelah *preprocessing* dataset CK+, IMED dan data video selesai maka masing-masing dataset akan memiliki data *training* dan *testing* yang didalamnya sudah dikelompokkan berdasarkan kelas. Data *training* CK+ dapat digabung ke dalam data *training* IMED dan data video dan data *testing* CK+ dapat digabung ke dalam data *testing* IMED dan data video untuk membuat satu dataset yang siap dilakukan *training* dan *testing*.

4.3 Implementasi Tahap Training dengan YOLO

Tahap *training* dilakukan dengan menggunakan kerangka kerja darknet yang dibangun dengan bahasa pemrograman C. Pada

Kode Sumber 4.6, darknet harus di-*download* dengan melakukan *clone* repository github seperti yang ditunjukkan pada baris 1. Lalu masuk ke dalam *folder* darknet seperti yang ditunjukkan pada baris 2. Baris 3 hingga 5 berfungsi untuk mengubah file Makefile sehingga mendukung OpenCV dan dapat dijalankan menggunakan GPU. *File* Makefile digunakan sebagai file konfigurasi untuk menginstal darknet. Baris 6 berfungsi untuk menginstal darknet sehingga siap digunakan untuk melakukan *training*. Baris 7 berfungsi untuk melakukan *download* file *pretrained* model darknet dari ImageNet. *Pretrained* model ini digunakan untuk melakukan *training*.

```
1. !git clone https://github.com/AlexeyAB/darknet/
2. %cd darknet
3. !sed -i 's/OPENCV=0/OPENCV=1/g' Makefile
4. !sed -i 's/GPU=0/GPU=1/g' Makefile
5. !sed -i 's/CUDNN=0/CUDNN=1/g' Makefile
6. !make
7. !wget https://pjreddie.com/media/files/darknet53.conv.74
```

Kode Sumber 4.6 Instalasi Darknet

Folder data *training* dan data *testing* diletakkan ke dalam *folder data* pada *folder* darknet. *Path* gambar dalam file path_train.txt dan path_test.txt disesuaikan ke *path* tempat data *training* dan *testing* berada dan kedua file tersebut diletakkan pada *folder* data dalam *folder* darknet. Setelah itu, file *obj.names* dibuat pada *folder* data dalam *folder* darknet yang berisi nama kelas ekspresi wajah yang ditunjukkan pada Kode Sumber 4.7.

```
1. neutral
2. angry
3. disgust
4. fear
5. happy
6. sadness
7. surprise
```

Kode Sumber 4.7 File obj.names

File *obj.data* dibuat pada *folder* utama darknet. File ini berisi konfigurasi dataset yang digunakan untuk *training*. Konfigurasi tersebut dapat dilihat pada Kode Sumber 4.8. Baris 1 menyatakan jumlah kelas pada dataset. Baris 2 menyatakan letak file yang berisi *path* data *training*. Baris 3 menyatakan letak file yang berisi *path* data *testing*. Baris 4 menyatakan letak file nama kelas pada dataset. Baris 5 menyatakan tempat model akan disimpan.

```
1. classes= 7
2. train  = data/train.txt
3. valid  = data/train.txt
4. names = data/obj.names
5. backup = backup/
```

Kode Sumber 4.8 File obj.data

Setelah itu pada file yolov3.cfg yang terletak pada *folder* cfg dilakukan konfigurasi. Parameter *batch* diubah menjadi 64, *subdivisions* menjadi 16, *width* dan *height* menjadi 416, *max_batches* menjadi 6000, dan *steps* menjadi 4800, 5400. Lalu semua parameter *classes* pada layer yolo diubah menjadi 7 menyesuaikan jumlah kelas pada dataset. Setelah itu, semua parameter *filters* pada layer *convolutional* tepat sebelum layer yolo diubah menjadi 36. Parameter *filters* ini didapatkan dengan Persamaan (4.1).

$$filters = (jumlah\ kelas + 5) \times 3 \qquad (4.1)$$

Setelah konfigurasi dilakukan, proses *training* dapat dilakukan dengan Kode Sumber 4.9.

```
1. !./darknet detector train obj.data cfg/yolov3.cfg
   darknet53.conv.74 -map -dont_show
```

Kode Sumber 4.9 Training YOLO

Setelah *training* selesai, model akan tersimpan pada *folder backup*. Model dapat digunakan untuk melakukan pengujian dan evaluasi.

4.4 Implementasi Pengenalan Ekspresi Wajah dengan YOLO

Setelah dataset dilakukan *training* menggunakan YOLO dan menghasilkan model. Model dapat digunakan untuk melakukan pengujian terhadap citra ekpresi wajah yang belum pernah dilihat oleh model. Tetapi, untuk melakukan pengujian menggunakan TensorFlow, model yang sudah dilakukan *training* oleh darknet harus diubah menjadi format yang dapat dibaca oleh TensorFlow.

Pada Kode Sumber 4.10, terdapat fungsi *convert_model* untuk mengubah model yang telah dilakukan *training* oleh darknet menjadi model yang dapat dibaca oleh TensorFlow. Fungsi ini memiliki 2 parameter *input* yaitu path file yang berekstensi *weights* yang merupakan model yang terlah dilakukan *training* oleh darknet dan path file *output* yang berekstensi *ckpt* yang merupakan *output* dari model yang telah diubah. Pada baris 3 hingga 6 berfungsi untuk menyiapkan model YOLOv3 yang berisi arsitektur YOLOv3. Baris 8 hingga 10 berfungsi untuk menyiapkan *output path*. Baris 14 berfungsi untuk menyiapkan dimensi input yang dapat dikenali oleh model. Baris 20 berfungsi untuk membaca model YOLO yang dilakukan *training* oleh darknet. Baris 24 hingga 27 berfungsi untuk mengubah model darknet menjadi model TensorFlow.

```
1. def convert_model(input_path, output_path):
2.     # inisialisasi model YOLOv3
3.     model = Yolo_v3(n_classes=7, model_size=(416, 416),

4.                     max_output_size=5,
5.                     iou_threshold=0.5,
6.                     confidence_threshold=0.5)
7.     # inisialisasi output path
8.     output_folder_path = '/'.join(output_path.split('/')
   [0:-1])
```

```
9.
10.         Path(output_folder_path).mkdir(parents=True, ex
    ist_ok=True)
11.
12.         tf.reset_default_graph()
13.         # menyiapkan dimensi input untuk deteksi
14.         inputs = tf.placeholder(tf.float32, [1, 416, 41
    6, _CHANNEL])
15.
16.         model(inputs, training=False)
17.
18.         model_vars = tf.global_variables(scope='yolo_v3
    _model')
19.         # membaca model yolov3 darknet
20.         assign_ops = load_weights(model_vars, input_pat
    h)
21.
22.         saver = tf.train.Saver(tf.global_variables(scop
    e='yolo_v3_model'))
23.         # mengubah model darknet menjadi model TensorFl
    ow
24.         with tf.Session() as sess:
25.             sess.run(assign_ops)
26.             saver.save(sess, output_path)
27.             print('Model has been saved successfully.')

28.             sess.close()
```

Kode Sumber 4.10 Mengubah Model Darknet ke Model TensorFlow

Pada Kode Sumber 4.11. terdapat fungsi bernama *detect_img_batch* yang berfungsi untuk mendeteksi citra ekspresi wajah secara *batch*. Fungsi ini mempunyai 5 parameter yaitu *path* model yang telah diubah menjadi model TensorFlow, *threshold* IOU, *confidence score threshold*, *list file* citra yang akan dideteksi dan diklasifikasi, dan jumlah citra dalam 1 *batch*. Pada baris 5 hingga 6 berfungsi untuk menyiapkan nama-nama kelas ekspresi wajah. Baris 10 menyiapkan model YOLOv3 untuk melakukan deteksi citra ekspresi wajah. Baris 15 berfungsi untuk membaca semua citra yang berasal dari *list path* citra ekspresi wajah. Baris

28 hingga 37 berfungsi untuk melakukan deteksi dan klasifikasi ekspresi wajah untuk setiap *batch* yang diukur waktu deteksinya. Tahap deteksi dan klasifikasi akan menghasilkan posisi *bounding box*, prediksi kelas ekpresi wajah, dan *confidence score* untuk semua file citra yang diuji akan masuk ke dalam variabel *detection_result*.

```
1. def detect_img_batch(model_path, iou_threshold, confiden
   ce_threshold, input_names, batch_size):
2.     global detection_result
3.     # Memuat nama-nama kelas
4.     # _CLASS_NAMES_FILE = letak file kelas
5.     class_names = load_class_names(_CLASS_NAMES_FILE)
6.     n_classes = len(class_names)
7.     # Memuat YOLOv3 model
8.     # _MODEL_SIZE = (416 x 416)
9.     # _MAX_OUTPUT_SIZE = 1
10.        model = Yolo_v3(n_classes=n_classes, model_size
   =_MODEL_SIZE,
11.                        max_output_size=_MAX_OUTPUT_SIZ
   E,
12.                        iou_threshold=iou_threshold,
13.                        confidence_threshold=confidence
   _threshold)
14.        # membaca semua gambar pada list nama gambar
15.        batch = load_images(input_names, model_size=_MO
   DEL_SIZE)
16.        # inisialisasi perhitungan waktu deteksi
17.        time_total = 0
18.        # inisialisasi hasil deteksi
19.        detection_result = []
20.
21.        for b in create_batch(batch, batch_size):
22.          tf.reset_default_graph()
23.          # _CHANNEL =  jumlah kanal warna pada model y
   ang telah ditraining
24.          inputs = tf.placeholder(tf.float32, [len(b),
   *_MODEL_SIZE, _CHANNEL])
25.          detections = model(inputs, training=False)
26.          saver = tf.train.Saver(tf.global_variables(sc
   ope='yolo_v3_model'))
```

```
27.          # melakukan deteksi untuk setiap batch gambar

28.          with tf.Session() as sess:
29.              saver.restore(sess, model_path)
30.              # mendapatkan waktu awal deteksi
31.              start = time.time()
32.              # melakukan deteksi dan menyimpan hasil d
   eteksi ke dalam list
33.              detection_result.append(sess.run(detectio
   ns, feed_dict={inputs: b}))
34.              # mendapatkan waktu akhir deteksi
35.              end = time.time()
36.              # mendapatkan hasil waktu deteksi
37.              time_total = time_total + (end-start)
38.
39.          print("Detection Time:", time_total)
40.          print('Detections have been saved successfully.
   ')
```

Kode Sumber 4.11 Melakukan Deteksi dan Klasifikasi Citra Ekspresi Wajah

4.5 Implementasi Evaluasi

Setelah melakukan tahap deteksi dan klasifikasi citra ekspresi wajah dan mendapatkan hasil deteksi yang berupa posisi *bounding box*, prediksi kelas ekspresi wajah, dan *confidence score*, maka tahap evaluasi dapat dilakukan dengan membandingkan prediksi kelas ekpresi wajah dengan kelas yang sebenarnya atau *ground truth*.

Pada Kode Sumber 4.12, terdapat fungsi *evaluate* yang berfungsi untuk mengevaluasi hasil dari deteksi dan klasifikasi YOLO. Fungsi ini mempunyai 3 parameter *input* yang berupa *list* dari hasil deteksi yang telah dilakukan pada tahap sebelumnya, nama-nama kelas ekspresi wajah, dan list kelas *ground truth*. Pada baris 10 hingga 23 berfungsi untuk mengambil kelas prediksi dari *list* hasil deteksi, lalu kelas prediksi akan dimasukkan ke dalam sebuah *list*. Baris 26 berfungsi untuk melakukan evaluasi menggunakan *library* scikit-learn yang akan menghasilkan *output*

accuracy (akurasi), *precision* (presisi), *recall*, dan *F1-Score*. Baris 27 hingga 30 berfungsi untuk menghasilkan *confusion matrix*.

```
1.  from sklearn import metrics
2.  import seaborn as sns
3.  import pandas as pd
4.
5.  def evaluate(results, class_names, y_true):
6.    y_pred = []
7.    results = sum(results, []) # Menggabungkan isi list
8.    total = len(results)
9.
10.   for idx in range(total):
11.     box = results[idx]
12.     # menghilangkan kelas prediksi yang tidak mempunyai
     bounding box
13.     box = {k: v for k, v in box.items() if np.size(v) !
    = 0}
14.     box = {k: v for k, v in sorted(box.items(), key=lam
    bda item: item[1][0][4])}
15.     # mengambil kelas prediksi
16.     list_box_keys = list(box.keys())
17.     k = 8
18.     if list_box_keys:
19.       k = list(box.keys())[0]
20.     elif ('no_detect' not in class_names):
21.       class_names.append('no_detect')
22.
23.     y_pred.append(int(k))
24.   # menampilkan hasil evaluasi dengan classificaion rep
    ort
25.   # dan confusion matrix
26.   print(metrics.classification_report(y_true, y_pred, t
    arget_names=class_names))
27.   conf_mat = metrics.confusion_matrix(y_true, y_pred)
28.   df_cm = pd.DataFrame(conf_mat, index = [i for i in cl
    ass_names],
29.                        columns = [i for i in class_names])
30.   sns.set(font_scale=1.4)
31.   sns.heatmap(df_cm,annot=True, annot_kws={"size": 16})
```

Kode Sumber 4.12 Evaluasi YOLO

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 menggunakan aplikasi web Google Colaboratory yang memiliki spesifikasi 2 Intel(R) Xeon(R) CPU @ 2.20 GHz, *Random Access Memory* (RAM) sebesar 13 GB, dan mempunyai *Graphics Processing Unit* (GPU) yaitu NVIDIA Tesla K80 dengan RAM sebesar 12 GB. Pada sisi perangkat lunak, uji coba pada tugas akhir ini dilakukan dengan menggunakan bahasa pemograman Python 3.6 dilengkapi dengan *library* antara lain MTCNN, TensorFlow, OpenCV, Numpy, Matplotlib, dan Scikit-learn.

Tabel 5.1 Spesifikasi Dataset Training dan Testing

Keterangan	**Spesifikasi**
Ukuran gambar	416×416
Ekstensi	.jpg
Jumlah kelas	7 kelas
Jumlah total file data *training*	**1400**
Jumlah file data *training* CK+	525
Jumlah file data *training* IMED	350
Jumlah file data *training* Data Video	525
Jumlah total file data *testing*	**595**
Jumlah file data *testing* CK+	210
Jumlah file data *testing* IMED	175
Jumlah file data *testing* Data Video	210
Ukuran file	20 - 40 kB
Kanal warna	1 (Grayscale) dan 3 (RGB)

5.2 Deskripsi Dataset

Dataset yang digunakan pada tugas akhir ini adalah dataset CK+, IMED, dan data video yang telah dideskripsikan pada subbab 3.1. Dataset CK+, IMED dan data video dilakukan *preprocessing* dan digabung menjadi satu dataset baru untuk *training* dan *testing*. Dataset *training* dan *testing* ini mempunyai spesifikasi yang ditunjukkan pada Tabel 5.1. Sedangkan data video dapat dilakukan evaluasi dengan menghitung *majority vote* dan membandingkannya dengan kelas *ground truth*.

5.3 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, terdapat skenario uji coba dan hasil uji coba antara lain:

1. Skenario Uji Coba Parameter pada YOLO
2. Skenario Uji Coba berdasarkan Dataset
3. Skenario Uji Coba Perbandingan Metode
4. Skenario Uji Coba pada Data Video

5.3.1 Skenario Uji Coba Parameter pada YOLO

Skenario uji coba parameter pada YOLO bertujuan untuk mencari nilai parameter terbaik yang dapat menghasilkan model dengan performa terbaik. Skenario yang diuji coba pada YOLO adalah dengan mengganti parameter *channel*, *learning rate*, *momentum*, *decay*, dan *anchor box* dari *hyperparameter* awal. Skenario ini diuji coba pada data *testing* dengan 595 citra ekspresi wajah. Spesifikasi *hyperparameter* awal YOLO dapat dilihat pada Tabel 5.2.

Tabel 5.2 Spesifikasi *Hyperparameter* Awal YOLO

Keterangan	**Spesifikasi**
Ukuran gambar input	416×416
Max_batches (iterasi)	6000
Batch	64
Subdivisions	16
Channel (kanal warna)	3 (RGB)
Learning Rate	0,01
Momentum	0,9
Decay	0,0005
Anchor Box	(10, 13), (16, 30), (33, 23), (30, 61), (62, 45), (59, 119), (116, 90), (156, 198), (373, 326)

5.3.1.1 Channel (Kanal Warna)

YOLO diuji coba dengan memvariasikan parameter *channel* atau kanal warna input dengan nilai 1 (*grayscale*) dan 3 (RGB). *Hyperparameter* lain seperti parameter *channel*, *learning rate*, *momentum*, *decay*, dan *anchor box* menggunakan niai dari *hyperparameter* awal. Hasil uji coba penggantian parameter *channel* pada arsitektur YOLO dapat dilihat pada Tabel 5.3.

5.3.1.2 Learning Rate

Learning rate merupakan parameter yang menentukan seberapa cepat model dapat mempelajari masalah. YOLO diuji coba dengan memvariasikan *learning rate* dengan nilai 0,1, 0,01,

Tabel 5.3 Hasil Uji Coba pada Variasi *Channel*

Channel	**Akurasi (%)**	**Precision (%)**	**Recall (%)**	**F1-Score (%)**	**Waktu Deteksi (detik)**
1	64	68	64	64	64,45
3	**68**	**70**	**68**	**68**	**63,56**

Tabel 5.5 Hasil Uji Coba pada Variasi *Learning Rate*

Learning Rate	Akurasi (%)	Precision (%)	Recall (%)	F1-Score (%)	Waktu Deteksi (detik)
0,1	65	68	65	65	63,65
0,01	**68**	**70**	**68**	**68**	**63,56**
0,001	65	68	65	65	64,41

dan 0,001. Parameter *channel* yang digunakan pada skenario uji coba ini adalah *channel* yang mendapatkan hasil terbaik dari pengujian sebelumnya yaitu *channel* 3. Sedangkan parameter *momentum*, *decay*, dan *anchor box* menggunakan nilai dari *hyperparameter* awal. Hasil uji coba penggantian parameter *learning rate* pada arsitektur YOLO dapat dilihat pada Tabel 5.5.

5.3.1.3 Momentum

Momentum merupakan parameter *optimizer* yang mempengaruhi perubahan bobot selanjutnya. YOLO diuji coba dengan memvariasikan *momentum* dengan nilai 0,85, 0,90, dan 0,95. Nilai parameter *channel* dan *learning rate* yang digunakan pada skenario uji coba ini adalah nilai yang mendapatkan hasil terbaik dari pengujian sebelumnya. Sedangkan parameter *decay* dan *anchor box* menggunakan nilai dari *hyperparameter* awal. Hasil uji coba penggantian parameter *momentum* pada arsitektur YOLO dapat dilihat pada Tabel 5.4.

Tabel 5.4 Hasil Uji Coba pada Variasi *Momentum*

Momentum	Akurasi (%)	Precision (%)	Recall (%)	F1-Score (%)	Waktu Deteksi (detik)
0,85	64	67	64	64	64,02
0,90	**68**	**70**	**68**	**68**	**63,56**
0,95	65	67	65	65	64,41

Tabel 5.6 Hasil Uji Coba pada Variasi Decay

Decay	Akurasi (%)	Precision (%)	Recall (%)	F1-Score (%)	Waktu Deteksi (detik)
0,005	67	69	67	67	64,64
0,0005	**68**	**70**	**68**	**68**	**63,56**
0,00005	66	70	66	67	64,91

5.3.1.4 Decay

Decay merupakan parameter *optimizer* yang melakukan perubahan bobot yang lebih lemah dari *momentum*. YOLO diuji coba dengan memvariasikan *decay* dengan nilai 0,005, 0,0005, dan 0,00005. Nilai parameter *channel*, *learning rate*, dan *momentum* yang digunakan pada skenario uji coba ini adalah nilai yang mendapatkan hasil terbaik dari pengujian sebelumnya. Sedangkan parameter *anchor box* menggunakan nilai dari *hyperparameter* awal. Hasil uji coba penggantian parameter *decay* pada arsitektur YOLO dapat dilihat pada Tabel 5.6.

5.3.1.5 Anchor Box

YOLO diuji coba dengan memvariasikan *anchor box* dengan nilai awal atau *default* (10, 13), (16, 30), (33, 23), (30, 61), (62, 45), (59, 119), (116, 90), (156, 198), (373, 326) dan nilai *custom* (85, 136), (99, 140), (99, 170), (109, 163), (118, 181), (120, 203), (135, 198), (157, 228), (179, 261) yang dihitung dengan darknet menggunakan algoritma *k-means*. Nilai parameter

Tabel 5.7 Hasil Uji Coba pada Variasi *Anchor Box*

Anchor Box	Akurasi (%)	Precision (%)	Recall (%)	F1-Score (%)	Waktu Deteksi (detik)
Default	**68**	**70**	**68**	**68**	**63,56**
Custom	66	69	66	66	68,18

channel, *learning rate*, *momentum*, dan *decay* yang digunakan pada skenario uji coba ini adalah nilai yang mendapatkan hasil terbaik dari pengujian sebelumnya. Hasil uji coba penggantian parameter *anchor box* pada arsitektur YOLO dapat dilihat pada Tabel 5.7.

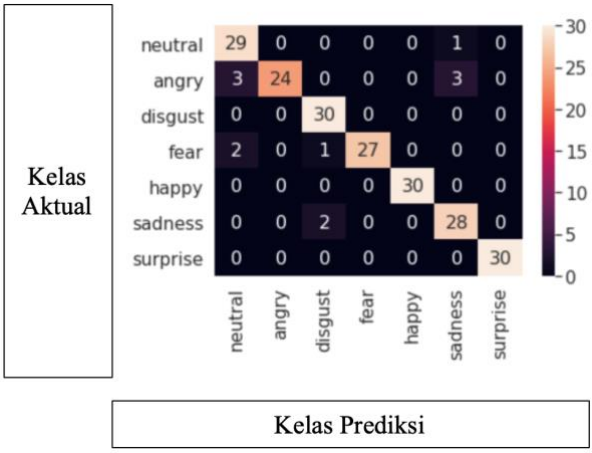
5.3.2 Skenario Uji Coba berdasarkan Dataset

Skenario uji coba berdasarkan dataset bertujuan untuk mengetahui dataset yang memiliki performa terbaik saat diuji dengan model yang sudah dilakukan *training*. Pada skenario uji coba ini, model YOLO diuji coba dengan data *testing* dari dua dataset berbeda yaitu *The Extended Cohn-Kanade AU-Coded Facial Expression Database* (CK+) yang memiliki 210 citra ekspresi wajah dengan 30 citra pada masing-masing kelas ekspresi,

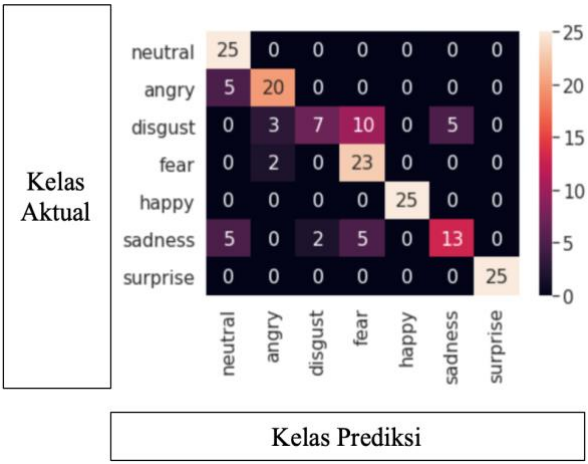
Tabel 5.8 Hasil Uji Coba berdasarkan Dataset

Dataset	Akurasi (%)	Precision (%)	Recall (%)	F1-Score (%)	Waktu Deteksi (detik)
CK+	94	95	94	94	22,74
IMED	79	80	79	77	18,5
Data Video	31	36	31	32	22,4

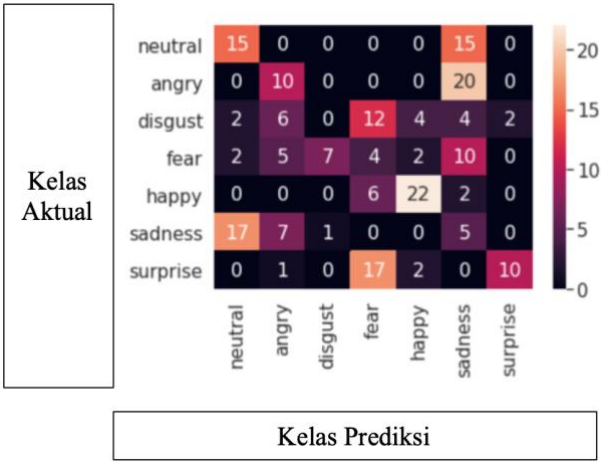
Tabel 5.9 *Confusion Matrix* Dataset CK+



Tabel 5.11 *Confusion Matrix* Dataset IMED



Tabel 5.10 *Confusion Matrix* Data Video



The Indonesian Mixed Emotion Dataset (IMED) yang memiliki 175 citra ekspresi wajah dengan 25 citra pada masing-masing kelas

Tabel 5.12 Spesifikasi *Hyperparameter* Inception v2 pada Faster R-CNN

Keterangan	**Spesifikasi**
Ukuran gambar input	416×416
Num_steps (iterasi)	6000
Batch	32
Channel (kanal warna)	3 (RGB)
Learning Rate	0,01
Momentum	0,9

ekspresi dan data video yang memiliki 210 citra ekspresi wajah dengan 30 citra pada masing-masing kelas. Spesifikasi dataset dapat dilihat pada Tabel 5.1. Model yang digunakan untuk pengujian telah dilakukan *training* menggunakan parameter *channel, learning rate, momentum, decay,* dan *anchor box* terbaik dari subbab 5.3.1. Hasil uji coba berdasarkan dataset dapat dilihat pada Tabel 5.8 dengan *confusion matrix* untuk CK+ pada Tabel 5.9, *confusion matrix* untuk IMED pada Tabel 5.11, dan *confusion matrix* untuk data video pada Tabel 5.10.

5.3.3 Skenario Uji Coba Perbandingan Metode

Skenario uji coba perbandingan metode bertujuan untuk membandingkan performa metode YOLO dengan metode deteksi objek lain. Pada skenario uji coba ini, YOLO dibandingkan dengan metode deteksi objek lain yang bernama Faster R-CNN dengan *preset* Inception v2 milik Google dengan spesifikasi konfigurasi

Tabel 5.13 Hasil Uji Coba Perbandingan YOLOv3 dengan Faster R-CNN

Metode	**Akurasi (%)**	**Precision (%)**	**Recall (%)**	**F1-Score (%)**	**Waktu Deteksi (detik)**
YOLOv3	**68**	**70**	**68**	**68**	**63,56**
Faster R-CNN	50	53	50	49	678,66

parameter Faster R-CNN yang dapat dilihat pada Tabel 5.12. Konfigurasi parameter yang digunakan pada Faster R-CNN ini telah disesuaikan agar mendekati dengan parameter terbaik yang digunakan pada YOLO. Skenario ini diuji coba pada data *testing* dengan 595 citra ekspresi wajah. Hasil uji coba perbandingan metode YOLO dengan Faster R-CNN dapat dilihat pada Tabel 5.13.

5.3.4 Skenario Uji Coba pada Data Video

Skenario uji coba pada data video bertujuan untuk menguji performa model YOLO dengan data yang belum pernah dilihat sebelumnya dan diambil dalam kondisi lingkungan yang tidak terkontrol. Uji coba pada data video dilakukan dengan data video dari 8 subjek yang merupakan mahasiswa Teknik Informatika ITS dengan berbagai macam posisi wajah yang telah dideskripsikan pada subbab 3.1.

Evaluasi dilakuan dengan menghitung *majority vote* dari masing-masing video tersebut. Data video diuji dengan menggunakan dua model YOLO yang dilakukan *training* dengan dataset yang berbeda. Model yang pertama dilakukan *training* dengan dataset CK+ dan IMED saja yang hasilnya dapat dilihat pada Tabel 5.14, sedangkan model yang kedua dilakukan *training*

Tabel 5.14 Hasil Uji Coba pada Data Video (Model CK+ dan IMED)

Kelas Ekspresi	Jumlah Subjek diprediksi Benar	Jumlah Subjek
Netral	3	8
Marah	2	8
Jijik	6	8
Takut	0	8
Senang	6	8
Sedih	0	8
Terkejut	4	8
Total	**21**	**56**

Tabel 5.15 Hasil Uji Coba pada Data Video (Model CK+, IMED, Data Video)

Kelas Ekspresi	Jumlah Subjek diprediksi Benar	Jumlah Subjek
Netral	7	8
Marah	6	8
Jijik	5	8
Takut	5	8
Senang	7	8
Sedih	6	8
Terkejut	5	8
Total	**41**	**56**

dengan dataset CK+, IMED, dan data video yang hasilnya dapat dilihat pada Tabel 5.15.

5.4 Pembahasan dan Evaluasi

Hasil pengujian penggantian parameter pada YOLO dengan memvariasikan parameter *channel* atau kanal warna menunjukkan bahwa YOLO dengan 3 *channel* mendapatkan hasil akurasi sebesar 68% yang lebih baik dibandingkan dengan 1 *channel* yang memiliki akurasi 64%. Hal ini dikarenakan karena model yang dilakukan *training* dengan 3 *channel* dapat telah mempelajari fitur dari ketiga *channel* sehingga saat dilakukan pengujian, model dapat melihat ekspresi wajah dengan lebih jelas.

Hasil pengujian penggantian parameter dengan memvariasikan parameter *learning rate* menunjukkan bahwa YOLO dengan *learning rate* 0,01 mendapatkan hasil akurasi sebesar 68% yang lebih baik dibandingkan dengan *learning rate* 0,1 dan 0,001. Hal ini menunjukkan bahwa dengan *learning rate* 0,01, model dapat melakukan koreksi bobot yang optimal pada saat proses *training*.

Parameter *momentum* dan *decay* merupakan parameter *optimizer* yang dapat mempercepat konvergensi saat *training* jika diatur dengan nilai yang tepat, sedangkan parameter *anchor box*

(a) (b)

Gambar 5.1 Contoh Ekspresi Wajah yang Mirip pada Data Video

mendefinisikan *bounding box* awal yang digunakan saat *testing*. Hasil pengujian penggantian parameter dengan memvariasikan parameter *momentum*, *decay*, dan *anchor box* menunjukkan bahwa YOLO dengan parameter awal *momentum 0,90*, *decay 0,0005*, dan *anchor box default* mempunyai hasil yang lebih baik dibandingkan jika ketiga parameter tersebut diubah.

Hasil uji coba berdasarkan dataset menunjukkan bahwa dataset CK+ memiliki akurasi yang lebih tinggi dibandingkan dengan dataset IMED dan data video dengan akurasi 94%. Hal ini disebabkan karena data video diambil dalam lingkungan yang tidak terkontrol dan berbeda-beda pada setiap videonya, tidak seperti dataset yang diambil dalam lingkungan yang tekontrol seperti dalam hal pencahayaan, jarak wajah dengan kamera, dan kondisi

(a) (b) (c)

Gambar 5.2 Contoh Perbandingan Kelas Terkejut pada Dataset (a) CK+, (b) IMED, dan (c) Data Video

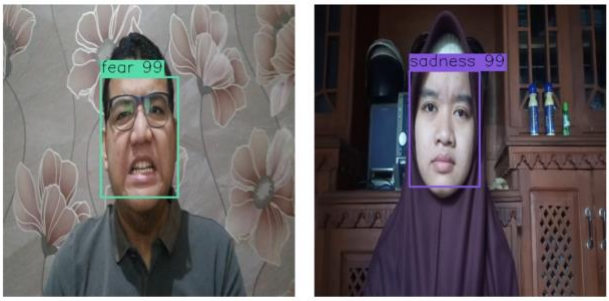
(a) (b)

Gambar 5.3 Contoh Salah Deteksi pada Dataset CK+

background. Data video juga memiliki posisi wajah yang berbeda dan ekspresi wajah satu dengan ekspresi wajah yang lainnya memiliki kemiripan seperti pada Gambar 5.1. Gambar 5.1(a) merupakan kelas jijik yang diprediksi sebagai kelas senang, sedangkan Gambar 5.1(b) merupakan kelas senang yang benar diprediksi sebagai kelas senang. Selain itu, terdapat subjek yang kurang dapat mengekspresikan dirinya, seperti pada Gambar 5.2(c). Sebagai contoh kelas terkejut yang umumnya mempunyai karakteristik mata terbelalak dengan mulut yang terbuka lebar seperti Gambar 5.2(a) dan Gambar 5.2(b).

(a) (b)

Gambar 5.4 Contoh Salah Deteksi pada Dataset IMED

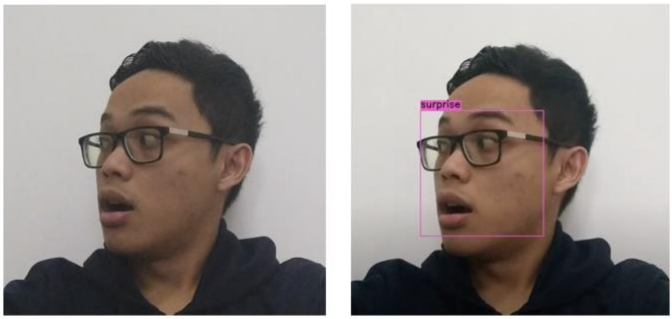


(a) (b)

Gambar 5.5 Contoh Salah Deteksi pada Data Video

Berdasarkan tabel *confusion matrix* pada dataset CK+ yang dapat dilihat pada Tabel 5.9, kesalahan klasifikasi paling banyak terjadi pada kelas marah (*angry*) dan takut (*fear*) yang diprediksi sebagai kelas netral (*neutral*). Ekspresi wajah marah pada Gambar 5.3(a) dan ekspresi wajah takut pada Gambar 5.3(b) diprediksi sebagai kelas netral. Berdasarkan tabel *confusion matrix* pada dataset IMED yang dapat dilihat pada Tabel 5.11, kesalahan klasifikasi paling banyak terjadi pada kelas jijik (*disgust*) dan sedih (*sadness*). Ekspresi wajah jijik pada Gambar 5.4(a) diprediksi sebagai kelas takut dan pada Gambar 5.4(b) ekspresi wajah sedih diprediksi sebagai kelas takut (*fear*). Sedangkan berdasarkan *confusion matrix* pada data video yang dapat dilihat pada Tabel 5.10, kesalahan klasifikasi paling banyak terjadi pada kelas jijik dan takut. Ekspresi wajah jijik pada Gambar 5.5(a) diprediksi sebagai kelas takut, sedangkan kelas takut pada Gambar 5.5(b) diprediksi sebagai kelas sedih.

Hasil perbandingan antara YOLO dan Faster R-CNN menunjukkan bahwa YOLO memiliki akurasi yang lebih tinggi yaitu 68% dan waktu deteksi yang lebih cepat yaitu 63,56 detik pada data *testing* dengan 595 citra ekspresi wajah. Hal ini dikarenakan karena Faster R-CNN membuat *feature map* dengan CNN terlebih dahulu, kemudian mencari *Region of Interest* (ROI) dan setiap ROI dilakukan klasifikasi dengan *Region Proposal*



(a) (b)

Gambar 5.6 Contoh Subjek 3 Kelas Terkejut Menoleh ke Kanan

Network (RPN), sedangkan YOLO hanya melakukan sekali CNN untuk mendeteksi dan mengklasifikasi objek.

Hasil pengujian pengenalan ekspresi wajah dari data video dengan YOLO dan menggunakan *majority vote* menunjukkan bahwa 41 video diprediksi dengan benar dari total 56 video dengan menggunakan model yang dilakukan *training* dengan dataset CK+, IMED dan data video dan lebih baik hampir dua kali lipat dibandingkan model yang dilakukan *training* dengan dataset CK+ dan IMED saja yang menghasilkan 21 video diklasifikasi dengan benar. Perbandingan hasil model yang dilakukan *training* dengan dataset CK+ dan IMED saja dan model yang dilakukan *training* dengan CK+, IMED, dan data video dapat dilihat pada Gambar 5.6. Gambar 5.6 menunjukkan bahwa model yang dilakukan *training* tanpa menggunakan data video tidak dapat mendeteksi wajah seperti pada Gambar 5.6(a) sedangkan model yang dilakukan *training* dengan tambahan data video dapat mendeteksi wajah dan mengklasifikasi ekspresi wajah seperti pada Gambar 5.6(b).

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 merupakan 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. Proses *preprocessing* pada data CK+, IMED, dan data video telah berhasil dilakukan dengan memisahkan dataset menjadi data *training* dan data *testing*, dan melakukan *resize* menjadi 416×416 piksel. Pada data *training*, *bounding box ground truth* pada daerah wajah dibuat untuk setiap citra.
2. Arsitektur YOLO telah berhasil diimplementasikan dengan menggunakan darknet untuk proses *training* dan TensorFlow untuk proses *testing*. Model YOLO yang dibangun menghasilkan akurasi yang paling baik yaitu sebesar 68% dan waktu deteksi 63,56 detik dengan *channel* 3, *learning rate* 0,01, *momentum* 0,90, *decay* 0,0005, *dan anchor box default*.
3. Berdasarkan uji coba parameter pada arsitektur YOLO yang digunakan, model yang dibangun menghasilkan akurasi yang paling baik yaitu sebesar 68% dan waktu deteksi 63,56 detik dengan *channel* 3, *learning rate* 0,01, *momentum* 0,90, *decay* 0,0005, *dan anchor box default*.
4. Pada uji coba berdasarkan dataset yang diuji coba pada dataset CK+, dataset IMED, dan data video, dataset CK+ menghasilkan akurasi yang paling baik sebesar 94%.
5. Berdasarkan uji coba perbandingan metode YOLO dengan Faster R-CNN, model YOLO menghasilkan akurasi paling

baik yaitu sebesar 68% dan waktu deteksi yang paling singkat 63,56 detik.

6. Berdasarkan uji coba pada data video dengan arsitektur YOLO yang digunakan, model dapat memprediksi ekspresi wajah pada 41 video dengan benar dari 56 video.

6.2 Saran

Saran yang diberikan untuk pengembangan pengenalan ekspresi wajah menggunakan YOLO, yaitu dataset harus memiliki ekspresi wajah yang jelas dan dapat dibedakan antara satu ekspresi wajah dengan ekspresi wajah yang lain.

DAFTAR PUSTAKA

[1] G. Luh, H. Wu, Y. Yong, Y. Lai, and Y. Chen, "Facial Expression Based Emotion Recognition Employing YOLOv3 Deep Neural Networks," in *2019 International Conference on Machine Learning and Cybernetics (ICMLC)*, 2019, pp. 1–7, doi: 10.1109/ICMLC48188.2019.8949236.

[2] N. B. Kar, K. S. Babu, A. K. Sangaiah, and S. Bakshi, "Face expression recognition system based on ripplet transform type II and least square SVM," *Multimed. Tools Appl.*, vol. 78, no. 4, pp. 4789–4812, 2019, doi: 10.1007/s11042-017-5485-0.

[3] N. K. Benamara *et al.*, "Real-Time Emotional Recognition for Sociable Robotics Based on Deep Neural Networks Ensemble," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11486 LNCS, 2019, pp. 171–180.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, vol. 27, no. 3, pp. 779–788, doi: 10.1109/CVPR.2016.91.

[5] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, "The extended Cohn-Kanade dataset (CK+): A complete dataset for action unit and emotion-specified expression," *2010 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. - Work. CVPRW 2010*, no. July, pp. 94–101, 2010, doi: 10.1109/CVPRW.2010.5543262.

[6] D. Y. Liliana, T. Basaruddin, and I. I. D. Oriza, "The Indonesian Mixed Emotion Dataset (IMED): A facial

expression dataset for mixed emotion recognition," *ACM Int. Conf. Proceeding Ser.*, pp. 56–60, 2018, doi: 10.1145/3293663.3293671.

[7] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights Imaging*, vol. 9, no. 4, pp. 611–629, 2018, doi: 10.1007/s13244-018-0639-9.

[8] Z. Li, W. Yang, S. Peng, and F. Liu, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," Apr. 2020.

[9] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 56, pp. 1929–1958, 2014.

[10] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 6517–6525, 2017, doi: 10.1109/CVPR.2017.690.

[11] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," Apr. 2018.

[12] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression."

[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks."

[14] A. Tharwat, "Classification assessment methods," *Appl. Comput. Informatics*, Aug. 2018, doi: 10.1016/j.aci.2018.08.003.

[15] "About Python™ | Python.org." [Online]. Available: https://www.python.org/about/. [Accessed: 22-Jun-2020].

[16] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, “Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks,” *IEEE Signal Process. Lett.*, vol. 23, no. 10, pp. 1499–1503, 2016, doi: 10.1109/LSP.2016.2603342.

[17] M. Abadi *et al.*, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems,” *Netw. Comput. Neural Syst.*, vol. 16, no. 2–3, pp. 121–138, Mar. 2016, doi: 10.1080/09548980500300507.

[18] M. Naveenkumar, “OpenCV for Computer Vision Applications,” no. March 2015, pp. 52–56, 2016.

[19] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, “The NumPy array: a structure for efficient numerical computation,” *Comput. Sci. Eng.*, vol. 13, no. 2, pp. 22–30, Feb. 2011, doi: 10.1109/MCSE.2011.37.

[20] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in Python,” *J. Mach. Learn. Res.*, vol. 12, no. 85, pp. 2825–2830, 2011.

[21] J. D. Hunter, “Matplotlib: A 2D graphics environment,” *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 99–104, May 2007, doi: 10.1109/MCSE.2007.55.

[22] M. Peemen, B. Mesman, and H. Corporaal, “Efficiency optimization of trainable feature extractors for a consumer platform,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6915 LNCS, 2011, pp. 293–304.

[23] A. Kathuria, “What’s new in YOLO v3? – Towards Data Science,” *Medium*, 2018. [Online]. Available: https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b. [Accessed: 22-Jun-2020].

[24] Amro Kamal, “YOLO, YOLOv2 and YOLOv3: All You

want to know - Amro Kamal - Medium." [Online]. Available: https://medium.com/@amrokamal_47691/yolo-yolov2-and-yolov3-all-you-want-to-know-7e3e92dc4899. [Accessed: 22-Jun-2020].

[25] Sambasivarao. K, "Non-maximum Suppression (NMS) - Towards Data Science." [Online]. Available: https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c. [Accessed: 22-Jun-2020].

LAMPIRAN

L.1 Hasil Uji Coba YOLO dengan Channel 1, Learning Rate 0,01, Momentum 0,90, Decay 0,0005, dan Default Anchor Box pada Dataset CK+, IMED, dan Data Video

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	61	73	66
Marah	57	52	54
Jijik	79	39	52
Takut	56	68	62
Senang	79	91	85
Sedih	44	61	51
Terkejut	98	66	79
Rata-Rata	**68**	**64**	**64**

L.2 Hasil Uji Coba YOLO dengan Channel 1, Learning Rate 0,1, Momentum 0,90, Decay 0,0005, dan Default Anchor Box pada Dataset CK+

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	83	100	91
Marah	100	80	89
Jijik	100	100	100
Takut	100	100	100
Senang	100	100	100
Sedih	100	100	100
Terkejut	100	100	100
Rata-Rata	**98**	**97**	**97**

L.3 Hasil Uji Coba YOLO dengan Channel 1, Learning Rate 0,01, Momentum 0,90, Decay 0,0005, dan Default Anchor Box pada Dataset IMED

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	73	96	83
Marah	55	48	51
Jijik	33	4	7
Takut	56	96	71
Senang	100	100	100
Sedih	62	60	61
Terkejut	100	100	100
Rata-Rata	**68**	**72**	**68**

L.4 Hasil Uji Coba YOLO dengan Channel 1, Learning Rate 0,01, Momentum 0,90, Decay 0,0005, dan Default Anchor Box pada Data Video

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	24	27	25
Marah	26	27	26
Jijik	22	7	10
Takut	13	13	13
Senang	52	73	61
Sedih	11	23	15
Terkejut	50	3	6
Rata-Rata	**28**	**25**	**23**

L.5 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,01, Momentum 0,90, Decay 0,0005, dan Default Anchor Box pada Dataset CK+, IMED, dan Data Video

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	66	81	73
Marah	69	64	66
Jijik	74	44	55
Takut	52	64	57
Senang	91	91	91
Sedih	43	54	48
Terkejut	97	76	86
Rata-Rata	**70**	**68**	**68**

L.6 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,01, Momentum 0,90, Decay 0,0005, dan Default Anchor Box pada Dataset CK+

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	85	97	91
Marah	100	80	89
Jijik	91	100	95
Takut	100	90	95
Senang	100	100	100
Sedih	88	93	90
Terkejut	100	100	100
Rata-Rata	**95**	**94**	**94**

L.7 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,01, Momentum 0,90, Decay 0,0005, dan Default Anchor Box pada Dataset IMED

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	71	100	83
Marah	80	80	80
Jijik	78	28	41
Takut	61	92	73
Senang	100	100	100
Sedih	72	52	60
Terkejut	100	100	100
Rata-Rata	**80**	**79**	**77**

L.8 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,01, Momentum 0,90, Decay 0,0005, dan Default Anchor Box pada Data Video

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	42	50	45
Marah	34	33	34
Jijik	0	0	0
Takut	10	13	12
Senang	73	73	73
Sedih	9	17	12
Terkejut	83	33	48
Rata-Rata	**36**	**31**	**32**

L.9 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,1, Momentum 0,90, Decay 0,0005, dan Default Anchor Box pada Dataset CK+, IMED, dan Data Video

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	60	59	59
Marah	64	56	60
Jijik	83	45	58
Takut	57	69	63
Senang	87	85	86
Sedih	43	61	50
Terkejut	82	79	80
Rata-Rata	**68**	**65**	**65**

L.10 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,1, Momentum 0,90, Decay 0,0005, dan Default Anchor Box pada Dataset CK+

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	90	90	90
Marah	100	77	87
Jijik	100	100	100
Takut	100	100	100
Senang	100	100	100
Sedih	81	100	90
Terkejut	100	100	100
Rata-Rata	**96**	**95**	**95**

L.11 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,1, Momentum 0,90, Decay 0,0005, dan Default Anchor Box pada Dataset IMED

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	68	84	75
Marah	100	60	75
Jijik	80	32	46
Takut	56	100	71
Senang	100	96	98
Sedih	60	60	60
Terkejut	100	100	100
Rata-Rata	**80**	**76**	**75**

L.12 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,1, Momentum 0,90, Decay 0,0005, dan Default Anchor Box pada Data Video

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	9	7	8
Marah	27	33	30
Jijik	0	0	0
Takut	14	13	14
Senang	62	60	61
Sedih	12	23	16
Terkejut	44	40	42
Rata-Rata	**24**	**25**	**24**

L.13 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,001, Momentum 0,90, Decay 0,0005, dan Default Anchor Box pada Dataset CK+, IMED, dan Data Video

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	69	80	74
Marah	60	61	61
Jijik	74	40	52
Takut	43	61	51
Senang	97	82	89
Sedih	44	51	47
Terkejut	88	78	82
Rata-Rata	**68**	**65**	**65**

L.14 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,001, Momentum 0,90, Decay 0,0005, dan Default Anchor Box pada Dataset CK+

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	93	90	92
Marah	93	83	88
Jijik	100	100	100
Takut	85	97	91
Senang	100	90	95
Sedih	85	93	89
Terkejut	100	100	100
Rata-Rata	**94**	**93**	**93**

L.15 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,001, Momentum 0,90, Decay 0,0005, dan Default Anchor Box pada Dataset IMED

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	81	100	89
Marah	73	96	83
Jijik	80	16	27
Takut	52	88	66
Senang	100	96	98
Sedih	92	44	59
Terkejut	89	100	94
Rata-Rata	**81**	**77**	**74**

L.16 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,001, Momentum 0,90, Decay 0,0005, dan Default Anchor Box pada Data Video

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	41	53	46
Marah	12	10	11
Jijik	0	0	0
Takut	2	3	3
Senang	90	63	75
Sedih	8	13	10
Terkejut	65	37	47
Rata-Rata	**31**	**26**	**27**

L.17 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,01, Momentum 0,85, Decay 0,0005, dan Default Anchor Box pada Dataset CK+, IMED, dan Data Video

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	67	71	69
Marah	56	60	58
Jijik	77	35	48
Takut	44	51	47
Senang	94	91	92
Sedih	43	56	49
Terkejut	85	84	84
Rata-Rata	**67**	**64**	**64**

L.18 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,01, Momentum 0,85, Decay 0,0005, dan Default Anchor Box pada Dataset CK+

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	97	100	98
Marah	97	93	95
Jijik	100	100	100
Takut	100	93	97
Senang	100	100	100
Sedih	94	100	97
Terkejut	100	100	100
Rata-Rata	**98**	**98**	**98**

L.19 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,01, Momentum 0,85, Decay 0,0005, dan Default Anchor Box pada Dataset IMED

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	65	88	75
Marah	62	64	63
Jijik	0	0	0
Takut	34	56	42
Senang	100	96	98
Sedih	67	40	50
Terkejut	74	100	85
Rata-Rata	**57**	**63**	**59**

L.20 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,01, Momentum 0,85, Decay 0,0005, dan Default Anchor Box pada Data Video

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	33	27	30
Marah	19	23	21
Jijik	0	0	0
Takut	3	3	3
Senang	82	77	79
Sedih	12	27	17
Terkejut	80	53	64
Rata-Rata	**33**	**30**	**31**

L.21 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,01, Momentum 0,95, Decay 0,0005, dan Default Anchor Box pada Dataset CK+, IMED, dan Data Video

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	64	67	66
Marah	60	58	59
Jijik	83	46	59
Takut	49	53	51
Senang	90	92	91
Sedih	43	60	50
Terkejut	83	79	81
Rata-Rata	**67**	**65**	**65**

L.22 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,01, Momentum 0,95, Decay 0,0005, dan Default Anchor Box pada Dataset CK+

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	83	83	83
Marah	100	77	87
Jijik	100	100	100
Takut	100	93	97
Senang	100	100	100
Sedih	77	100	87
Terkejut	100	100	100
Rata-Rata	**94**	**93**	**93**

L.23 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,01, Momentum 0,95, Decay 0,0005, dan Default Anchor Box pada Dataset IMED

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	71	100	83
Marah	73	64	68
Jijik	90	36	51
Takut	52	60	56
Senang	100	96	98
Sedih	70	56	62
Terkejut	71	100	83
Rata-Rata	**75**	**73**	**72**

L.24 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,01, Momentum 0,95, Decay 0,0005, dan Default Anchor Box pada Data Video

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	29	23	26
Marah	27	33	30
Jijik	0	0	0
Takut	6	7	6
Senang	73	80	76
Sedih	12	23	16
Terkejut	75	40	52
Rata-Rata	**32**	**30**	**29**

L.25 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,01, Momentum 0,90, Decay 0,005, dan Default Anchor Box pada Dataset CK+, IMED, dan Data Video

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	68	72	70
Marah	65	75	70
Jijik	87	46	60
Takut	50	53	51
Senang	91	82	86
Sedih	47	58	52
Terkejut	78	81	79
Rata-Rata	**69**	**67**	**67**

L.26 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,01, Momentum 0,90, Decay 0,005, dan Default Anchor Box pada Dataset CK+

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	97	100	98
Marah	100	97	98
Jijik	97	100	98
Takut	100	97	98
Senang	100	100	100
Sedih	97	97	97
Terkejut	100	100	100
Rata-Rata	**99**	**99**	**99**

L.27 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,01, Momentum 0,90, Decay 0,005, dan Default Anchor Box pada Dataset IMED

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	83	100	91
Marah	100	100	100
Jijik	90	36	51
Takut	48	60	54
Senang	100	96	98
Sedih	75	60	67
Terkejut	71	100	83
Rata-Rata	**81**	**79**	**78**

L.28 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,01, Momentum 0,90, Decay 0,005, dan Default Anchor Box pada Data Video

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	21	20	20
Marah	22	33	27
Jijik	0	0	0
Takut	3	3	3
Senang	70	53	60
Sedih	9	17	12
Terkejut	58	47	52
Rata-Rata	**26**	**25**	**25**

L.29 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,01, Momentum 0,90, Decay 0,00005, dan Default Anchor Box pada Dataset CK+, IMED, dan Data Video

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	70	73	71
Marah	62	65	64
Jijik	82	42	56
Takut	43	56	49
Senang	100	88	94
Sedih	43	58	49
Terkejut	90	78	84
Rata-Rata	**70**	**66**	**67**

L.30 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,01, Momentum 0,90, Decay 0,00005, dan Default Anchor Box pada Dataset CK+

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	88	100	94
Marah	96	77	85
Jijik	100	100	100
Takut	90	93	92
Senang	100	100	100
Sedih	87	90	89
Terkejut	100	100	100
Rata-Rata	**94**	**94**	**94**

L.31 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,01, Momentum 0,90, Decay 0,00005, dan Default Anchor Box pada Dataset IMED

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	73	88	80
Marah	96	88	92
Jijik	75	24	36
Takut	49	80	61
Senang	100	92	96
Sedih	75	60	67
Terkejut	83	100	91
Rata-Rata	**79**	**76**	**75**

L.32 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,01, Momentum 0,90, Decay 0,00005, dan Default Anchor Box pada Data Video

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	40	33	36
Marah	24	33	28
Jijik	0	0	0
Takut	0	0	0
Senang	100	73	85
Sedih	11	23	15
Terkejut	85	37	51
Rata-Rata	**37**	**29**	**31**

L.33 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,01, Momentum 0,90, Decay 0,0005, dan Custom Anchor Box pada Dataset CK+, IMED, dan Data Video

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	63	79	70
Marah	59	64	61
Jijik	83	41	55
Takut	50	53	51
Senang	97	92	95
Sedih	47	55	51
Terkejut	79	81	80
Rata-Rata	**69**	**66**	**66**

L.34 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,01, Momentum 0,90, Decay 0,0005, dan Custom Anchor Box pada Dataset CK+

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	84	90	87
Marah	96	80	87
Jijik	97	100	98
Takut	100	90	95
Senang	100	100	100
Sedih	86	100	92
Terkejut	100	100	100
Rata-Rata	**95**	**94**	**94**

L.35 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,01, Momentum 0,90, Decay 0,0005, dan Custom Anchor Box pada Dataset IMED

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	70	92	79
Marah	71	80	75
Jijik	100	20	33
Takut	44	64	52
Senang	100	100	100
Sedih	94	60	73
Terkejut	78	100	88
Rata-Rata	**80**	**74**	**72**

L.36 Hasil Uji Coba YOLO dengan Channel 3, Learning Rate 0,01, Momentum 0,90, Decay 0,0005, dan Custom Anchor Box pada Data Video

Kelas	Precision (%)	Recall (%)	F1-Score (%)
Netral	41	57	48
Marah	26	33	29
Jijik	0	0	0
Takut	7	7	7
Senang	92	77	84
Sedih	4	7	5
Terkejut	56	47	51
Rata-Rata	**32**	**32**	**32**

L.37 Hasil Majority Vote pada Data Video dengan Model YOLO yang dilakukan Training pada Dataset CK+, IMED, dan Data Video

Subjek	Kelas Aktual	Kelas Prediksi	Jumlah Frame Tidak Terdeteksi Wajah
Subjek 1	marah	sedih	0
	jijik	jijik	0
	takut	takut	0
	senang	senang	0
	netral	netral	0
	sedih	sedih	0
	terkejut	terkejut	0
Subjek 2	marah	marah	0
	jijik	takut	0
	takut	takut	0
	senang	senang	0
	netral	netral	0
	sedih	sedih	0
	terkejut	terkejut	0
Subjek 3	marah	marah	0
	jijik	jijik	0
	takut	takut	0
	senang	takut	0
	netral	netral	0
	sedih	netral	0

Subjek	Kelas Aktual	Kelas Prediksi	Jumlah Frame Tidak Terdeteksi Wajah
Subjek 3	terkejut	terkejut	0
Subjek 4	marah	marah	0
	jijik	jijik	0
	takut	takut	0
	senang	senang	0
	netral	netral	0
	sedih	sedih	0
	terkejut	takut	0
Subjek 5	marah	marah	0
	jijik	jijik	0
	takut	takut	0
	senang	senang	0
	netral	netral	0
	sedih	marah	0
	terkejut	takut	0
Subjek 6	marah	marah	0
	jijik	jijik	0
	takut	jijik	0
	senang	senang	0
	netral	netral	0
	sedih	sedih	0
	terkejut	senang	0
Subjek 7	marah	marah	0
	jijik	netral	0

Subjek	Kelas Aktual	Kelas Prediksi	Jumlah Frame Tidak Terdeteksi Wajah
Subjek 7	takut	netral	0
	senang	senang	0
	netral	netral	0
	sedih	sedih	0
	terkejut	terkejut	0
Subjek 8	marah	sedih	0
	jijik	takut	0
	takut	sedih	0
	senang	senang	0
	netral	sedih	0
	sedih	sedih	0
	terkejut	terkejut	0

L.38 Hasil Majority Vote pada Data Video dengan Model YOLO yang dilakukan Training pada Dataset CK+ dan IMED

Subjek	Kelas Aktual	Kelas Prediksi	Jumlah Frame Tidak Terdeteksi Wajah
Subjek 1	marah	marah	55
	jijik	sedih	115
	takut	sedih	63
	senang	senang	106
	netral	sedih	117
	sedih	marah	43

Subjek	Kelas Aktual	Kelas Prediksi	Jumlah Frame Tidak Terdeteksi Wajah
Subjek 1	terkejut	terkejut	70
Subjek 2	marah	jijik	0
	jijik	jijik	19
	takut	jijik	0
	senang	jijik	11
	netral	jijik	0
	sedih	jijik	35
	terkejut	terkejut	1
Subjek 3	marah	marah	57
	jijik	jijik	46
	takut	jijik	32
	senang	senang	41
	netral	jijik	56
	sedih	jijik	51
	terkejut	terkejut	33
Subjek 4	marah	jijik	24
	jijik	jijik	29
	takut	jijik	44
	senang	jijik	11
	netral	sedih	16
	sedih	jijik	32
	terkejut	jijik	42
Subjek 5	marah	jijik	207
	jijik	jijik	238

Subjek	Kelas Aktual	Kelas Prediksi	Jumlah Frame Tidak Terdeteksi Wajah
Subjek 5	takut	senang	222
	senang	senang	167
	netral	jijik	99
	sedih	jijik	228
	terkejut	terkejut	187
Subjek 6	marah	netral	28
	jijik	jijik	25
	takut	jijik	23
	senang	senang	4
	netral	netral	8
	sedih	netral	25
	terkejut	netral	20
Subjek 7	marah	jijik	78
	jijik	jijik	61
	takut	jijik	45
	senang	senang	33
	netral	netral	26
	sedih	jijik	79
	terkejut	jijik	46
Subjek 8	marah	netral	29
	jijik	senang	26
	takut	netral	33
	senang	senang	50
	netral	netral	44

Subjek	Kelas Aktual	Kelas Prediksi	Jumlah Frame Tidak Terdeteksi Wajah
Subjek 8	sedih	netral	26
	terkejut	netral	35

BIODATA PENULIS

Ayas Faikar Nafis, lahir di Semarang pada tanggal 28 November 1998. Penulis menempuh pendidikan mulai dari TK Aisiyah BA 14 Surabaya (2002 – 2004), SDN Muhammadiyah 22 Surabaya (2004 – 2010), SMP Negeri 16 Surabaya (2010 – 2013), SMA Negeri 15 Surabaya (2013 – 2016), dan sekarang sedang menjalani pendidikan S1 Teknik Informatika di ITS. Penulis aktif dalam organisasi Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) dan Schematics. Di antaranya adalah menjadi *staff* Media dan Informasi HMTC, *staff* Departemen *National Logic Competition* Schematics ITS 2017, dan *staff* ahli Departemen *National Logic Competition* Schematics ITS 2018. Komunikasi dengan penulis dapat melalui telepon: +6282244748000 dan *email*: **ayasfn@gmail.com**.