TUGAS AKHIR - IF184802

Pengenalan Ekspresi Wajah pada Sistem Monitoring Perkuliahan Menggunakan Metode Deep Learning

Marde Fasma Ul'aza
NRP 05111640000046

Dosen Pembimbing I
Dr. Eng. NANIK SUCIATI, S.Kom., M.Kom.

Dosen Pembimbing II
SHINTAMI CHUSNUL HIDAYATI, S.Kom., M.Sc., Ph.D

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

TUGAS AKHIR – IF184802

Pengenalan Ekspresi Wajah pada Sistem Monitoring Perkuliahan Menggunakan Metode Deep Learning

Marde Fasma Ul'aza
NRP 05111640000046

Dosen Pembimbing I
Dr. Eng. NANIK SUCIATI, S.Kom., M.Kom.

Dosen Pembimbing II
SHINTAMI CHUSNUL HIDAYATI, S.Kom., M.Sc., Ph.D.

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 in the Lecture Monitoring System Using the Deep Learning Method

Marde Fasma Ul'aza
NRP 05111640000046

First Advisor
Dr. Eng. NANIK SUCIATI, S.Kom., M.Kom.

Second Advisor
SHINTAMI CHUSNUL HIDAYATI, S.Kom., M.Sc., Ph.D.

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 Sistem Monitoring Perkuliahan Menggunakan Metode Deep Learning

TUGAS AKHIR

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

Oleh:
Marde Fasma Ul'aza
NRP: 05111640000046

Disetujui oleh Pembimbing Tugas Akhir:

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

2. SHINTAMI CHUSNUL HIDAYATI, S.Kom., M.Sc., Ph.D. ..... (Pembimbing 2)
   (NID. 0009018705)

SURABAYA
JUNI, 2020

(Halaman ini sengaja dikosongkan)

Pengenalan Ekspresi Wajah pada Sistem Monitoring Perkuliahan Menggunakan Metode Deep Learning

Nama : Marde Fasma Ul'aza
NRP : 05111640000046
Departemen : Teknik Informatika, FTEIC-ITS
Pembimbing I : Dr. Eng. Nanik Suciati, S.Kom., M.Kom.
Pembimbing II : Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D.

ABSTRAK

Ekspresi wajah merupakan salah satu bentuk komunikasi manusia berupa non-verbal. Manusia dapat mengenali ekspresi wajah dengan mudah, berbeda dengan komputer yang sulit mengenali ekspresi wajah. Dengan perkembangan teknologi sekarang, pengenalan ekspresi wajah sangat diperlukan dalam memenuhi kebutuhan dibeberapa bidang.

Dalam tugas akhir ini, dilakukan pengembangan pengenalan ekspresi wajah manusia dengan menggunakan metode deep learning yang berbasis Convolutional Neural Network (CNN). Dataset yang digunakan adalah dataset yang diolah secara mandiri dari data rekaman pekuliahan mahasiswa S2 Tahun Ajaran 2019/2020 dan dataset ekspresi wajah CK+. Pengujian dilakukan dengan menggunakan data testing yang akan menghasilkan hasil berupa bounding box, prediksi kelas ekspresi wajah dan confidence score.

Hasil dari tugas akhir ini menunjukkan akurasi pada data diolah secara mandiri untuk pengenalan ekspresi wajah paling baik menggunakan arsitektur VGG16 dengan menggunakan algoritma optimasi Adam dan learning rate sebesar 0,0001. Dan pada data CK+ untuk pengenalan ekspresi wajah paling baik menggunakan arsitektur VGG16 dengan menggunakan algoritma optimasi Adam dan learning rate sebesar 0,0001.

Berdasarkan evaluasi terhadap deteksi wajah dengan menggunakan IoU threshold, model yang dibangun menghasilkan precision sebesar 98,83%, recall sebesar 99,88%, F1-score

sebesar 99,35%, dan Average Precision sebesar 99,72% yaitu pada nilai IoU threshold sebesar 50%.

Berdasarkan evaluasi terhadap waktu proses sistem, didapatkan rata-rata waktu proses deteksi wajah selama ± 300 milidetik, rata-rata waktu proses pengenalan ekspresi wajah selama ± 8 milidetik, dan rata-rata waktu proses pelacakan wajah selama ± 10 milidetik.

Kata kunci: ***Pengenalan Ekspresi Wajah, Deteksi Wajah, CNN, VGGFace***

Facial Expressions Recognitions in the Lecture Monitoring System Using the Deep Learning Method

Student's Name : Marde Fasma Ul'aza
Student's ID : 05111640000046
Department : Informatics, Faculty of ELECTICS-ITS
First Advisor : Dr. Eng. Nanik Suciati, S.Kom., M.Kom.
Second Advisor : Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D.

ABSTRACT

Facial expression in humans is a form of non-verbal. Humans can recognize facial expressions easily, in contrast to computers that are difficult to recognize facial expressions. With the development of technology, facial expressions recognition is very necessary in several fields.

In this final project, the development of human facial expression recognition using a deep learning on Convolutional Neural Network (CNN) is carried out. The dataset used is a dataset that is independently processed from the Computer Vision lecture data recording for Academic Year 2019/2020 and the CK+ facial expression dataset.Testing is done using data testing that will produce results in the form of bounding boxes, prediction of facial expression classes and confidence scores.

The results of this final project indicate the accuracy of the data independently processed for the best facial expression recognition using the VGG16 architecture using Adam's optimization algorithm and learning rate of 0,0001. And the CK + data for facial expression recognition is best to use VGG16 architecture using Adam's optimization algorithm and learning rate of 0,0001.

Based on the evaluation of face detection using the IoU threshold, the model built produces a precision of 98.83%, a recall of 99.88%, an F1-score of 99.35%, and an Average Precision of 99.72% ie the IoU threshold value by 50%.

Based on the evaluation of the system processing time, the average face detection time is ± 300 milliseconds, the average facial expression recognition time is ± 8 milliseconds, and the average face tracking time is ± 10 milliseconds.

Keywords: ***Facial Expressions Recognition, Facial Detections, CNN, VGGFace***

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 Sistem Monitoring Perkuliahan Menggunakan Metode Deep Learning**"

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

1. Orang tua dan keluarga penulis, yang telah memberikan dukungan doa, moral, dan material kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir ini.
2. Dr. Eng. Nanik Suciati, S.Kom., M.Kom. dan Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D.. selaku pembimbing I dan II yang telah membimbing dan memberikan motivasi, nasihat dan bimbingan dalam menyelesaikan Tugas Akhir ini.
3. Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. selaku Ketua Departemen Teknik Informatika ITS dan seluruh dosen dan karyawan Departemen Teknik Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa kuliah di Teknik Informatika ITS.
4. Muhajir Bin Abdul Latif, Rasyid Fajar, dan Aldinata Rizky yang membantu dan menemani penulis selama perkuliahan semester akhir dan pengerjaan Tugas Akhir ini.
5. Admin-admin Laboratorium *Mobile Innovation Studio* (MIS) Fino, Zevi, Yuda, Andre, Naja, Shania, Anggar, Hisam, Syubban, Kana, Ella yang memberikan

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

6. Seluruh rekan D'Joko, Kontrakan Barokah, Arya Fajar Production yang telah menjadi sahabat penulis selamanya.
7. Dinas Pendidikan Kabupaten Ponorogo yang memberikan kesempatan penulis untuk fokus mengerjakan Tugas Akhir ini dan menyediakan tempat di Gedung Terpadu.
8. Seluruh mahasiswa *user* TA Teknik Informatika ITS angkatan 2016 yang telah menjadi teman penulis selama pengerjaan Tugas Akhir ini.
9. Seluruh mahasiswa Teknik Informatika ITS angkatan 2016 yang telah menjadi teman penulis selama menjalani masa kuliah di Teknik Informatika ITS.
10. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

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

Surabaya, Juli 2020

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

DAFTAR KODE SUMBER

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

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. Mempelajari ekspresi wajah seseorang menjadi sangatlah penting, agar mengetahui respon seseorang terhadap suatu hal. Ekspresi wajah manusia baik yang disengaja maupun yang tidak sengaja dapat muncul akibat perasaan atau emosi manusia.

Ekspresi wajah memainkan peran penting dalam interaksi manusia-komputer, pengawasan perilaku manusia, teknik pendidikan, psikologis, robot *sociable* dan lain-lain [1]. Manusia dapat mengenali ekspresi wajah dengan mudah, berbeda dengan komputer yang sulit mengenali ekspresi wajah. Tetapi komputer dapat dilatih untuk mendeteksi pola-pola ekspresi manusia dengan teknik *deep learning* yang merupakan bagian dari *machine learning* [2].

Dengan perkembangan teknologi sekarang, pengenalan ekspresi wajah sangat diperlukan dalam memenuhi kebutuhan dibeberapa bidang. Salah satu bidang yang memerlukan pengenalan ekspresi wajah adalah bidang pendidikan. Perkembangan sistem monitoring pada perkuliahan mulai menggunakan pengenalan wajah sebagai salah satu cara mengenali seseorang. Pada sebuah ruangan dapat dipasang sebuah kamera yang berfungsi untuk mengenali wajah mahasiswa yang sedang mengikuti proses belajar mengajar.

Hasil pengenalan wajah dapat diolah kembali sebagai data untuk melakukan pengenalan ekspresi wajah. Dengan menyatukan hasil dari pengenalan wajah dan pengenalan ekspresi wajah dihasilkan beberapa informasi yang memiliki manfaat tertentu. Sebagai contoh untuk pemanfaatan monitoring, absensi maupun deteksi. Namun dalam mengenali ekspresi wajah terdapat

perbedaan ekspresi antara subjek yang membuat akurasi dalam mengenali ekspresi wajah berkurang.

Untuk meningkatkan akurasi dalam mengenali ekspresi wajah dapat dilakukan teknik augmentasi pada kumpulan ekspresi wajah. Teknik augmentasi adalah sebuah teknik memanipulasi sebuah data tanpa kehilangan inti atau esensi dari data tersebut [3]. Hasil dari kumpulan manipulasi ini akan digunakan sebagai dataset dalam pembentukan model selanjutnya.

Dataset yang terbentuk dari proses augmentasi dapat digunakan sebagai data latih dalam metode *transfer learning*. *Transfer learning* adalah suatu teknik atau metode yang memanfaatkan model yang sudah dilatih terhadap suatu dataset untuk menyelesaikan permasalahan lain yang serupa dengan cara menggunakannya sebagai *starting point*, memodifikasi dan memperbaharui parameternya sehingga sesuai dengan *dataset* yang baru [4].

Pada metode *transfer learning* dibutuhkan sebuah model *pretrained* yang sebelumnya telah dilatih. Dalam pengenalan ekspresi wajah, model *pretrained* yang sesuai dengan kebutuhan pengenalan ekspresi wajah adalah *keras_vggface*. Model ini menyediakan resnet50, senet50 [5] dan vgg16 [6] sebagai arsitektur model. Ketiga arsitektur tersebut dapat menunjang proses pembuatan model pada pengenalan ekspresi wajah.

Pada tugas akhir ini, akan dilakukan pengenalan ekspresi wajah dengan menggunakan teknik augmentasi, model *pretrained keras_vggface* dengan arsitektur resnet50, senet50 atau vgg16, deteksi wajah, pengenalan ekspresi, serta kalkulasi evaluasi yang diharapkan dapat menunjang monitoring perkuliahan. Hal ini tentunya diharapkan dapat meningkatkan efisiensi waktu dan kemudahan dalam mengenali ekspresi wajah. Selain itu, penelitian ini juga dapat menjadi referensi untuk pengembangan maupun optimasi bagi studi yang berkaitan.

1.2 Rumusan Masalah

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

1. Bagaimana mendeteksi area wajah manusia pada sebuah gambar?
2. Bagaimana membangun suatu penggolong berbasis *Deep Learning* untuk mengenali ekspresi wajah manusia?
3. Bagaimana mengevaluasi performa dari sistem yang telah diimplementasikan?

1.3 Batasan Permasalahan

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

1. Implementasi tugas akhir ini menggunakan bahasa pemrograman Python.
2. Ada 2 ekspresi mahasiswa yang akan dikenali yaitu ekspresi negatif, dan ekspresi positif.
3. Menggunakan video monitoring perkuliahan di dalam kelas yang diambil dengan satu kamera.
4. *Dataset* yang digunakan bersumber dari rekam digital kamera CCTV pada pekuliahan mahasiswa S2 Tahun Ajaran 2019/2020 Departemen Teknik Informatika, ITS dan *The Extended Cohn-Kanade Dataset* (CK+).

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah melakukan implementasi metode *Deep Learning* dalam pengenalan ekspresi wajah mahasiswa pada video monitoring perkuliahan.

1.5 Manfaat

Tugas akhir ini diharapkan dapat diterapkan pada sistem yang membutuhkan pengenalan ekspresi wajah serta dapat mempermudah dalam melakukan monitoring.

1.6 Metodologi

Pembuatan Tugas Akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Proposal Tugas Akhir

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

1.6.2 Studi Literatur

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

1.6.3 Implementasi Perangkat Lunak

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

1.6.4 Pengujian dan Evaluasi

Tahap pengujian dan evaluasi dilakukan menggunakan data video yang diambil dari kamera CCTV serta untuk mengetahui hasil dan performa arsitektur yang telah dibangun. Evaluasi dilakukan dengan mengevaluasi metode *Deep Learning* dari segi akurasi, *precision, recall,* serta dengan menggunakan data video

sebagai data *testing* untuk melihat performa model yang telah dibuat.

1.6.5 Penyusunan Buku

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

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan Tugas Akhir adalah sebagai berikut:

Bab I Pendahuluan

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

Bab II Tinjauan Pustaka

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

Bab III Perancangan Sistem

Bab ini berisi pembahasan mengenai perancangan dari metode *Deep Learning* 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.

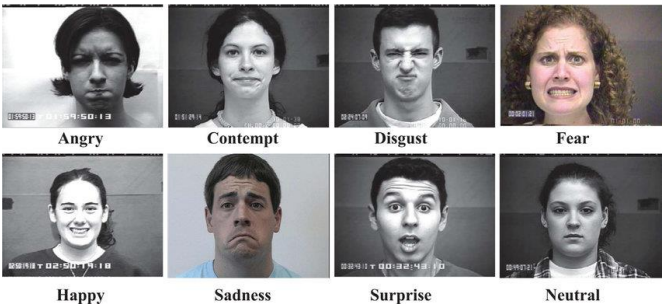
BAB II
TINJAUAN PUSTAKA

Bab ini membahas mengenai teori-teori dasar yang digunakan dalam Tugas Akhir. Teori-teori tersebut adalah *Deep Learning* 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 Pengenalan Ekspresi Wajah

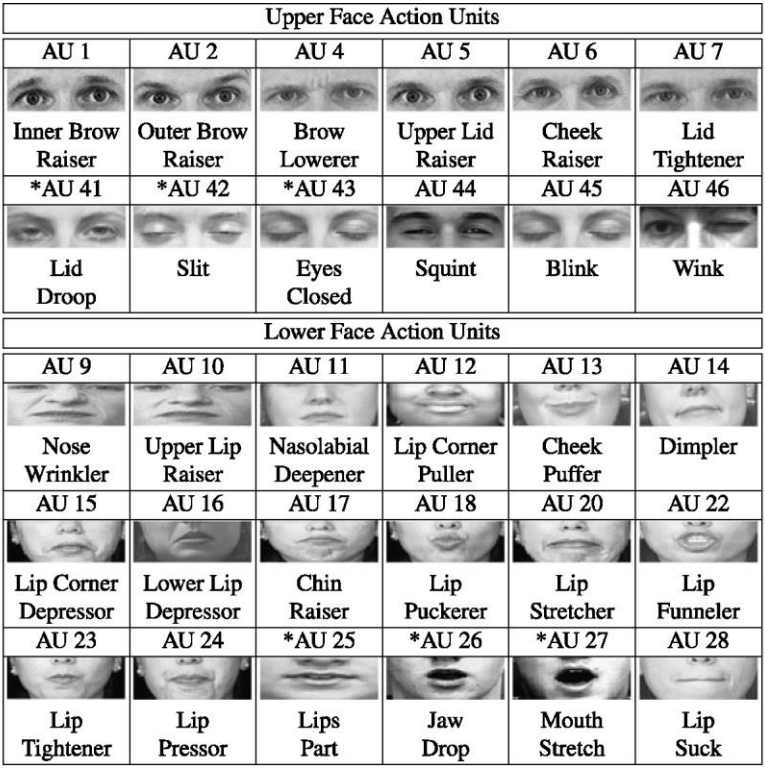
Ekspresi adalah salah satu bentuk komunikasi nonverbal yang merupakan hasil dari satu atau lebih gerakan atau posisi otot pada wajah serta dapat menyampaikan keadaan emosi dari seseorang kepada orang yang mengamatinya. Melalui ekspresi wajah, maka dapat dipahami emosi yang sedang bergejolak pada diri individu.

Ekspresi wajah dapat menyampaikan apa yang sedang terjadi terhadap seseorang. Sebagai contoh, sebuah senyum mengungkap keramahtamahan dan kasih sayang, mengangkat alis mata menunjukkan ekspresi heran, mengernyitkan dahi menunjukkan ketakutan dan kegelisahan. Ilustrasi ekspresi wajah dapat dilihat pada Gambar 2.1.1



Gambar 2.1.1. Ilustrasi Ekspresi Wajah [7].

Pengenalan ekspresi wajah merupakan salah satu cara untuk mengenali ekspresi. Otot-otot yang membentuk ekspresi wajah disebut *musculli facialis* yang merupakan otot-otot penggerak wajah. Ekman dan Friesen mengembangkan sistem yang paling komprehensif untuk sintesa ekspresi wajah berbasis apa yang disebut *Action Units* (AU) [8]. Dengan mendefinisikan sistem pengkodean aksi wajah atau *Facial Action Coding System* (FACS). FACS terdiri atas 46 AU, yang menggambarkan gerakan wajah dasar. Otot-otot wajah menggambarkan secara detail pengaruh masing-masing AU pada ciri wajah. Contoh AU pada ekspresi wajah dapat dilihat pada Gambar 2.1.2



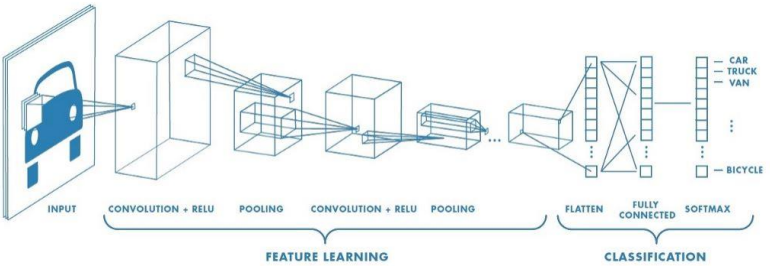
Gambar 2.1.2. *Action Unit* Pada Wajah [9].

2.2 Deep Learning

Deep learning adalah teknik pada *machine learning* dengan banyak layer untuk pengolahan informasi *non-linear* untuk melakukan ekstraksi fitur, pengenalan pola, dan klasifikasi [10]. *Deep learning* didasarkan pada pembelajaran representasional. Implementasi *deep learning* terdiri dari banyak lapisan jaringan syaraf, di mana setiap lapisan mendapat input dari lapisan sebelumnya dan dibagi ke lapisan berikutnya. *Deep learning* dapat digunakan untuk pembelajaran yang bersifat *supervised*, *semi-supervised*, atau bahkan *unsupervised*. Beberapa contoh model *deep learning* antara lain: *Convolutional Neural Network* (CNN), *Recurrent Neural Network* (RNN), *Boltzmann Machine*, dan lain-lain.

2.3 Convolutional Neural Network (CNN)

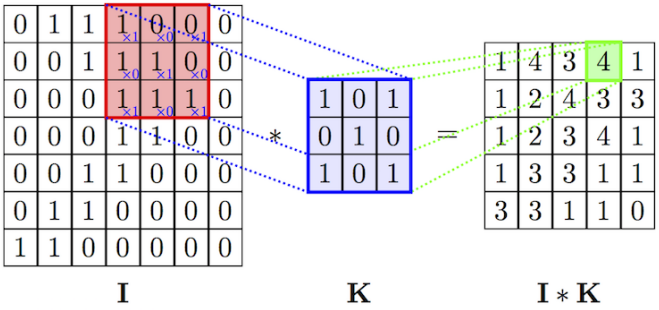
Convolutional Neural Network (CNN) adalah salah satu jenis *deep learning* berbasis *neural network* yang biasa digunakan untuk menganalisis citra visual [2]. CNN biasa digunakan untuk mendeteksi dan mengenali objek. CNN terdiri dari *neuron* yang memiliki *weight*, bias, dan fungsi aktivasi. Arsitektur dari CNN dibagi menjadi 2 bagian besar, yaitu *Feature Extraction Layer* dan *Fully Connected Layer* seperti pada Gambar 2.3.1.



Gambar 2.3.1. Model Arsitektur *CNN* [4].

2.3.1 Convolutional Layer

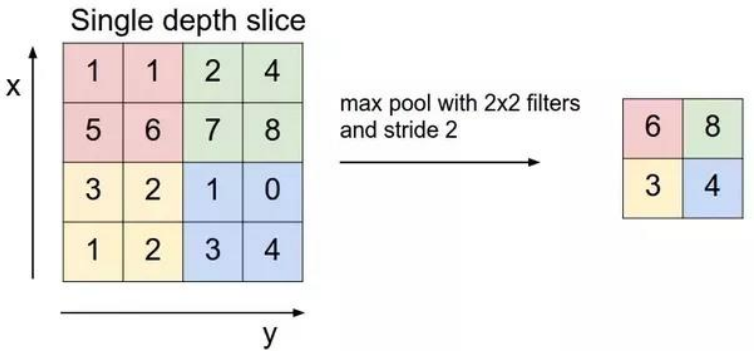
Convolution Layer melakukan operasi konvolusi pada output dari lapisan sebelumnya. Konvolusi adalah istilah matematis yang artinya mengaplikasikan sebuah fungsi pada *output* fungsi lain secara berulang. Tujuan dilakukannya konvolusi pada data citra adalah untuk mengekstrak fitur dari citra masukan [11]. Ilustrasi cara kerja konvolusi bisa dilihat pada Gambar 2.3.2,dimana *I* adalah citra, *K* adalah *filter* atau kernel yang digunakan, *I* * *K* adalah hasil operasi konvolusi.



Gambar 2.3.2. Ilustrasi Lapisan Konvolusi [12].

2.3.2 Pooling Layer

Fungsi dari *Pooling Layer* adalah mereduksi ukuran dari data. Terdapat beberapa tipe *Pooling Layer* diantaranya yaitu *max*, *average*, *sum* dan lainnya. Metode *Pooling* dalam *Convolutional Neural Network* (CNN) yang biasa digunakan adalah *Max Pooling* & *Average Pooling*. *Max Pooling* membagi *output* dari *Convolution Layer* menjadi beberapa matriks kecil lalu mengambil nilai maksimal dari tiap matriks untuk menyusun matriks citra yang telah direduksi, sedangkan *Average Pooling* akan memilih nilai rata-ratanya. Proses tersebut memas-tikan fitur yang didapatkan akan sama meskipun obyek citra mengalami translasi. Ilustrasi cara kerja *Max Pooling* bisa dilihat pada Gambar 2.3.3

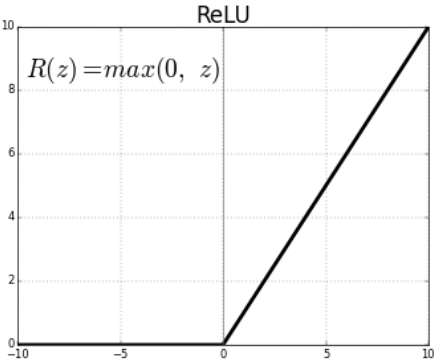


Gambar 2.3.3. Ilustrasi *Max Pooling* [4].

2.3.3 Fully Connected Layer

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

2.3.4 ReLU Activation Function



Gambar 2.3.4. ReLU *Activation Function* [3].

Fungsi aktivasi befungsi untuk menentukan apakah *neuron* tersebut harus aktif atau tidak berdasarkan nilai masukan. Salah satu contoh fungsi aktivasi adalah ReLU (*Rectified Linear Unit*) dimana fungsi ini melakukan *thresholding* dengan nilai nol terhadap nilai masukan, dimana seluruh nilai yang kurang dari nol akan dijadikan nol, seperti pada Gambar 2.3.4.

2.3.5 Softmax Activation Function

Fungsi *softmax* biasa digunakan dalam klasifikasi banyak kelas. *Softmax* memberikan nilai probabilitas untuk setiap label kelas, dimana jumlah seluruh probabilitas adalah 1. *Softmax* pada dasarnya adalah probabilitas eksponensial yang dinormalisasi dari nilai masukan sejumlah kelas pada model klasifikasi seperti pada Persamaan (1.1).

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \tag{1.1}$$

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

2.3.6 Cross Entropy

Loss function merupakan fungsi yang menggambarkan kerugian yang dihasilkan oleh model. *Loss function* dikatakan baik, ketika menghasilkan *error* yang diharapkan paling rendah. Pada permasalahan klasifikasi banyak kelas, *cross entropy* adalah *loss function* yang biasa digunakan. *Cross entropy* akan menghitung *error* antara nilai prediksi *S* dengan nilai sebenarnya *T*, seperti pada Persamaan (1.2). Selanjutnya, nilai *error* akhir diambil dari rata-rata hasil *cross entropy*, seperti pada Persamaan (2.3).

$$D(S_i, T_i) = -\sum_j T_{ij} \log S_{ij} \tag{1.2}$$

$$J(W, b) = \frac{1}{n} \sum_i D(S_i, T_i) \tag{1.3}$$

2.3.7 Adam Optimizer

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

2.3.8 RMSProp Optimizer

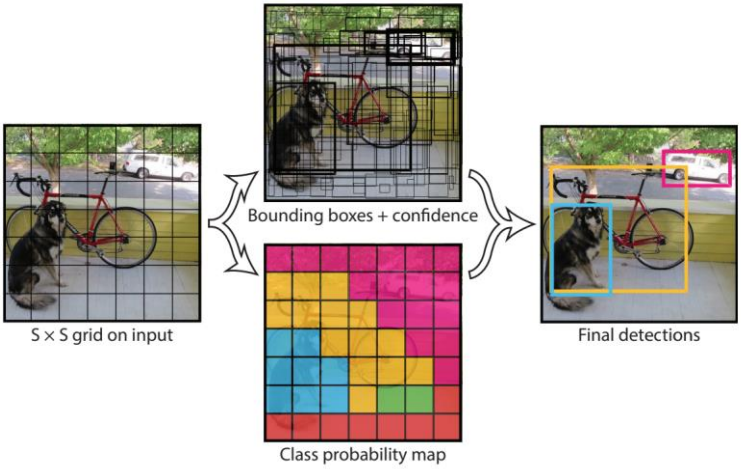
RMSProp (*Root Mean Square Propagation*) adalah metode pengoptimalan berbasis *adaptive learning rate* yang diusulkan oleh Geoffrey Hinton. RMSProp memodifikasi Adagrad dengan mengganti akumulasi gradien menjadi rata-rata bergerak gradien yang diberi bobot secara kuadratik.

2.4 You Only Look Once (YOLO)

You only look once adalah sebuah struktur yang berfungsi untuk mendeteksi sebuah objek yang berskala Real-time [13].

YOLO memiliki versi yaitu Yolo V1, Yolo V2, dan Yolo V3. Kebanyakan metode *object detection* selain YOLO pertama 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 CNN sekali saja untuk memprediksi objek apa saja yang ada pada gambar, hal ini bisa terjadi karena YOLO membagi gambar menjadi grid sebesar *S* x *S*, setiap sel bertanggung jawab untuk memprediksi *bounding boxes* dan tingkat *confidence* dari setiap sel serta probabilitas kelas. Keterangan, S merupakan besaran grid dari sebuah gambar. Untuk lebih jelasnya dapat dilihat pada Gambar 2.4.1



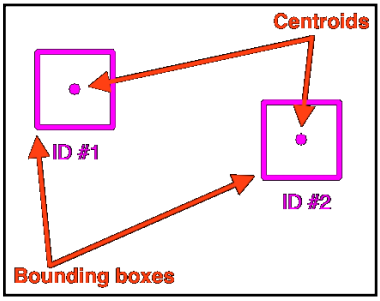
Gambar 2.4.1. Ilustrasi Deteksi YOLO [1].

2.5 Centroid Tracking

Dalam deteksi objek, salah satu masalah yang biasa terjadi adalah mengidentifikasikan objek dalam sebuah identitas yang sama. Hal ini menjadi tantangan dalam sebuah pendeteksian objek

utamanya yang memerlukan pelacakan objek. *Object tracking* merupakan salah satu solusi atas permasalahan tersebut. Dengan melakukan *object tracking*, objek yang dideteksi akan memiliki identitas yang berbeda dengan objek lain dalam label yang sama. Salah satu algoritma *object tracking* sederhana adalah menggunakan *centroid-based tracking* [14].

Algoritma ini disebut *centroid tracking* karena bergantung pada jarak *Euclidean* antara objek *centroid* yang ada (misalnya, objek yang memiliki *centroid* sebelumnya) dan objek baru antara frame-frame berikutnya dalam sebuah video. Ilustrasi *centroid* pada *bounding box* dapat dilihat Gambar 2.5.1.

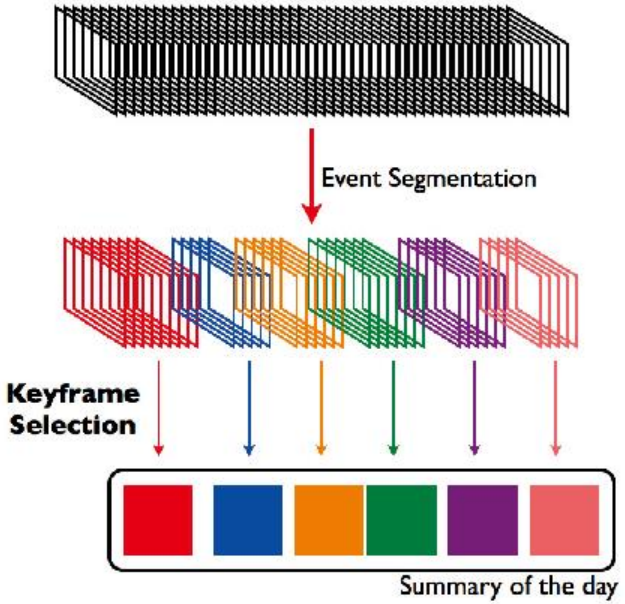


Gambar 2.5.1. *Centroid* dari *bounding box* [15]

2.6 Keyframe Selection

Keyframe Selection merupakan sebuah teknik untuk mengekstraksi *keyframe* dari sekuential video secara otomatis [16]. Teknik ini didesain untuk digunakan dalam sistem pencarian video di internet. Teknik ini terdiri dari tiga langkah yaitu, segmentasi *boundaries*, pemilihan *boundaries*, dan *keyframe selection* dari hasil pemilihan *boundaries*. *Boundaries* dan *keyframe* dipilih berdasarkan ukuran gerak dan aktivitas spasial seperti contoh aktifitas manusia. Selain berdasarkan ukuran gerak, terdapat teknik *Time Constrained* yang dapat digunakan dalam *keyframe selection*. Dengan menentukan sebuah nilai *k*, frame akan dipilih adalah frame dengan kelipatan $k + 1$. Keterangan *k* merupakan sebuah

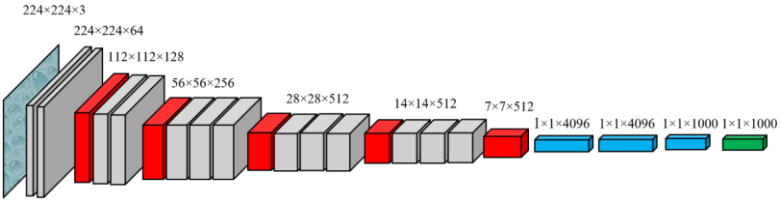
nilai dalam melakukan *keyframe selection* dengan teknik *Time Constrained*. Ilustrasi tahap *keyframe selection* dapat dilihat pada Gambar 2.6.1.



Gambar 2.6.1. Ilustrasi Proses Keyframe Selection [16].

2.7 VGG Network

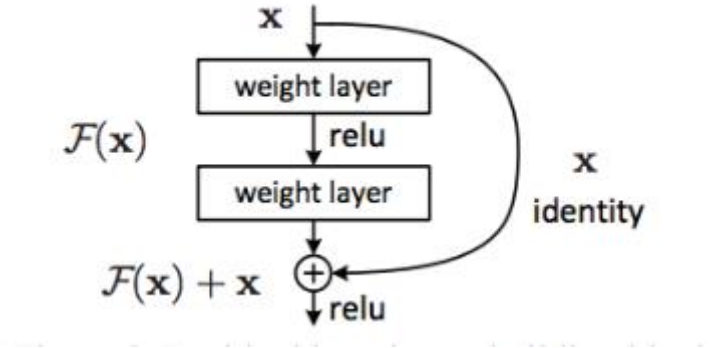
VGG *network* adalah arsitektur *Convolutional Neural Network* yang dirancang oleh Karen Simonyan dan Andrew Zisserman dari Visual Geometry Group, Department of Engineering Science, University of Oxford. Input yang digunakan berupa RGB berukuran 224 × 224 *pixels*. *Convolutional layer* yang digunakan dalam arsitektur ini ada 2 jenis, yaitu *convolutional layer* dengan ukuran filter 3x3 (conv3) dan ukuran filter 1x1 (conv1). Ukuran *convolutional layer* yang digunakan bermacam-macam, yaitu 64x64, 128x128, 256x256, dan 512x512 seperti yang ditunjukkan pada Gambar 2.7.1.



Gambar 2.7.1. Arsitektur VGG Network [17].

2.8 Residual Neural Network

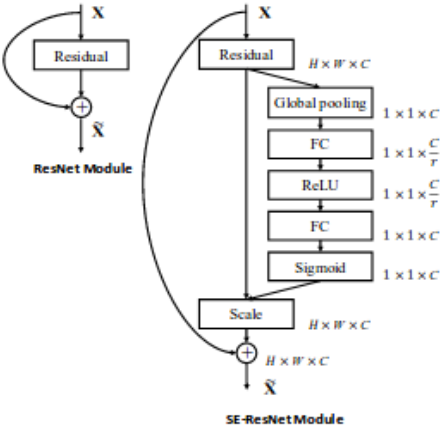
Residual neural network atau yang biasa disebut sebagai Resnet adalah salah satu jenis arsitektur yang cukup populer, arsitektur ini dibuat oleh Kaiming He et al [5]. Arsitektur ini cukup revolusioner pada saat itu karena arsitektur ini menjadi *state of the art* pada saat itu tidak hanya dalam klasifikasi, namun dalam semua kategori termasuk *object detection*, dan *semantic segmentation*. Arsitektur CNN yang memiliki kedalaman tinggi adalah salah satu hal penting dalam membangun model CNN yang memiliki performa yang baik, namun model CNN yang memiliki kedalaman yang tinggi juga memiliki masalah, yaitu *vanishing gradient problem* yang dapat menyebabkan suatu CNN tidak dapat belajar dari *error* yang telah dikalkulasi. Hal yang diusung oleh Kaiming He et al. pada saat itu adalah dengan menggunakan sesuatu yang bernama *residual block*, blok ini adalah blok yang ada pada tiap lapis arsitektur CNN Resnet dan menjadi fundamental dari arsitektur tersebut.



Gambar 2.8.1. Blok residual pada Resnet [5]

2.9 Squeeze and Excitation Network

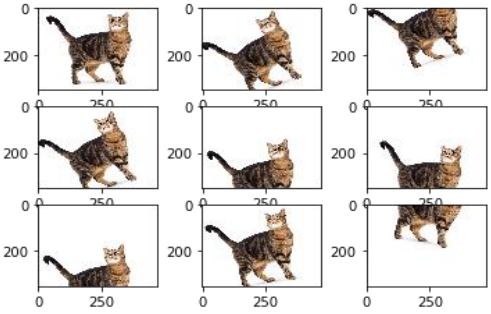
Squeeze-and-Excitation network atau yang biasa disebut sebagai SENets adalah arsitektur yang dirancang oleh Jie Hu, Li Shen dan Gang Sun [18]. Senet adalah *wrapper* untuk *neural network* yang ada seperti ResNet. Senet digunakan pada kompetisi ImageNet tahun 2017 dan meningkatkan hasil dari tahun sebelumnya sebesar 25%. SENets menggunakan strategi multi skala dan *multicrop* untuk mendapatkan 2,251% *loss* dan mendapatkan peringkat 5 teratas pada data uji dalam tantangan klasifikasi ILSVRC-2017. Secara umum, SENets terdiri atas operasi *squeeze* dan operasi *excitation*. Operasi *squeeze* bertujuan untuk menentukan *pooling layer* yang tidak memiliki pengaruh besar terhadap *depth network*. Operasi *Excitation* digunakan untuk memutuskan *feature layer* yang penting dan tidak. Ilustrasi operasi dapat dilihat pada Gambar 2.9.1.



Gambar 2.9.1. *Wrapper* SENet pada arsitektur Resnet [18]

2.10 Augmentation

Augmentation adalah sebuah teknik memanipulasi sebuah data tanpa kehilangan inti atau esensi dari data tersebut [3]. Proses ini digunakan untuk mengurangi *overfitting*. Jika terjadi *overfitting* maka performa *training* set cenderung lebih baik daripada *test* set. Augmentasi akan membuat data latih terlihat berbeda dengan menambahkan proses kombinasi seperti rotasi, *shift*, *flip*¸*shear* dan lainya. Contoh augmentasi gambar dapat dilihat pada Gambar 2.10.1.



Gambar 2.10.1. Contoh Augmentasi [19].

2.11 Confusion Matrix

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

Tabel 1.1. Kombinasi Nilai.

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

TP (*True Positive*) adalah ketika yang diprediksi positif dan kenyataannya benar. Contohnya, wanita yang diprediksi hamil dan sesungguhnya benar hamil. *TN* (*True Negative*) adalah ketika yang diprediksi negatif dan kenyataannya benar. Contohnya, lelaki yang diprediksi tidak hamil dan sesungguhnya benar tidak hamil. *FP* (*False Positive*) adalah ketika yang diprediksi positif namun kenyataannya salah. Contohnya, lelaki yang diprediksi hamil namun sesungguhnya tidak hamil. *FN* (*False Negative*) adalah ketika yang diprediksi negatif namun kenyataannya salah. Contohnya, wanita yang diprediksi tidak hamil namun sesungguhnya hamil. Nilai-nilai di atas dapat digunakan untuk mengukur tingkat validasi data. Beberapa jenis teknik validasi yang umum digunakan antara lain:

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

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

(2)

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

(3)

2.12 Flask

Flask adalah sebuah web framework yang ditulis dengan bahasa Python dan tergolong sebagai jenis microframework [20]. Flask berfungsi sebagai kerangka kerja aplikasi dan tampilan dari suatu web. Dengan menggunakan Flask dan bahasa Python, pengembang dapat membuat sebuah web yang terstruktur dan dapat mengatur lingkungan suatu web dengan lebih mudah.

Flask termasuk pada jenis microframework karena tidak memerlukan suatu alat atau pustaka tertentu dalam penggunaannya. Sebagian besar fungsi dan komponen umum seperti validasi form, database, dan sebagainya tidak terpasang secara default di Flask. Hal ini dikarenakan fungsi dan komponen komponen tersebut sudah disediakan oleh pihak ketiga dan Flask dapat menggunakan ekstensi yang membuat fitur dan komponenkomponen tersebut seakan diimplementasikan oleh Flask sendiri.

Selain itu, meskipun Flask disebut sebagai microframework, bukan berarti Flask mempunyai kekurangan dalam hal fungsionalitas. Microframework disini berarti bahwa Flask bermaksud untuk membuat core dari aplikasi ini sesederhana mungkin tapi tetap dapat dengan mudah ditambahkan. Dengan begitu, fleksibilitas serta skalabilitas dari Flask dapat dikatakan cukup tinggi dibandingkan dengan framework lainnya.

2.13 MySQL

MySQL adalah sebuah *database management system* (manajemen basis data) menggunakan perintah dasar SQL (*Structured Query Language*) yang cukup terkenal. SQL sendiri

merupakan suatu bahasa yang dipakai di dalam pengambilan data pada *relational database* atau database yang terstruktur. Jadi MySQL adalah *database management system* yang menggunakan bahasa SQL sebagai bahasa penghubung antara perangkat lunak aplikasi dengan database server.

Meskipun menjadi database yang cukup populer, MySQL tentu mempunyai beberapa kelebihan dan kekurangan dibandingkan dengan database server lainnya. Salah satu kekurangan MySQL adalah performanya turun di saat beberapa database manajemen sistem mampu bekerja baik pada pengelolaan database yang besar.

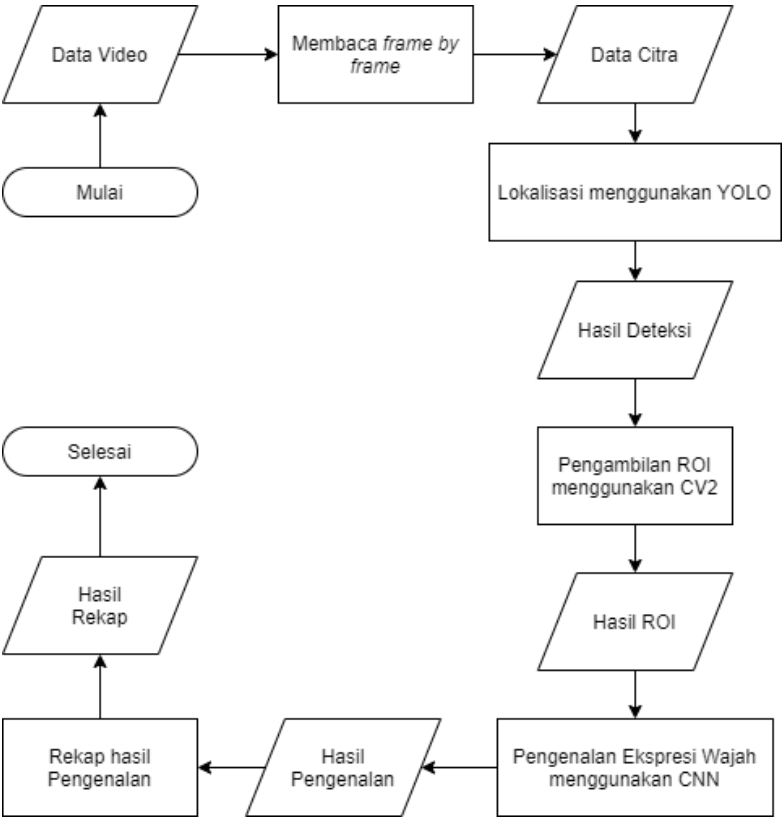
MySQL dapat dipasang pada server dengan spesifikasi kecil. Dengan kapasitas 1 GB, MySQL masih bisa digunakan sebagai database. MySQL adalah sistem manajemen database gratis. Selain itu, Banyak komunitas dan dokumentasi yang membahas soal MySQL.

MySQL mendukung berbagai macam sistem operasi (*cross-platform*), khususnya Linux dan Windows. Untuk pengguna Windows dapat menginstall XAMPP untuk menjalankan MySQL server yang di dalamnya sudah terdapat modul untuk menjalankan Apache, PHP, FileZilla, dan Tomcat. Sedangkan bagi pengguna Linux, dapat menginstall MySQL secara terpisah atau menginstall LAMP (Linux, Apache, MySQL, PHP) yang sudah disediakan modul Apache dan PHP juga.

BAB III
PERANCANGAN SISTEM

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

3.1 Desain Umum Sistem



Gambar 3.1.1. Diagram alir sistem yang dibangun

Sistem monitoring perkuliahan yang dibangun memiliki proses utama di antaranya lokalisasi, *Region Of Interest* (ROI), dan pengenalan ekspresi wajah. Diagram alir dari sistem ditunjukkan pada Gambar 3.1.1.

Proses lokalisasi merupakan tahap pertama dan dapat dibilang tahap yang paling penting dari sistem. Pada tahap ini posisi wajah ditentukan menggunakan deteksi objek dengan model yang sudah dilatih. Masukan pada tahap ini adalah citra CCTV dan keluarannya adalah hasil lokalisasi wajah. Hasil koordinat atau letak wajah dari proses ini nantinya akan digunakan pada tahap pengambilan *Region Of Interest.*

Pada proses pengambilan ROI, lokasi wajah hasil lokalisasi akan dipetakan dan disegmentasi menjadi gambar-gambar tersendiri. Masukan pada tahap ini adalah citra kamera CCTV dan keluarannya adalah kumpulan citra wajah yang berhasil tersegmentasi. Kumpulan citra wajah nantinya akan digunakan sebagai masukkan pada proses pengenalan ekspresi wajah

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

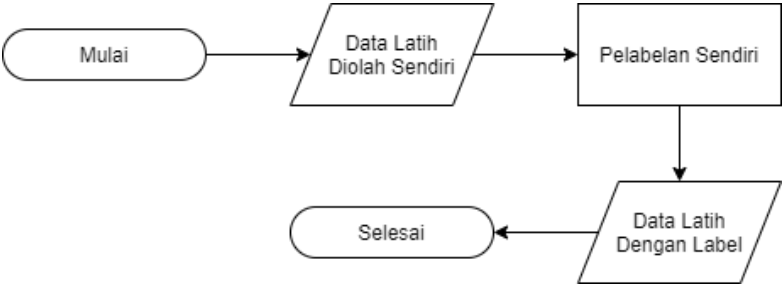
3.2 Perancangan Data

Perancangan data merupakan proses menentukan isi dan pengaturan data yang dibutuhkan untuk mendukung berbagai rancangan sistem yang akan dilakukan. Pada bagian ini, akan dilakukan 3 jenis perancangan data didasarkan pada tujuan dari masing masing perancangan data.

Perancangan data yang akan dilakukan antara lain, yang pertama adalah perancangan data diolah secara mandiri digunakan

untuk mendapatkan label pada masing-masing data yang akan dilatih pada tahap pengenalan ekspresi wajah. Yang kedua adalah perancangan data CK+ digunakan untuk mengurangi jumlah frame redundan dan menyesuaian area wajah. Yang ketiga adalah perancangan data deteksi wajah digunakan untuk membuat anotasi dan label yang sesuai dengan cara mandiri.

3.2.1 Perancangan Data Diolah Secara Mandiri



Gambar 3.2.1. Diagram alir perancangan data diolah secara mandiri

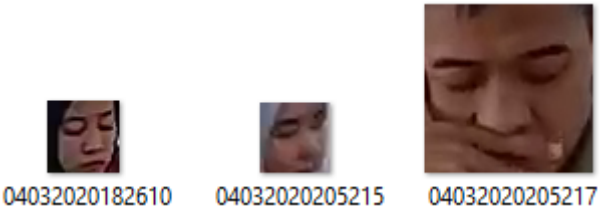
Data yang digunakan sebagai masukan awal dari tahap perancangan data diolah secara mandiri adalah data yang telah digunakan pada saat mata kuliah mahasiswa S2 Tahun Ajaran 2019/2020. Data ini masih belum terlabeli namun sudah dalam bentuk *Region of Interest*. Contoh data diolah secara mandiri dapat dilihat pada Gambar 3.2.2 dan Gambar 3.2.3. Spesifikasi data rekaman dapat dilihat pada Tabel 3.1. Tabel Spesifikasi Data Rekaman S2.

Tabel 3.1. Tabel Spesifikasi Data Rekaman S2

Keterangan	**Spesifikasi**
Ukuran resolusi asli	1920 x 1080
Ekstensi	.mp4
Durasi	1 – 10 menit
Ukuran file	8,212 – 45,210 KB
Kanal warna	3 (RGB)

Kriteria dalam melakukan pelabelan setiap data adalah dengan melakukan observasi terhadap data rekaman mahasiswa S2 Tahun Ajaran 2019/2020. Setelah melakukan observasi, dilakukan pemilahan potongan wajah yang sesuai dengan subjek yang diamati. Tidak sepenuhnya setiap subjek hanya dilabeli dalam sebuah kelas, terdapat beberapa frame potongan wajah yang dilabeli sebagai kelas yang lainnya. Contoh potongan wajah dengan label positif dapat dilihat pada Gambar 3.2.2 dan dengan label negatif pada Gambar 3.2.3.

Gambar 3.2.2. Dataset dengan label positif



Gambar 3.2.3. Dataset dengan label negatif

Spesifikasi lengkap data diolah secara mandiri dapat dilihat pada Tabel 3.2. Tabel spesifikasi data diolah secara mandiri.

Tabel 3.2. Tabel spesifikasi data diolah secara mandiri

Keterangan	**Spesifikasi**
Ukuran resolusi asli	32 x 32 s/d 90 x 90

Ekstensi	.jpg
Jumlah gambar	1070
Ukuran file	20 - 50 kB
Kanal warna	3 (RGB)

Proses melakukan pelabelan terhadap data diatas dilakukan secara mandiri dengan kemampuan penulis dalam melakukan pemilahan data menjadi 2 kelas. Jumlah data gambar setelah proses pelabelan pada kelas positif berjumlah 670 citra dan pada kelas negatif berjumlah 400 citra. Setelah melakukan proses pelabelan, data tersebut akan dipisah dengan persentase 80% untuk data latih dan 20% untuk data uji. Dengan jumlah data latih sebanyak 856 citra dan data uji sebanyak 214.

3.2.2 Pelabelan Kelompok Ekspresi

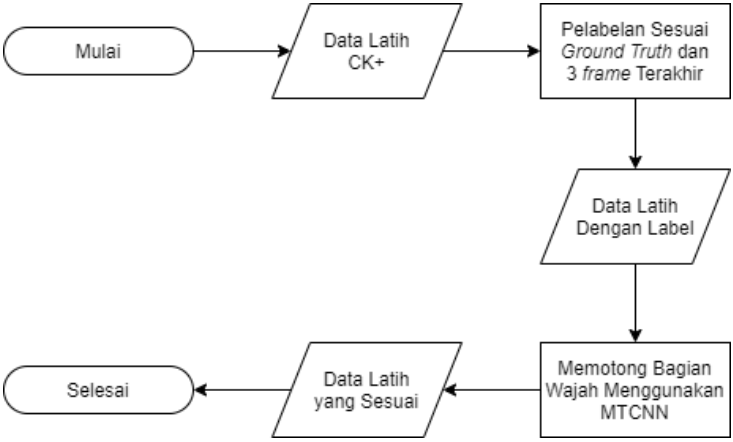
Penentuan data merupakan proses menentukan data yang akan dikelompokkan berdasarkan dari hasil metodologi survei yang telah diimplementasikan. Survei merupakan suatu teknik pengumpulan informasi yang dilakukan dengan cara menyusun daftar pertanyaan yang diajukan pada responden dalam berbentuk sample dari sebuah populasi.

Dengan menggunakan *platform* yang disediakan oleh *Google Form*, survei dapat dilaksanakan secara daring. Responden akan diberikan beberapa pertanyaan singkat yang terdiri dari 2 bagian. Bagian pertama merupakan pertanyaan identitas responden beserta tata cara pengisian, dan kedua merupakan pertanyaan mengenai kelompok ekspresi wajah.

Survei bersifat *blind question* yang dimana responden akan disajikan hanya berupa sebuah gambar ekspresi wajah, lalu untuk menjawab terdapat pilihan ganda dengan jawaban hanya ekspresi positif atau ekspresi negatif. Gambar ekspresi yang disajikan merupakan satu sampel dari ekspresi wajah pada data CK+. Keluaran dari survei dapat dikumpulkan dan digunakan sebagai

kelompok ekspresi wajah yang akan digunakan pada tahap selanjutnya.

3.2.3 Perancangan Data CK+



Gambar 3.2.4. Diagram alir perancangan data CK+

Data yang digunakan sebagai masukan awal dari tahap perancangan data CK+ adalah data yang digunakan pada beberapa referensi mengenai *facial expression* [7]. Data tersebut masih berupa kumpulan *frame* perubahan ekspresi wajah normal menuju ekspresi yang diinginkan. Contoh kumpulan *frame* data CK+ dapat dilihat pada Gambar 3.2.5.

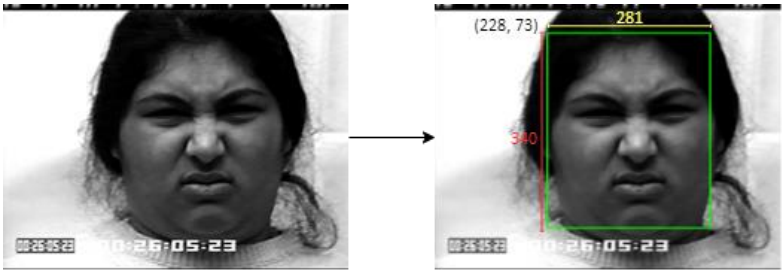
Gambar 3.2.5. Perubahan ekspresi normal menuju terkejut [7]

Spesifikasi lengkap CK+ dapat dilihat pada Tabel 3.3. Tabel spesifikasi data CK+.

Tabel 3.3. Tabel spesifikasi data CK+

Keterangan	**Spesifikasi**
Ukuran resolusi asli	640 x 490
Ekstensi	.png
Jumlah gambar	10714
Ukuran file	100 - 150 kB
Kanal warna	1 (Single)

Proses melakukan pelabelan terhadap data diatas dilakukan berdasarkan *ground truth* yang telah disediakan. Dari kumpulan data tersebut hanya akan diambil 3 *frame* terakhir yang dapat mewakili ekspresi wajah dan mengurangi redundansi. Setelah tahap pelabelan, data citra akan dilakukan *cropping* sesuai dengan *Region of Interest* yang akan digunakan. Proses menentukan ROI menggunakan *library* yang telah disediakan, yaitu *Multi-task Cascaded Convolutional Network* (MTCNN). Dengan data masuk adalah hasil dari proses pelabelan sebelumnya dan keluaran adalah titik ROI. Ilustrasi dapat dilihat pada Gambar 3.2.6.



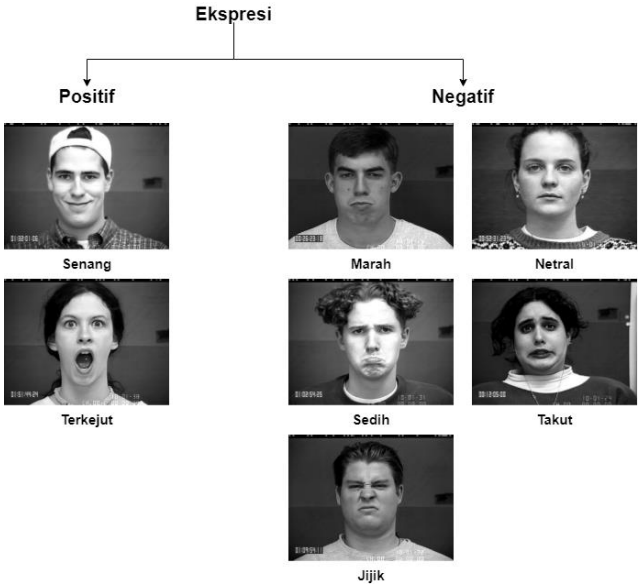
Gambar 3.2.6. Keluaran proses MTCNN

Jumlah data gambar setelah proses diatas, pada kelas *anger* (marah) berjumlah 135 citra, kelas *netral* (netral) berjumlah 54 citra, kelas *disgust* (jijik) berjumlah 177 citra, kelas *fear* (takut) berjumlah 75 citra, kelas *happy* (senang) berjumlah 207 citra, kelas *sadness* (sedih) berjumlah 84 citra dan kelas *surprise* (terkejut) berjumlah 249 citra. Setelah melakukan proses pelabelan dan penentuan ROI, data tersebut akan dipisah dengan persentase 80% untuk data latih dan 20% untuk data uji. Dengan jumlah data latih sebanyak 784 citra dan data uji sebanyak 197. Hasil dari tahap perancangan data dapat dilihat pada Gambar 3.2.7.

Gambar 3.2.7. Hasil perancangan data CK+

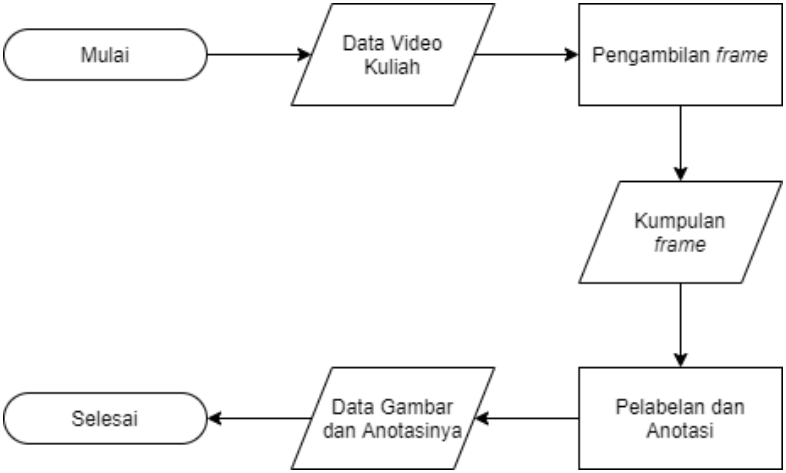
Untuk data CK+, digunakan 2 skenario dalam pembuatan model pengenalan ekspresi wajah. Pertama menggunakan kelas asli pada data CK+ seperti kelas *anger* (marah), kelas *netral* (netral), kelas *disgust* (jijik), kelas *fear* (takut), kelas *happy*

(senang), kelas *sadness* (sedih) dan kelas *surprise* (terkejut). Kedua menggunakan kelas dengan pengelompokan ekspresi positif terdiri dari kelas senang, dan terkejut. Sedangkan untuk kelas negatif terdiri dari kelas marah, sedih, netral, takut dan jijik. Contoh pengelompokan ekspresi dapat dilihat pada Gambar 3.2.8.



Gambar 3.2.8. Pengelompokan kelas data CK+

3.2.4 Perancangan Data Untuk Deteksi Wajah



Gambar 3.2.9. Diagram alir perancangan data untuk deteksi

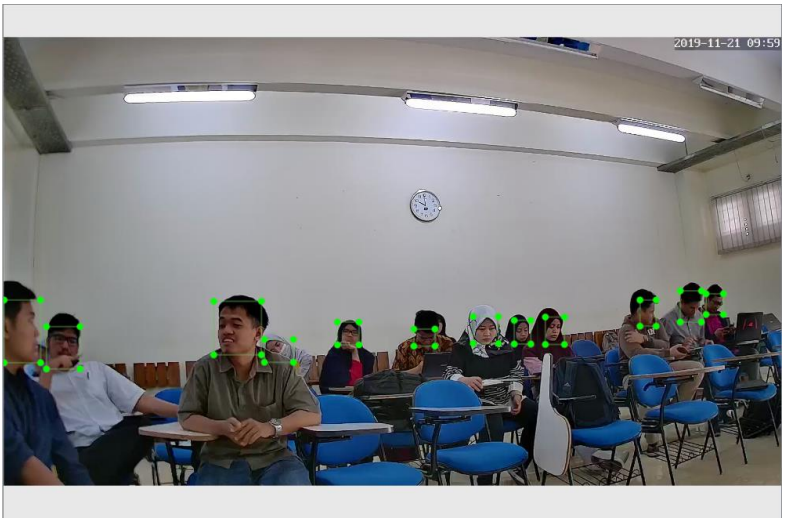
Data yang digunakan sebagai masukan awal dari tahap perancangan data untuk deteksi wajah adalah beberapa *frame* dari rekaman mata kuliah mahasiswa S2 Tahun Ajaran 2019/2020. Data ini masih belum terlabeli dan teranotasi. Proses selanjutnya akan membuat konfigurasi yang berisi informasi label dan anotasi dari kumpulan frame tersebut. Spesifikasi lengkap mengenai data untuk deteksi wajah dapat dilihat pada Tabel 3.4. Spesifikasi data untuk deteksi wajah. Contoh citra frame sebelum dilakukan pelabelan dan anotasi dapat dilihat pada Gambar 3.2.10.

Tabel 3.4. Spesifikasi data untuk deteksi wajah

Keterangan	**Spesifikasi**
Ukuran resolusi asli	1920 x 1080
Ekstensi	.png
Jumlah gambar	74
Ukuran file	300 - 400 kB
Kanal warna	3 (RGB)

Gambar 3.2.10. Contoh frame perancangan data

Pada proses pelabelan dan anotasi, setiap *frame* akan ditentukan secara mandiri label kelas wajah. Setelah menentukan label, dilakukan proses anotasi pada label yang telah ditentukan. Dengan menggunakan bantuan dari *Tools – LabelImg* [21], dapat dengan mudah melakukan anotasi pada label kelas wajah. Contoh hasil setelah dilakukan anotasi dapat dilihat pada Gambar 3.2.11.



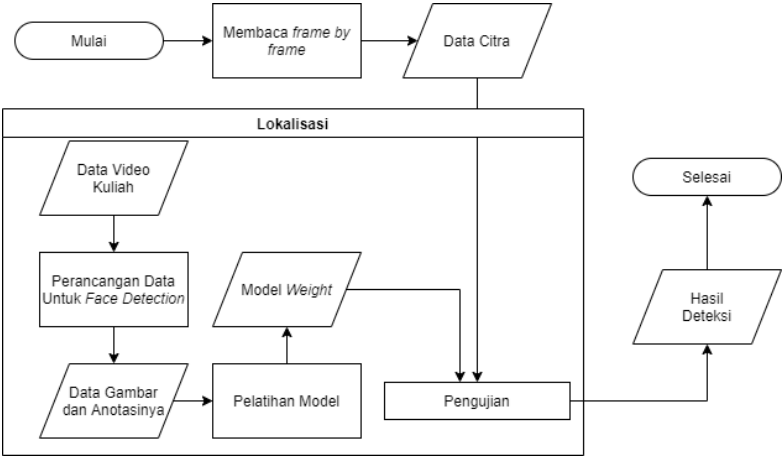
Gambar 3.2.11. Hasil setelah proses pelabelan dan anotasi

3.3 Perancangan Proses

Perancangan proses merupakan proses apa yang dilakukan oleh sistem yang akan dibuat. Pada bagian ini, akan dilakukan beberapa proses untuk mendapatkan hasil yang dapat menyelesaikan masalah yang disebutkan.

Pada tahap perancangan proses yang akan dilakukan antara lain, yang pertama adalah perancangan proses lokalisasi untuk mendapatkan ROI dari hasil deteksi wajah menggunakan *You Only Look Once* (YOLO). Yang kedua adalah pengambilan ROI yang digunakan untuk mendapatkan area wajah digunakan. Yang ketiga adalah pengenalan ekspresi wajah yang digunakan untuk melakukan prediksi terhadap area wajah yang telah ditentukan. Yang terakhir adalah rekap hasil pengenalan dengan proses penyimpana kedalam database dan informasi yang disediakan

3.3.1 Tahap Lokalisasi



Gambar 3.3.1. Perancangan Proses Lokalisasi

Pada tugas akhir ini, akan dilakukan tahap lokalisasi yang berfungsi untuk mendeteksi dan menentukan posisi wajah pada data masukan. Tahap ini secara garis besar dibagi menjadi 3 proses yang penting. Pertama, tahap perancangan data untuk deteksi yang sudah dijelaskan pada bagian **Perancangan Data Untuk Deteksi Wajah.** Kedua, tahap pelatihan model menggunakan *framework* Darknet dengan menggunakan *pretrained convolutional layer weights* yang telah disediakan. Ketiga, tahap pengujian dan evaluasi menggunakan *framework* Darknet dengan evaluasi menggunakan evaluasi mAP yang sudah disediakan oleh Darknet.

3.3.1.1 Perancangan Data Untuk Deteksi Wajah

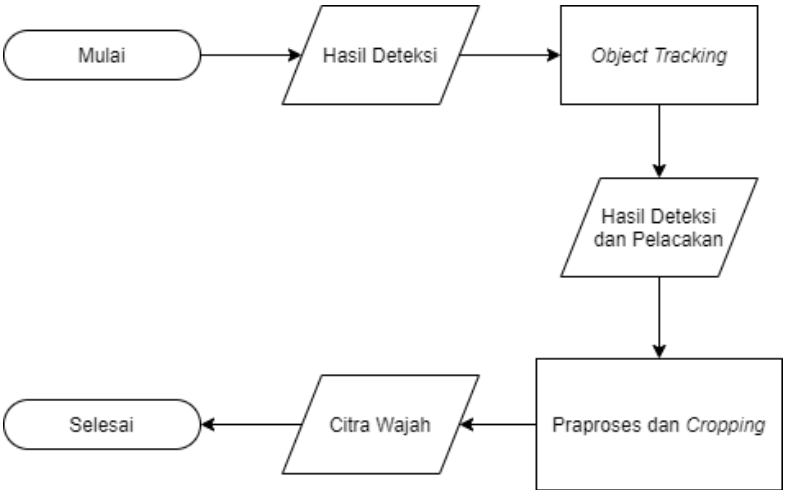
Tahap perancangan data untuk deteksi wajah terdiri dari 2 proses, yaitu pembacaan *frame* dan menentukan label serta anotasi. Adapun Langkah-langkah untuk melakukan tahap perancangan data untuk deteksi wajah sudah dijelaskan pada bagian **Perancangan Data Untuk Deteksi Wajah**.

3.3.1.2 Pelatihan Model

Data yang digunakan sebagai *ground truth* pada tahap ini adalah hasil pelabelan dan anotasi yang telah dilakukan pada perancangan data. Pelatihan model menggunakan *framework* Darknet dengan menggunakan pretrained *convolutional layer weights* dengan menggunakan data *WIDER FACE: A Face Detection Benchmark* yang telah disediakan. Pada tahap ini dilakukan konfigurasi *test* untuk arsitektur yang telah dibangun.

3.3.2 Tahap Pengambilan ROI

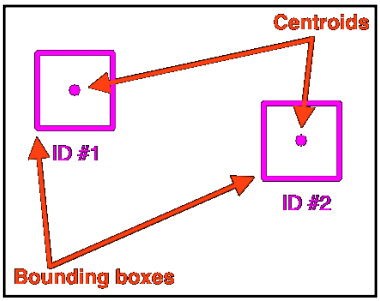
Pada tugas akhir ini, akan dilakukan tahap pengambilan ROI yang berfungsi untuk memecah citra pada hasil lokalisasi menjadi potongan area wajah. Tahap ini secara garis besar dibagi menjadi 3 bagian, yaitu *object tracking*, praproses dan *cropping*. Diagram alir tahap pengambilan ROI dapat dilihat pada Gambar 3.3.2.



Gambar 3.3.2. Diagram alir tahap pengambilan ROI.

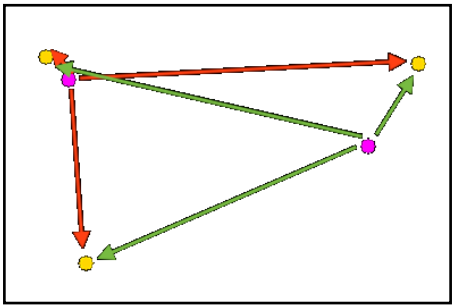
Tahap *object tracking* pada tahap ini digunakan untuk memberikan identitas pada hasil deteksi wajah. Tahap ini diawali dengan pembuatan *centroid* terhadap *bounding box* hasil deteksi

wajah. Setelah mendapatkan titik *centroid*, titik koordinat akan disimpan dalam sebuah *tuple* atau variabel. Pembuatan centroid dapat dilihat pada Gambar 3.3.3.



Gambar 3.3.3. Titik *centroid* pada *bounding box*

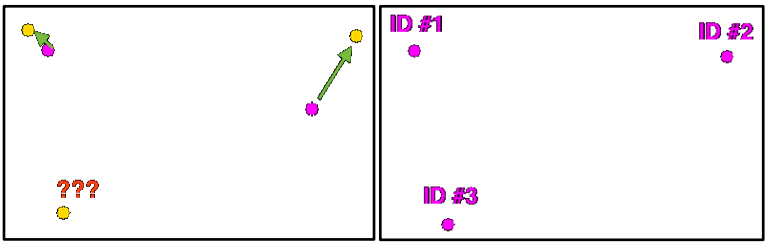
Setelah mendapatkan titik *centroid*, pada frame selanjutnya akan ditentukan *centroid* baru dan mengukur seluruh *Eucliedan* antara *centroid* lama dengan *centroid* baru. Ilustrasi centroid lama dan centroid baru dapat dilihat pada Gambar 3.3.4. Titik ungu merupakan titik *centroid* lama dan titik kuning merupakan titik *centroid* baru.



Gambar 3.3.4. *Euclidean centroid* lama dan baru

Setelah mendapatkan jarak antara *centroid*, dilakukan pembaharuan terhadap *tuple* atau variabel yang menyimpan titik centroid. Bila titik centroid tidak dapat menemukan centroid

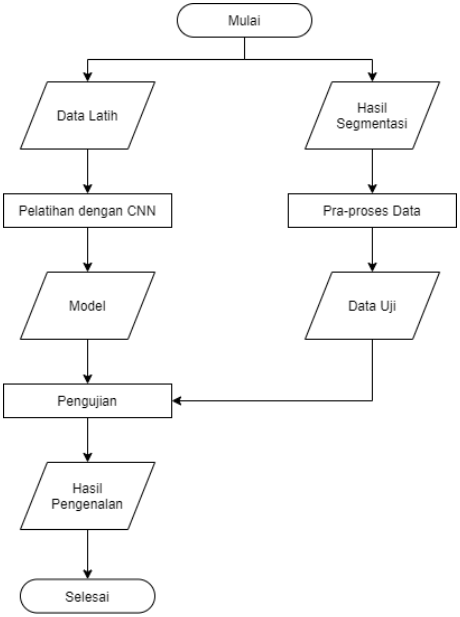
terdekat, maka akan didaftarkan sebagai identitas baru. Ilustrasi dapat dilihat pada Gambar 3.3.5.



Gambar 3.3.5. Ilustrasi identitas baru

Tahap praproses data pada bagian ini melakukan penyesuaian data citra agar dapat melewati tahap pengenalan ekspresi wajah yang akan dijelaskan pada bagian **Tahap Pengenalan Ekspresi Wajah.** Setelah dilakukan praproses data, data citra akan dilakukan pemotongan terhadap area wajah sesuai dengan hasil deteksi dan pelacakan.

3.3.3 Tahap Pengenalan Ekspresi Wajah

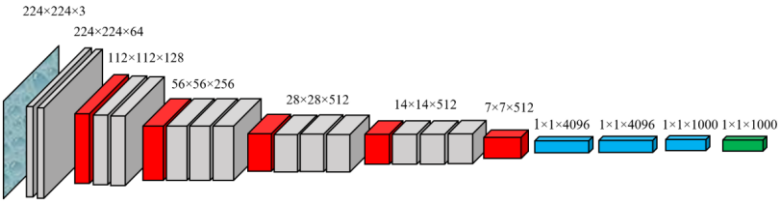


Gambar 3.3.6. Diagram alir tahap pengenalan ekspresi

Pada tugas akhir ini, akan dilakukan tahap pengenalan ekspresi wajah yang berfungsi untuk memberi label kelas pada citra-citra hasil lokalisasi. Tahap ini secara garis besar dibagi menjadi 3 bagian, yaitu pembangunan arsitektur CNN, pelatihan, dan pengujian. Diagram alir tahap segmentasi dapat dilihat pada Gambar 3.3.6.

3.3.3.1 Pembangunan Arsitektur

Pembangunan model bertujuan untuk menyiapkan *layer*, fungsi aktivasi, dan parameter apa saja yang dibutuhkan. Arsitektur VGG16 dapat dilihat pada Gambar 3.3.7.



Gambar 3.3.7. Arsitektur *VGG16 Network* [17].

Pembangunan arsitektur CNN mengikuti arsitektur *VGG16 Network* yang telah disediakan oleh Keras-TensorFlow. Selain menggunakan Keras-TensorFlow, terdapat beberapa model *pretrained* yang telah disediakan oleh keras_vggface [17]. Keras_vggface menyediakan arsitektur VGG16, Resnet50, dan Senet50.

3.3.3.2 Pelatihan Model

Dalam tahap ini, akan dilakukan proses pelatihan model dengan menggunakan 2 data latih. Diantaranya data latih diolah secara mandiri dan data latih CK+. Pada tahap perancangan data, telah dilakukan penyesuaian data agar dapat dimasukkan ke dalam tahap pelatihan model ini.

Untuk pelatihan data latih diolah secara mandiri, data akan dibagi menjadi 2, yaitu data latih dan data validasi dengan jumlah data latih sebanyak 856 citra sebelum augmentasi dan jumlah data validasi sebanyak 214 citra.

Untuk pelatihan data latih *The Extended Cohn-Kanade Dataset* (CK+), data akan dibagi menjadi 2, yaitu data latih dan data validasi dengan jumlah data latih sebanyak 784 citra sebelum augmentasi dan jumlah data validasi sebanyak 197 citra.

Proses pelatihan memanfaatkan data latih untuk membangun model CNN. Pelatihan menggunakan *activation function* ReLU dan Softmax dengan *learning rate* yang sudah ditentukan sebelumnya. Proses pelatihan akan dijalankan pada *batch size* 52 dan jumlah *epoch* menyesuaikan skenario uji coba. Pada proses ini

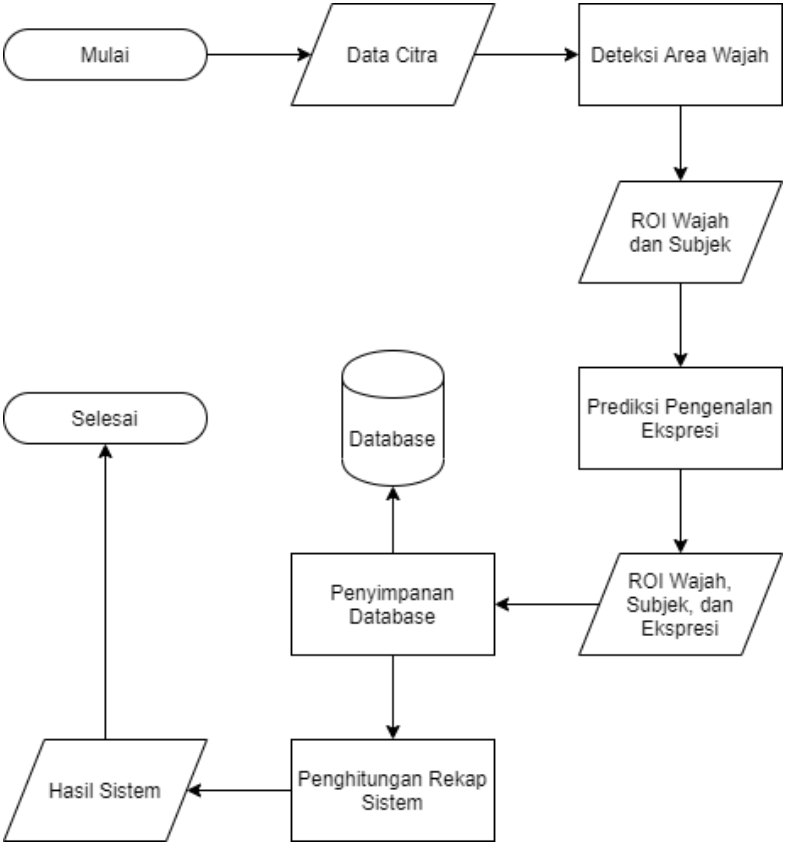
digunakan beberapa algoritma pengoptimalan seperti Adam, RMSProp, dan Adagrad. Dalam setiap akhir *epoch* terdapat proses pengujian model terhadap data validasi untuk mengetahui seberapa baik model dilatih. Lalu pada akhir pelatihan akan dilakukan pengujian untuk mendapatkan nilai akurasi, *precision,* dan *recall*.

3.3.3.3 Pengujian

Untuk tahap pengujian, citra hasil dari segmentasi akan diproses kembali terlebih dahulu. Setiap data dalam bentuk citra akan dilakukan proses pengubahan kedalam format RGB. Citra tersebut kemudian akan dilakukan *resize* untuk mengubah ukuran citra menjadi ukuran 224x224.

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

3.3.4 Tahap Rekap Hasil Pengenalan



Gambar 3.3.8. Diagram alir proses rekap hasil

Pada tugas akhir ini, akan dilakukan tahap rekap hasil pengenalan yang berfungsi untuk memberikan hasil rekap dari sistem monitoring. Setelah proses pengenalan ekspresi selesai, hasil prediksi akan disimpan kedalam database server. Segala informasi yang terdapat dalam database akan kumpulkan untuk mendapatkan rekap sistem monitoring ekspresi wajah.

Pada tahap perancangan antarmuka sistem, fitur yang disediakan antara lain, yang pertama adalah melihat file citra yang telah proses. Yang kedua adalah mengunggah file citra yang akan dilakukan proses pengenalan. Yang ketiga adalah fitur belakang untuk melakukan penyimpanan kedalam database server. Diperlukan struktur database yang dapat menunjang seluruh fitur dalam sistem monitoring ini. Spesifikasi struktur tabel dalam database dapat dilihat pada Tabel 3.5. Struktur tabel database.

Tabel 3.5. Struktur tabel database

Nama Kolom	**Tipe Data**
detect_id	INT (PRIMARY KEY)
file_name	VARCHAR(255)
frame_no	INT
rt	INT
rb	INT
lt	INT
lb	INT
ex_predict	VARCHAR(50)
ex_acc	FLOAT
face_predict	VARCHAR(50)
face_acc	FLOAT
face_name	VARCHAR(255)
centroid_x	INT
centroid_y	INT
timestamp	TIMESTAMP

Dari masing-masing nama kolom diatas memiliki fungsi dan tujuan dalam penyimpanan informasi sistem, berikut fungsinya:

- detect_id, menyimpan nilai id pada sebuah deteksi dan pengenalan

- file_name, meyimpan nama file citra yang digunakan

- frame_no, menyimpan nomor *frame* pada file citra yang digunakan

- rt, menyimpan nilai dari *right top* ROI

- rb, menyimpan nilai dari *right bottom* ROI

- lt, menyimpan nilai dari *left top* ROI

- lb, menyimpan nilai dari *left bottom* ROI

- ex_predict, menyimpan hasil prediksi kelas pengenalan ekspresi wajah

- ex_acc, menyimpan hasil akurasi prediksi pengenalan ekspresi wajah

- face_predict, menyimpan hasil prediksi kelas pengenalan wajah

- face_acc, menyimpan hasil akurasi prediksi pengenalan wajah

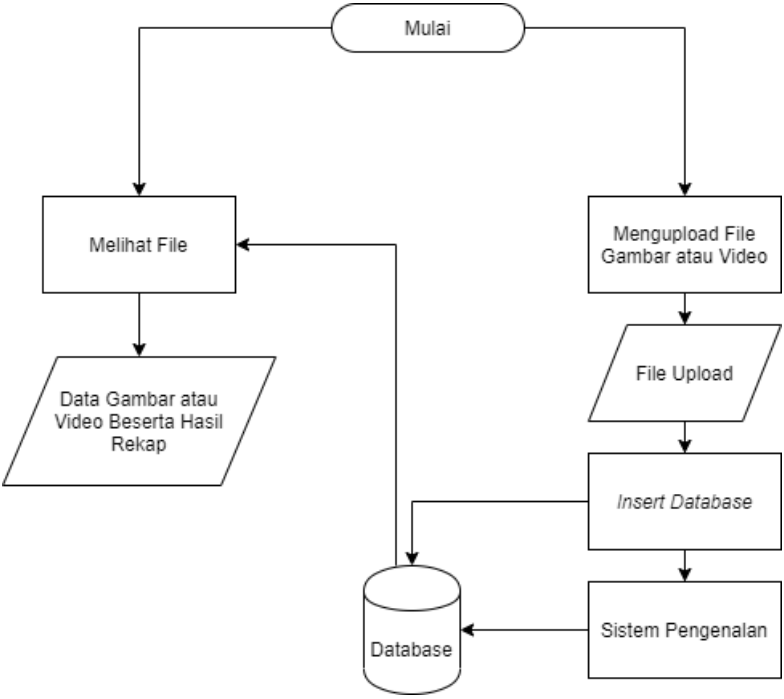
- face_name, sebagai indek pelacakan

- centroid_x, sebagai titik koordinat centroid *bounding box*

- centroid_y, sebagai titik koordinat centroid *bounding box*

- timestamp, menyimpan waktu eksekusi

3.4 Perancangan Antarmuka Sistem



Gambar 3.4.1. Diagram alir antarmuka sistem

Perancangan antarmuka sistem merupakan antarmuka sistem aplikasi yang telah dirancang dan diimplementasikan kepada pengguna. Pada bagian ini, akan dijelaskan alur dari perancangan sistem antarmuka. Diagram alir dapat dilihat pada Gambar 3.4.1.

3.4.1 Perancangan Antarmuka

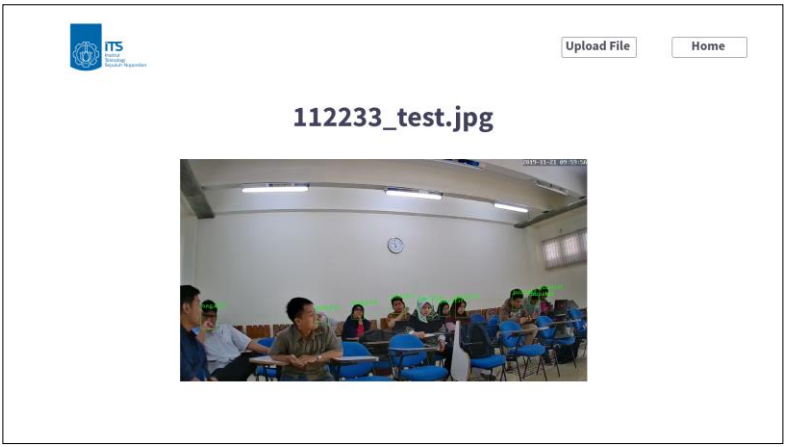
Sebagai salah satu alat berinteraksi dengan pengguna, perancangan antarmuka diharapkan dapat membantu pembangunan antarmuka pada tahap implementasi. Dari seluruh fitur yang disediakan, terdapat 3 antarmuka yang telah dirancang

untuk tahap implementasi. Rancangan antarmuka diantaranya halaman utama, halaman menggunggah, halaman menampilkan.

Pada halaman utama, sistem menyediakan informasi berupa daftar file yang berada pada server. Daftar file tersebut merupakan file yang telah diunggah melalui halaman unggah. Bila terjadi interaksi dengan daftar file, maka pengguna akan dialihkan menuju halaman menampilkan. Rancangan antarmuka halaman utama dapat dilihat pada Gambar 3.4.2.

Gambar 3.4.2. Desain antarmuka halaman utama

Pada halaman menampilkan, sistem menyediakan informasi berupa file yang tuju. File tersebut merupakan file yang telah diunggah melalui halaman unggah. Di halaman menampilkan ini, pengguna disajikan gambar atau video yang telah dilakukan proses pengenalan pada sistem monitoring kuliah. Rancangan antarmuka halaman menampilkan dapat dilihat pada Gambar 3.4.3.



Gambar 3.4.3. Desain antarmuka halaman menampilkan

Pada halaman menggunggah, sistem menyediakan formulir unggah file. File tersebut akan diunggah dan disimpan kedalam server sistem aplikasi. Di halaman menggunggah ini, pengguna akan menggunggah file gambar atau video untuk selanjutnya akan dilakukan proses pengenalan. Rancangan antarmuka halaman menggunggah dapat dilihat pada Gambar 3.4.4.

Gambar 3.4.4. Desain antarmuka halaman menggunggah

(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 *Google Collaboratory*. Sistem operasi yang digunakan adalah Ubuntu 18.04.3 LTS. *Cloud* yang digunakan memiliki spesifikasi Intel(R) Xeon(R) dengan kecepatan 2.00 GHz, *Random Access Memory* (RAM) sebesar 13 GB, dan mempunyai *Graphics Processing Unit* (GPU) yaitu NVIDIA Tesla K80 sebesar 16 GB.

Selain itu, tugas akhir ini menggunakan database server *cloud* dengan sistem operasi yang digunakan adalah Ubuntu 18.04.3 LTS. Memiliki spesifikasi Dual Core Intel Pentium (Xeon) family dengan kecepatan 2.00 GHz, *Random Access Memory* (RAM) sebesar 2 GB, dan *Hardisk Drive* (HDD) sebesar 20 GB.

4.1.2 Perangkat Lunak

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

4.2 Implementasi Perancangan Data

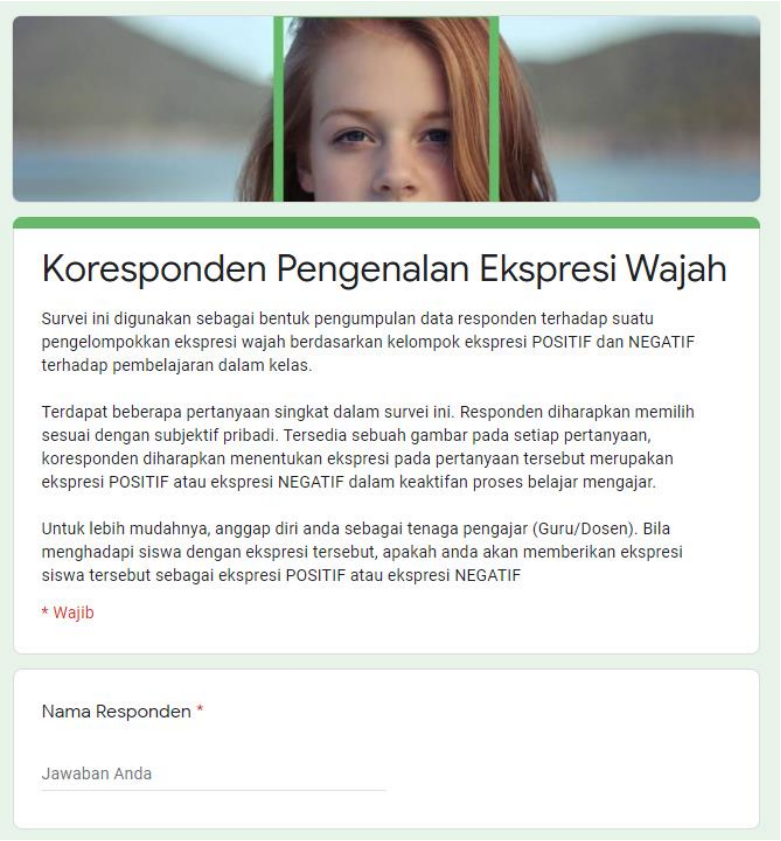
Pada subbab ini akan dijelaskan mengenai proses-proses yang diimplementasikan pada tahap perancangan data. Diantaranya implementasi perancangan data diolah secara mandiri, perancangan data CK+ dan perancangan data deteksi wajah.

4.2.1 Implementasi Perancangan Data Diolah Secara Mandiri

Pada implementasi perancangan data diolah secara mandiri terdapat satu tahap yang diimplementasikan. Pelabelan data secara mandiri dilakukan dengan melihat data wajah lalu memberikan label terhadap data wajah tersebut. Keluaran data pada implementasi ini adalah data gambar yang telah dilabeli.

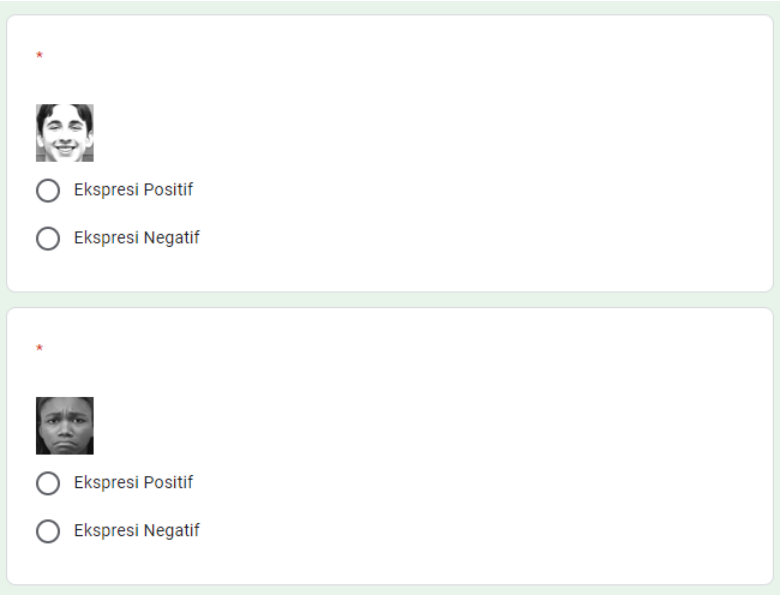
4.2.2 Implementasi Pelabelan Kelompok Ekspresi

Pada implementasi pelabelan kelompok ekspresi data CK+ terdapat satu tahap yang diimplementasikan. Pembuatan formulir pada *platform Google Form* dilakukan untuk dapat menyebarkan survei secara daring. Diberikan 2 bagian pertanyaan pada survei yang akan disebar. Bagian pertama merupakan tata cara pengisian dan pertanyaan identitas, hasil implementasi dapat dilihat pada Gambar 4.2.1.



Gambar 4.2.1. Bagian pertama survei

Pada bagian kedua, diberikan beberapa pertanyaan dengan setiap pertanyaan terdapat satu gambar ekspresi dengan pilihan ganda berupa ekspresi positif atau ekspresi negatif. Potongan survei bagian kedua dapat dilihat pada Gambar 4.2.2.



Gambar 4.2.2. Bagian kedua survei

Survei dapat diakses secara daring, maka disediakan sebuah halaman survei yang dapat diakses melalui sebuah tautan. Survei dapat dilihat pada tautan berikut ini:

https://intip.in/respondenfasma/

4.2.3 Implementasi Perancangan Data CK+

Pada implementasi perancangan data CK+ terdapat dua tahap yang diimplementasikan. Menentukan *ground truth* dari data CK+ dapat dilakukan dengan melihat label yang telah disediakan dari dataset *The Extended Cohn-Kanade Dataset* (CK+). Untuk melakukan pengambilan 3 *frame* terakhir menggunakan kode naskah python. Nama file dalam data CK+ mengandung informasi dengan pengkodean sebagai berikut :

'{*subject*}_{*video*}_{*frame_no*}.png'

Dengan informasi dan penyusunan nama file tersebut, dapat diambil 3 *frame* terakhir dengan melakukan penelusuran terhadap setiap *subject* dan *video*. Sebagai contoh, perhatikan Kode Sumber 4.1. Daftar nama file CK+. *Subject* contoh adalah S138 dan *video* adalah 001. Permasalahan ini dapat diselesaikan dengan menggunakan algoritma *sorting* ataupun prinsip *Last In First Out* (LIFO).

```
1.  S138_001_01515104.png
2.  S138_001_01515105.png
3.  S138_001_01515106.png
4.  S138_001_01515107.png
5.  S138_001_01515108.png
6.  S138_001_01515109.png
7.  S138_001_01515110.png
8.  S138_001_01515111.png
9.  S138_001_01515112.png
10. S138_001_01515113.png
11. S138_001_01515114.png
12. S138_001_01515115.png
```

Kode Sumber 4.1. Daftar nama file CK+

Implementasi selanjutnya adalah memotong area wajah menggunakan MTCNN. Dengan menggunakan library MTCNN, hasil dari proses deteksi area wajah dapat langsung digunakan untuk memotong area wajah. Keluaran dari deteksi menggunakan MTCNN adalah *array of object* yang terdeteksi. Masing-masing objek memiliki nilai *box*, *keypoints*, dan *confidence*. Untuk penggunaan MTCNN, implementasi dapat dilihat pada Kode Sumber 4.2. Implementasi MTCNN.

```
1.  filename = 'S138_001_01515112.png'
2.  pixels = cv2.imread(filename)
3.  detector = MTCNN()
4.  faces = detector.detect_faces(pixels)
```

Kode Sumber 4.2. Implementasi MTCNN

Pada baris pertama, inisialisasi variabel *filename* dengan nama file data CK+ yang akan dideteksi. Baris kedua, membaca file dengan nama file yang telah didefinisikan pada baris pertama lalu menyimpan kedalam *pixels*. Baris ketiga, membuat detektor dengan default weight yang telah disediakan. Baris keempat, mendeteksi *pixel* dengan menggunakan *detector* yang telah dibuat pada baris ketiga.

4.2.4 Implementasi Perancangan Data Deteksi Wajah

Pada implementasi perancangan data deteksi wajah terdapat dua tahap yang diimplementasikan. Yang pertama adalah implementasi pengambilan *frame.* Data awal merupakan data video akan diekstraksi menjadi kumpulan *frame*. Implementasi pengambilan *frame* dapat dilihat pada Kode Sumber 4.3. Implementasi pembacaan *frame*.

```
1. vidcap = cv2.VideoCapture('dataset.mp4')
2. success,image = vidcap.read()
3. count = 0
4. while success:
5.   cv2.imwrite("frame%d.png" % count, image)
6.   success,image = vidcap.read()
7.   print ('Read a new frame: ', success)
8.   count += 1
```

Kode Sumber 4.3. Implementasi pembacaan *frame*

Pada baris pertama, menginisiasi data video dan disimpan kedalam variabel *vidcap*. Baris kedua, melakukan pembacaan *frame* dengan variabel *success* bernilai benar saat *frame* dapat dibaca oleh sistem. Baris kelima, melakukan ekstraksi pada data video dan membuat file data gambar.

Implementasi selanjutnya adalah melakukan anotasi dan pelabelan pada data yang telah diekstraksi. Dengan menggunakan

library LabelImg, kumpulan data gambar dalam sebuah folder dapat dengan mudah dilakukan anotasi dan pelabelan. Implementasi pelabelan dan anotasi dapat dilihat pada

```
1. labelimg {direktori data gambar} {nama file
   kelas}
2. # contoh
3. labelimg ./extract ./class.txt
```

Kode Sumber 4.4. Implementasi anotasi dan pelabelan

4.3 Implementasi Perancangan Proses

Pada subbab ini akan dijelaskan mengenai proses-proses yang diimplementasikan pada tahap perancangan proses. Diantaranya implementasi perancangan. Diantaranya implementasi tahap lokalisasi, implementasi tahap pengambilan ROI, implementasi tahap pengenalan ekspresi wajah, dan implementasi tahap rekap hasil pengenalan.

4.3.1 Implementasi Tahap Lokalisasi

Pada implementasi tahap ini, keluaran yang diharapkan adalah data gambar yang telah dilabeli dan telah dianotasi. Adapun implementasi untuk melakukan tahap perancangan data untuk deteksi wajah sudah dijelaskan pada bagian **Implementasi Perancangan Data Deteksi Wajah.**

Tahap selanjutnya adalah implementasi pelatihan model. Implementasi pelatihan model menggunakan *framework* Darknet dengan menggunakan *pretrained convolutional layer weights* yang telah disediakan. Untuk mendapatkan *pretrained convolutional layer weights*, dapat dilihat pada Kode Sumber 4.5. Implementasi mengunduh *pretrained*.

```
1. !wget
   http://pjreddie.com/media/files/darknet53.conv.74
```

Kode Sumber 4.5. Implementasi mengunduh *pretrained*

Setelah mengunduh *convolutional layer weights,* diperlukan konfigurasi lapisan arsitektur yang akan dibentuk. Pada konfigurasi ini ditentukan juga beberapa parameter pendukung seperti jumlah *batch*, *subdivision*, jumlah *step per epoch*, dan jumlah kelas sesuai dengan konfigurasi pada *WIDER FACE: A Face Detection Benchmark*. Setelah konfigurasi dan *pretrained weight* siap, dilakukan proses pelatihan dengan menggunakan Darknet. Implementasi tahap pelatihan model dapat dilihat pada Kode Sumber 4.6. Implementasi pelatihan deteksi wajah.

```
1. !./darknet detector train data/obj.data
   cfg/yolov3_custom.cfg darknet53.conv.74 -dont_show
```

Kode Sumber 4.6. Implementasi pelatihan deteksi wajah

Perintah diatas terdiri dari beberapa parameter yang memiliki fungsi masing-masing. *Detector* merupakan salah satu modul pada Darknet yang digunakan dalam tahap deteksi objek. *Train* merupakan parameter fungsi untuk melakukan pelatihan model diikuti dengan parameter data yang akan dilatih dilanjutkan dengan parameter konfigurasi dan parameter selanjutnya adalah model *pretrained* yang digunakan.

Tahap selanjutnya adalah implementasi tahap pengujian. Pada implementasi ini, model yang telah dilatih pada implementasi sebelumnya akan diuji untuk melakukan deteksi. Implementasi tahap pengujian dapat dilihat pada Kode Sumber 4.7. Implementasi pengujian deteksi wajah.

```
1. !./darknet detector test data/obj.data cfg/yolov3_custom.cfg
   backup/yolov3_custom_last.weights /images/test.jpg -
   thresh 0.5
2. !./darknet detector map data/obj.data
   cfg/yolov3_custom.cfg
   backup/yolov3_custom_last.weights -iou_thresh 0.5
```

Kode Sumber 4.7. Implementasi pengujian deteksi wajah

Baris pertama digunakan untuk melakukan pengujian pada sebuah data gambar '*image/test.jpg*' dengan menggunakan model yang telah dilatih. *Detector* merupakan salah satu modul pada Darknet yang digunakan dalam tahap deteksi objek. *Test* merupakan parameter fungsi untuk melakukan pengujian model diikuti dengan parameter data uji dilanjutkan dengan parameter konfigurasi dan parameter selanjutnya adalah model *pretrained* yang digunakan. Parameter '*-thres*' digunakan untuk melakukan penyesuaian nilai *threshold* pada deteksi objek.

Baris kedua digunakan untuk melakukan evaluasi pada data yang telah disebutkan pada '*obj.data*'. *Map* merupakan parameter fungsi untuk melakukan penghitungan evaluasi dengan menggunakan *mean, accuracy, precision* (mAP) pada deteksi objek. Dilanjutkan dengan parameter data uji dan data latih pada file '*obj.data*', kemudian parameter konfigurasi dan parameter selanjutnya adalah model *pretrained* yang digunakan. Parameter '*-iou_thres*' digunakan untuk melakukan penyesuaian nilai *threshold* terhadap *Intersect Over Union* (IOU).

4.3.2 Implementasi Tahap Pengambilan ROI

Implementasi tahap pengambilan ROI berfungsi untuk memecah citra pada hasil lokalisasi menjadi potongan area wajah. Implementasi tahap ini secara garis besar dibagi menjadi 3 bagian, yaitu *object tracking*, praproses dan *cropping*. Pada tahap pertama dilakukan *object tracking* dengan masukkan adalah hasil deteksi wajah. Diperlukan penghitungan titik koordinat *centroid* dari hasil

deteksi wajah yang berupa titik minimal x, maksimal x, minimal y, dan maksimal y. Implementasi perhitungan titik koordinat *centroid* dapat pada Kode Sumber 4.8. Implementasi titik koordinat *centroid*

```
1. cX = int((startX + endX) / 2.0)
2. cY = int((startY + endY) / 2.0)
3. inputCentroids[i] = (cX, cY)
```

Kode Sumber 4.8. Implementasi titik koordinat *centroid*

Variabel *startX* merupakan nilai minimal koordinat x pada hasil deteksi wajah. Variabel *endX* merupakan nilai maksimal koordinat x pada hasil deteksi wajah. Variabel *startY* merupakan nilai minimal koordinat y pada hasil deteksi wajah. Variabel *endY* merupakan nilai maksimal koordinat y pada hasil deteksi wajah. Selanjutnya menghitung jarak *Euclidean* pada titik *centroid* lama dan baru. Setelah mendapatkan jarak, diambil jarak paling minimal dari setiap centroid. Implementasi kode sumber dapat dilihat pada Kode Sumber 4.9. Implementasi *Euclidean* terdekat.

```
1. D = dist.cdist(np.array(objectCentroids),
   inputCentroids)
2. rows = D.min(axis=1).argsort()
3. cols = D.argmin(axis=1)[rows]
```

Kode Sumber 4.9. Implementasi *Euclidean* terdekat

Pada baris pertama, digunakan *library scipy.spatial* untuk menghitung jarak dari setiap *centroid.* Pada baris kedua dan ketiga adalah mencari jarak terpendek antara setiap titik *centroid*. Implementasi proses *cropping* dapat dilihat pada Kode Sumber 4.10. Implementasi tahap pengambilan ROI.

```
4.  crop_img = img[y:y+h, x:x+w]
```

Kode Sumber 4.10. Implementasi tahap pengambilan ROI

Pada kode sumber diatas terdapat beberapa nilai variabel. Variabel '*y*' merupakan nilai letak terhadap sumbu y. Variabel '*x*' merupakan nilai letak terhadap sumbu x. Variabel '*h*' merupakan nilai dari tinggi area wajah. Dan variabel '*w*' merupakan nilai dari lebar area wajah.

4.3.3 Implementasi Tahap Pengenalan Ekspresi Wajah

Implementasi tahap pengenalan ekspresi wajah berfungsi untuk memberi label kelas pada citra-citra hasil lokalisasi. Implementasi tahap ini secara garis besar dibagi menjadi 3 bagian, yaitu implementasi pembangunan arsitektur CNN, implementasi pelatihan, dan implementasi pengujian.

4.3.3.1 Implementasi Pembangunan Arsitektur CNN

Implementasi pembangunan arsitektur bertujuan untuk menyiapkan *layer*, fungsi aktivasi, dan parameter apa saja yang dibutuhkan. Arsitektur yang digunakan merupakan *pretrained model* yang disediakan oleh *keras_vggface* [17]. Model yang disediakan adalah VGG16, Resnet50 dan Senet50.

```
1.  vgg_model = VGGFace(include_top=False, input_shape
    = (224, 224, 3))
2.  last_layer = vgg_model.get_layer('pool5').output
3.  x = Flatten(name='flatten')(last_layer)
4.  x = Dense(512, activation='relu', name='fc6')(x)
5.  x = Dense(512, activation='relu', name='fc7')(x)
6.  out = Dense(2, activation='softmax', name='fc8')(x)
7.  model = Model(vgg_model.input, out)
```

Kode Sumber 4.11. Implementasi pembangunan 2 kelas VGG16

Implementasi pembangunan 2 kelas menggunakan VGG16 dapat dilihat pada Kode Sumber 4.11. Implementasi pembangunan 2 kelas VGG16. Pada baris pertama, dilakukan proses inisiasi model *pretrained* dengan model VGG16 dengan beberapa parameter. Parameter *'include_top'* merupakan penanda lapisan atas disertakan atau tidak pada model. Parameter '*input_shape*' merupakan nilai dari dimensi input model.

Pada baris kedua, dilakukan pencarian lapisan dengan nama '*pool5*'. Pada baris 6, dilakukan pengubahan nilai jumlah kelas dengan menggunakan aktivasi fungsi *softmax*. Pada baris 7, dilakukan perangkaian arsitektur dengan masukkan dari model dan dan keluaran yang telah dimodifikasi pada baris 3-6.

```
1. vgg_model = VGGFace( model = 'resnet50', include_top
   = False, input_shape = (224, 224, 3))
2. last_layer = vgg_model.get_layer('avg_pool').output
3. x = Flatten(name='flatten')(last_layer)
4. out = Dense(2, activation='softmax', name='classifier')(x)
5. model = Model(vgg_model.input, out)
```

Kode Sumber 4.12. Implementasi pembangunan 2 kelas Resnet50

Implementasi pembangunan 2 kelas menggunakan Resnet50 dapat dilihat pada Kode Sumber 4.12. Implementasi pembangunan 2 kelas Resnet50. Pada baris pertama, dilakukan proses inisiasi model *pretrained* dengan model Resnet50 dengan beberapa parameter. Parameter *'include_top'* merupakan penanda lapisan atas disertakan atau tidak pada model. Parameter '*input_shape*' merupakan nilai dari dimensi input model.

Pada baris kedua, dilakukan pencarian lapisan dengan nama '*avg_pool*'. Pada baris 4, dilakukan pengubahan nilai jumlah kelas dengan menggunakan aktivasi fungsi *softmax*. Pada baris 5, dilakukan perangkaian arsitektur dengan masukkan dari model dan dan keluaran yang telah dimodifikasi.

```
1. vgg_model = VGGFace( model = 'senet50', include_top =
   False, input_shape = (224, 224, 3))
2. last_layer = vgg_model.get_layer('avg_pool').output
3. x = Flatten(name='flatten')(last_layer)
4. out = Dense(2, activation='softmax', name='classifier')(x)
5. model = Model(vgg_model.input, out)
```

Kode Sumber 4.13. Implementasi pembangunan 2 kelas Senet50

Implementasi pembangunan 2 kelas menggunakan Senet50 dapat dilihat pada Kode Sumber 4.13. Implementasi pembangunan 2 kelas Senet50. Pada baris pertama, dilakukan proses inisiasi model *pretrained* dengan model Senet50 dengan beberapa parameter. Parameter *'include_top'* merupakan penanda lapisan atas disertakan atau tidak pada model. Parameter '*input_shape*' merupakan nilai dari dimensi input model.

Pada baris kedua, dilakukan pencarian lapisan dengan nama '*avg_pool*'. Pada baris 4, dilakukan pengubahan nilai jumlah kelas dengan menggunakan aktivasi fungsi *softmax*. Pada baris 5, dilakukan perangkaian arsitektur dengan masukkan dari model dan dan keluaran yang telah dimodifikasi.

```
1. vgg_model = VGGFace(include_top=False, input_shape
   = (224, 224, 3))
2. last_layer = vgg_model.get_layer('pool5').output
3. x = Flatten(name='flatten')(last_layer)
4. x = Dense(512, activation='relu', name='fc6')(x)
5. x = Dense(512, activation='relu', name='fc7')(x)
6. out = Dense(7, activation='softmax', name='fc8')(x)
7. model = Model(vgg_model.input, out)
```

Kode Sumber 4.14. Implementasi pembangunan 7 kelas VGG16

Implementasi pembangunan 7 kelas menggunakan VGG16 dapat dilihat pada Kode Sumber 4.14. Implementasi pembangunan 7 kelas VGG16. Pada baris pertama, dilakukan proses inisiasi

model *pretrained* dengan model VGG16 dengan beberapa parameter. Parameter *'include_top'* merupakan penanda lapisan atas disertakan atau tidak pada model. Parameter '*input_shape*' merupakan nilai dari dimensi input model. Arsitektur ini nantinya akan digunakan dalam pelatihan model menggunakan data CK+.

```
1. vgg_model = VGGFace( model = 'resnet50', include_top
   = False, input_shape = (224, 224, 3))
2. last_layer = vgg_model.get_layer('avg_pool').output
3. x = Flatten(name='flatten')(last_layer)
4. out = Dense(7, activation='softmax', name='classifier')(x)
5. model = Model(vgg_model.input, out)
```

Kode Sumber 4.15. Implementasi pembangunan 7 kelas Resnet50

Implementasi pembangunan 7 kelas menggunakan Resnet50 dapat dilihat pada Kode Sumber 4.15. Implementasi pembangunan 7 kelas Resnet50Kode Sumber 4.14. Implementasi pembangunan 7 kelas VGG16. Pada baris pertama, dilakukan proses inisiasi model *pretrained* dengan model Resnet50 dengan beberapa parameter. Arsitektur ini nantinya akan digunakan dalam pelatihan model menggunakan data CK+. Arsitektur ini nantinya akan digunakan dalam pelatihan model menggunakan data CK+.

```
1. vgg_model = VGGFace( model = senet50, include_top =
   False, input_shape = (224, 224, 3))
2. last_layer = vgg_model.get_layer('avg_pool').output
3. x = Flatten(name='flatten')(last_layer)
4. out = Dense(7, activation='softmax', name='classifier')(x)
5. model = Model(vgg_model.input, out)
```

Kode Sumber 4.16. Implementasi pembangunan 7 kelas Senet50

Implementasi pembangunan 7 kelas menggunakan Senet50 dapat dilihat pada Kode Sumber 4.16. Implementasi pembangunan 7 kelas Senet50. Pada baris pertama, dilakukan proses inisiasi

model *pretrained* dengan model Resnet50 dengan beberapa parameter. Arsitektur ini nantinya akan digunakan dalam pelatihan model menggunakan data CK+. Arsitektur ini nantinya akan digunakan dalam pelatihan model menggunakan data CK+.

4.3.3.2 Implementasi Pelatihan Model

Implementasi pelatihan model bertujuan menentukan parameter terbaik dalam sebuah model pengenalan ekspresi wajah. Beberapa algoritma pengoptimalan yang digunakan adalah Adam, RMSProp dan Adagrad. *Learning rate* yang digunakan pada setiap algoritma pengoptimalan adalah 0,001, 0,0001, dan 0,00001. Ditentukan nilai *Epoch* sebanyak 25 dan nilai *batch_size* sebanyak 52.

```
1. INIT_LR = [1e-3, 1e-4, 1e-5]
2. EPOCHS = 25
3. BS = 52
```

Kode Sumber 4.17. Implementasi pelatihan parameter

Implementasi pelatihan dengan melakukan inisiasi nilai dari setiap skenario dapat dilihat pada Kode Sumber 4.17. Implementasi pelatihan parameter. Variabel '*INIT_LR*' menyimpan nilai dari *learning_rate* yang akan digunakan dalam setiap skenario. Variabel '*EPOCHS*' menyimpan nilai jumlah *epoch* yang akan digunakan. Variabel '*BS*' menyimpan jumlah *batch*_size yang akan digunakan.

```
1. opt = Adam(lr=INIT_LR)
2. model.compile(loss="categorical_crossentropy",
   optimizer=opt, metrics=["accuracy"])
```

Kode Sumber 4.18. Implementasi pelatihan Adam

Implementasi pelatihan dengan menggunakan pengoptimalan Adam dapat dilihat pada **Kode Sumber 4.18.**

Implementasi pelatihan Adam. Parameter '*lr*' digunakan sebagai parameter jumlah *learning rate*. Pada baris kedua, dilakukan penyusunan model dengan *loss function* menggunakan *crossentropy*, *optimizer* menggunakan Adam, dan *metrics* menggunakan evaluasi performasi berdasarkan akurasi.

```
1. opt = RMSprop(lr=INIT_LR)
2. model.compile(loss="categorical_crossentropy",
   optimizer=opt, metrics=["accuracy"])
```

Kode Sumber 4.19. Implementasi pelatihan RMSprop

Implementasi pelatihan dengan menggunakan pengoptimalan RMSprop dapat dilihat padaKode Sumber 4.19. Implementasi pelatihan RMSprop. Parameter '*lr*' digunakan sebagai parameter jumlah *learning rate*. Pada baris kedua, dilakukan penyusunan model dengan *loss function* menggunakan *crossentropy*, *optimizer* menggunakan RMSprop, dan *metrics* menggunakan evaluasi performasi berdasarkan akurasi.

```
1. opt = Adagrad(lr=INIT_LR)
2. model.compile(loss="categorical_crossentropy",
   optimizer=opt, metrics=["accuracy"])
```

Kode Sumber 4.20. Implementasi pelatihan Adagrad

Implementasi pelatihan dengan menggunakan pengoptimalan Adagrad dapat dilihat pada Kode Sumber 4.20. Implementasi pelatihan Adagrad. Parameter '*lr*' digunakan sebagai parameter jumlah *learning rate*. Pada baris kedua, dilakukan penyusunan model dengan *loss function* menggunakan *crossentropy*, *optimizer* menggunakan Adagrad, dan *metrics* menggunakan evaluasi performasi berdasarkan akurasi.

```
1. model.fit(
2. aug.flow(trainX, trainY, batch_size=BS),
3. steps_per_epoch=len(trainX) // BS,
4. validation_data=(testX, testY),
5. validation_steps=len(testX) // BS,
6. epochs=EPOCHS
7. )
```

Kode Sumber 4.21. Implementasi pelatihan

Implementasi pelatihan dapat dilihat pada Kode Sumber 4.21. Implementasi pelatihan. Pada kode sumber tersebut terdapat beberapa parameter yang dimasukkan. Variabel '*trainX*' dan '*trainY*' merupakan data yang telah dibagi dengan pembagian sebanyak 80:20. Parameter *step_per_epoch* digunakan sebagai jumlah *step* yang dilakukan pada setiap *epoch.* Parameter *validation_data* digunakan sebagai masukkan data uji. Parameter *epochs* merupakan jumlah *epoch* yang digunakan.

4.3.3.3 Implementasi Pengujian Model

Implementasi pengujian model bertujuan menentukan hasil evaluasi dalam pelatihan sebuah model. Data masuk dalam tahap ini adalah data uji dengan variabel '*testX*'. Implementasi pengujian model dapat dilihat pada Kode Sumber 4.22. Implementasi pengujian model.

```
1. model.predict(testX, batch_size=BS)
```

Kode Sumber 4.22. Implementasi pengujian model

4.3.4 Implementasi Tahap Rekap Hasil Pengenalan

Implementasi tahap rekap hasil pengenalan bertujuan untuk memberikan hasil rekap dari sistem monitoring. Setelah proses pengenalan ekspresi selesai, hasil prediksi akan disimpan kedalam database server. Segala informasi yang terdapat dalam database

akan kumpulkan untuk mendapatkan rekap sistem monitoring ekspresi wajah.

Implementasi awal dalam tahap ini adalah membentuk struktur database yang akan digunakan. Segala informasi yang terjadi dalam aplikasi harus disimpan kedalam database. Pada bagian **Tahap Rekap Hasil Pengenalan.** Implementasi tahap rekap hasil pengenalan dapat dilihat pada Kode Sumber 4.23. Implementasi *Data Definition Language*.

```
1.  CREATE TABLE IF NOT EXISTS `detections` (
2.    `detect_id` int(10) unsigned zerofill NOT NULL
    AUTO_INCREMENT,
3.    `file_name` varchar(255) NOT NULL,
4.    `frame_no` int(11) NOT NULL,
5.    `rt` int(11) NOT NULL,
6.    `rb` int(11) NOT NULL,
7.    `lt` int(11) NOT NULL,
8.    `lb` int(11) NOT NULL,
9.    `ex_predict` varchar(50) DEFAULT NULL,
10.   `ex_acc` float DEFAULT NULL,
11.   `face_predict` varchar(50) DEFAULT NULL,
12.   `face_acc` float DEFAULT NULL,
13.   `face_name` varchar(255) NOT NULL,
14.   `centroid_x` int(11) NOT NULL,
15.   `centroid_y` int(11) NOT NULL,
16.   `timestamp` timestamp NULL DEFAULT
    CURRENT_TIMESTAMP,
17.   PRIMARY KEY (`detect_id`)
18. ) ENGINE=InnoDB AUTO_INCREMENT=54 DEFAULT
    CHARSET=latin1;
```

Kode Sumber 4.23. Implementasi *Data Definition Language*

Pada baris pertama, dilakukan pengecekan nama tabel terhadap sebuah database. Bila nama tabel belum didefinisikan,

maka akan dilakukan pembentukan struktur tabel sesuai dengan baris 2-15.

```
1. sql_insert_detections = "INSERT INTO detections (file_name,
   face_name, frame_no, rt, rb, lt, lb, ex_predict, ex_acc)
   VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)"
2. sql_get_detections = "SELECT * FROM detections where
   file_name = %s"
3. SELECT face_name, frame_no, ex_predict, ex_acc,
   detect_conf FROM faces_bener WHERE file_name = '%s'
   AND face_name = '%s';
4. SELECT face_name,  AVG(detect_conf) AS avg_conf,
   AVG(ex_acc) AS AVG, COUNT(if(ex_predict =
   'negatif',1,NULL)) AS jml_neg, COUNT(if(ex_predict =
   'positif',1,NULL)) AS jml_pos, COUNT(*) AS jml_frame,
   COUNT(if(ex_predict = 'negatif',1,NULL)) / COUNT(*) AS
   percen_neg, COUNT(if(ex_predict = 'positif',1,NULL)) /
   COUNT(*) AS percen_pos FROM faces_bener WHERE
   file_name = '%s' and face_name != '-' GROUP BY face_name;
```

Kode Sumber 4.24. Implementasi model *query*

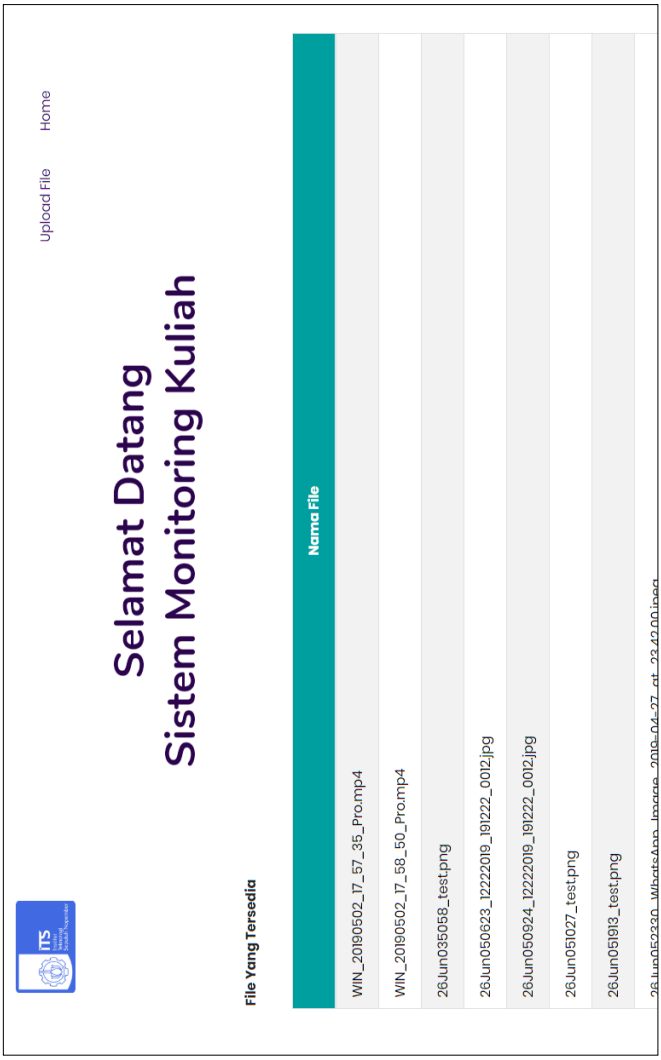
Baris pertama merupakan kode sumber untuk melakukan penyimpanan data kedalam database. Baris kedua merupakan kode sumber untuk melakukan pengambilan data dari database.

Baris ketiga merupakan kode sumber untuk menampilkan rekap hasil pengenalan terhadap sebuah subjek tertentu. Baris keempat merupakan kode sumber untuk menampilkan rekap hasil rata-rata setiap subjek dalam sebuah video.

4.3.5 Implementasi Tahap Antarmuka

Implementasi tahap antarmuka menjadi salah satu alat berinteraksi dengan pengguna. Dari seluruh fitur yang disediakan, terdapat 3 antarmuka yang telah dirancang untuk tahap implementasi. Rancangan antarmuka diantaranya halaman utama,

halaman menggunggah, halaman menampilkan. Halaman utama yang sudah diimplementasikan dapat dilihat pada Gambar 4.3.1. Dan implementasi pembuatan antarmuka halaman utama dapat dilihat pada Kode Sumber 4.25. Implementasi antarmuka halaman utama. Halaman penampilan yang sudah diimplementasikan dapat dilihat pada Gambar 4.3.2 dan implementasi pembuatan antarmuka halaman penampilan dapat dilihat pada Kode Sumber 4.26. Implementasi antarmuka halaman penampilan. Halaman pengunggahan yang sudah diimplementasikan dapat dilihat pada Gambar 4.3.3 dan implementasi pembuatan antarmuka halaman pengunggahan dapat dilihat pada Kode Sumber 4.27. Implementasi antarmuka halaman pengunggahan.

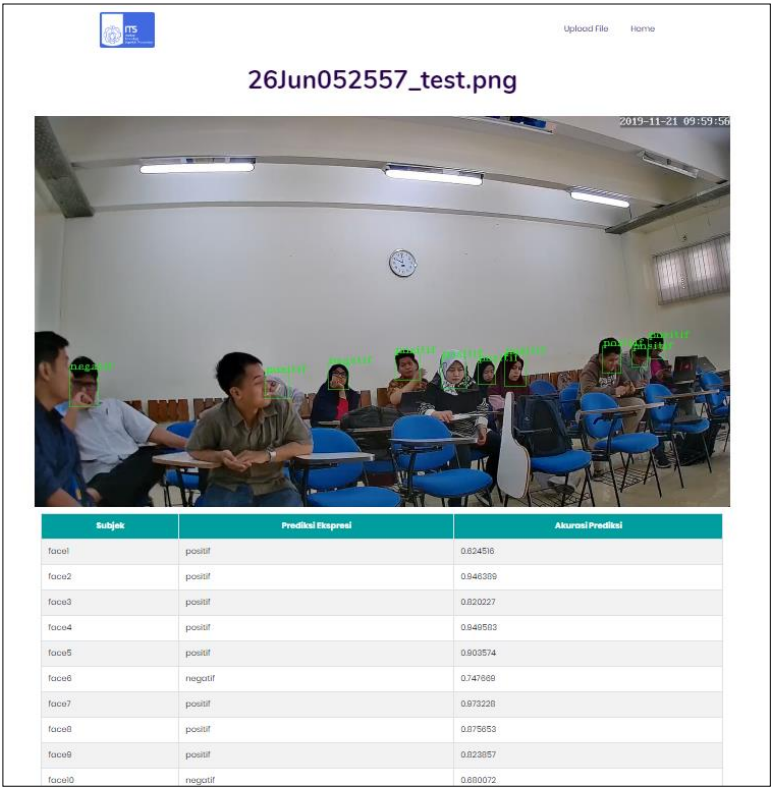


Gambar 4.3.1. Implementasi antarmuka halaman utama

```
1.  %%writefile templates/home.html
2.  {% extends "base.html" %}
3.  {% block content %}
4.  <section class="slider-area service-area section-padding2">
5.  <div class="single-slider container">
6.  <div class="row d-flex justify-content-center">
7.  <div class="row">
8.  <div class="col-lg-12">
9.  <div class="section-tittle text-center  slider-active">
10. <h2 data-animation="fadeInUp" data-delay=".4s">Selamat
    Datang<br>Sistem Monitoring Kuliah</h2>
11. </div>
12. </div>
13. <div class="col-md-12">
14. <div class="form-group row">
15. <label id="cari" class="col-md-12 font-weight-bold">File
    Yang Tersedia</label>
16. </div>
17. </div>
18. <div class="table-responsive">
19. <table id="example" class="table table-striped table-
    bordered dt-responsive" cellspacing="0" width="100%">
20. <thead>
21. <tr class="judul" style="font-weight: 800;">
22. <th>Nama File</th>
23. </tr>
24. </thead>
25. <tbody>
26. {% for idx in range(0,len) %}
27. <tr>
28. <td><a style='color:black;' href="/view/{{
    data['listFile'][idx] }}">{{ data["listFile"][idx] }}</a></td>
29. </tr>
30. {% endfor %}
31. </tbody>
```

```
32. </table>
33. </div>
34. </div>
35. </div>
36. </div>
37. </section>
38. {% endblock %}
```

Kode Sumber 4.25. Implementasi antarmuka halaman utama



Subjek	Prediksi Ekspresi	Akurasi Prediksi
face1	positif	0.824516
face2	positif	0.946389
face3	positif	0.820227
face4	positif	0.949503
face5	positif	0.903574
face6	negatif	0.747869
face7	positif	0.973228
face8	positif	0.875653
face9	positif	0.823857
face10	negatif	0.690072

Gambar 4.3.2. Implementasi antarmuka halaman penampilan

```
1.  %%writefile templates/view.html
2.  {% extends "base.html" %}
3.
4.  {% block content %}
5.  <section class="slider-area service-area section-padding2">
6.  <div class="single-slider container">
7.  <div class="row d-flex justify-content-center">
8.  <div class="row" style="justify-content: center;">
9.  <div class="col-lg-12">
10. <div class="section-tittle text-center  slider-active">
11. <h2 data-animation="fadeInUp" data-
    delay=".4s">{{filename}}</h2>
12. </div>
13. </div>
14. {% if ext != 'mp4' %}
15. <img src="{{ data }}" alt="Gambar">
16. <div class="table-responsive">
17. <table id="example" class="table table-striped table-bordered
    dt-responsive" cellspacing="0" width="100%">
18. <thead>
19. <tr class="judul" style="font-weight: 800;">
20. <th>Subjek</th>
21. <th>Prediksi Ekspresi</th>
22. <th>Akurasi Prediksi</th>
23. </tr>
24. </thead>
25. <tbody>
26. {% for a_faces in range(0,len_faces) %}
27. <tr>
28. <td>{{faces[a_faces][11]}}</td>
29. <td>{{faces[a_faces][7]}}</td>
30. <td>{{faces[a_faces][8]}}</td>
31. </tr>
32. {% endfor %}
33. </tbody>
```

```
34. </table>
35. </div>
36. {% else %}
37. <video width="640" height="480" controls>
38. <source src="{{data}}" type="video/mp4">
39. Your browser does not support the video tag.
40. </video>
41. {% endif %}
42. </div>
43. </div>
44. </div>
45. </section>
46. {% endblock %}
```

Kode Sumber 4.26. Implementasi antarmuka halaman penampilan

Gambar 4.3.3. Implementasi antarmuka halaman pengunggahan

```
1.  %%writefile templates/upload.html
2.  {% extends "base.html" %}
3.
4.  {% block content %}
5.  <section class="slider-area service-area section-
    padding2">
6.  <div class="single-slider container">
7.  <h3>Upload File</h3>
8.  <form action="" enctype="multipart/form-data"
    method="post">
9.  <input type="file" name="file">
10. <input type="submit" name="" value="upload gambar">
11. </form>
12. </div>
13. </div>
14. {% endblock %}
```

Kode Sumber 4.27. Implementasi antarmuka halaman pengunggahan

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 *Google Collaboratory*. Sistem operasi yang digunakan adalah Ubuntu 18.04.3 LTS. *Cloud* yang digunakan memiliki spesifikasi Intel(R) Xeon(R) dengan kecepatan 2,00 GHz, *Random Access Memory* (RAM) sebesar 13 GB, dan mempunyai *Graphics Processing Unit* (GPU) yaitu NVIDIA Tesla P100-PCIE-16GB sebesar 16 GB. Serta database server *cloud* dengan sistem operasi yang digunakan adalah Ubuntu 18.04.3 LTS. Memiliki spesifikasi Dual Core Intel Pentium (Xeon) family dengan kecepatan 2,00 GHz, *Random Access Memory* (RAM) sebesar 2 GB, dan *Hardisk Drive* (HDD) sebesar 20 GB. Pada sisi perangkat lunak, uji coba pada tugas akhir ini dilakukan dengan menggunakan bahasa pemograman Python 3.6 dilengkapi dengan *library* antara lain Keras, Tensorflow, OpenCV, Numpy, Matplotlib, Scikit-learn, Scikit-image, dan SciPy.

5.2 Deskripsi Dataset

Perancangan data merupakan proses menentukan isi dan pengaturan data yang dibutuhkan untuk mendukung berbagai rancangan sistem yang akan dilakukan. Pada bagian ini, akan dilakukan 3 jenis uji coba data didasarkan pada tujuan dari masing masing perancangan data.

5.2.1 Dataset Diolah Secara Mandiri

Proses melakukan pelabelan terhadap data pada data diolah secara mandiri dengan kemampuan penulis dalam melakukan pemilahan data menjadi 2 kelas. Jumlah data gambar setelah proses pelabelan pada kelas positif berjumlah 670 citra dan pada kelas

negatif berjumlah 400 citra. Setelah melakukan proses pelabelan, data tersebut akan dipisah dengan persentase 80% untuk data latih dan 20% untuk data uji. Dengan jumlah data latih sebanyak 856 citra dan data uji sebanyak 214. Spesifikasi lengkap data diolah secara mandiri dapat dilihat pada Tabel 3.2. Tabel spesifikasi data diolah secara mandiri

5.2.2 Penentuan Kelompok Ekspresi

Pada tahap sebelumnya telah disebar tautan untuk melakukan survei penentuan kelompok data ekspresi wajah. Terdapat 49 responden yang telah memberikan subjektifnya terhadap ekspresi wajah yang disediakan. Hasil dari survei dapat dilihat pada Tabel 5.1. Hasil survei penentuan kelompok ekspresi.

Tabel 5.1. Hasil survei penentuan kelompok ekspresi

Inisial Responden	Marah	Netral	Jijik	Takut	Senang	Sedih	Terkejut
FAP	-	-	-	-	+	-	+
RSR	-	-	-	-	+	-	+
JS	-	-	-	-	+	-	-
SD	-	-	-	-	+	-	+
JN	-	-	-	-	+	-	+
FN	-	-	-	-	+	-	+
AAS	-	-	-	-	+	-	+
AY	-	-	-	-	+	-	+
DAN	+	+	+	-	-	-	-
VAL	-	-	-	-	+	-	-
MFZ	-	-	-	-	+	-	+

BPA	-	+	-	-	+	-	+
RTZ	+	-	+	+	-	+	-
YY	+	-	-	-	+	-	+
KSP	-	-	-	-	+	-	-
SM	-	+	-	+	+	-	+
AW	+	-	+	-	+	-	+
MKW	+	-	+	+	+	-	+
AP	-	-	-	-	+	-	+
DS	-	-	-	-	+	-	-
SM	+	-	-	-	+	-	+
AP	-	-	-	-	+	-	+
RF	-	-	-	-	+	-	+
ANA	-	-	-	-	+	-	+
DS	-	-	-	-	+	-	-
AN	+	-	-	+	+	-	+
DKI	+	-	-	-	+	-	-
KZL	-	-	-	-	+	-	+
AF	-	-	-	-	+	-	+
TZS	-	-	-	-	+	-	+
AP	+	-	-	-	+	-	+
YA	-	-	+	-	+	-	+
RAW	-	-	-	-	+	-	+
AIM	-	-	-	-	+	-	-

PV	-	-	-	-	+	-	+
PM	-	-	-	-	+	-	+
AF	-	-	-	-	+	-	+
YN	-	-	+	-	+	-	+
FF	-	-	-	-	+	-	+
VZ	-	-	-	-	+	-	+
RAW	-	-	-	-	+	-	+
SZK	-	-	-	-	+	-	+
NS	+	+	+	+	+	+	+
SL	-	+	-	-	+	-	+
AB	-	-	-	-	+	-	-
AAN	-	-	-	-	+	-	+
ARR	-	-	-	-	+	-	+
DT	+	-	-	+	+	-	+
AA	-	-	-	-	+	-	+
Ekspresi Positif	22,4 %	10,2 %	14,3 %	12,2 %	**95,9 %**	4,1%	**79,6 %**
Ekspresi Negatif	**77,6 %**	**89,8 %**	**85,7 %**	**87,8 %**	4,1%	**95,9 %**	20,4 %

Pada tabel diatas, didapatkan bahwa hasil survei kelompok ekspresi wajah positif berupa senang dan terkejut, sedangkan kelompok ekspresi wajah negative berupa marah, netral, jijik, takut, dan sedih.

5.2.3 Dataset CK+

Jumlah data gambar setelah proses diatas, pada kelas *anger* (marah) berjumlah 135 citra, kelas *netral* (netral) berjumlah 54 citra, kelas *disgust* (jijik) berjumlah 177 citra, kelas *fear* (takut) berjumlah 75 citra, kelas *happy* (senang) berjumlah 207 citra, kelas *sadness* (sedih) berjumlah 84 citra dan kelas *surprise* (terkejut) berjumlah 249 citra. Setelah melakukan proses pelabelan dan penentuan ROI, data tersebut akan dipisah dengan persentase 80% untuk data latih dan 20% untuk data uji. Dengan jumlah data latih sebanyak 784 citra dan data uji sebanyak 197. Spesifikasi lengkap data diolah secara mandiri dapat dilihat pada Tabel 3.3. Tabel spesifikasi data CK+.

5.2.4 *Ground Truth* Deteksi Wajah

Data yang digunakan sebagai masukan awal dari tahap perancangan data untuk deteksi wajah adalah beberapa *frame* dari rekaman mata kuliah mahasiswa S2 Tahun Ajaran 2019/2020. Data ini masih belum terlabeli dan teranotasi. Proses selanjutnya akan membuat konfigurasi yang berisi informasi label dan anotasi dari kumpulan frame tersebut. Spesifikasi lengkap mengenai data untuk deteksi wajah dapat dilihat pada Tabel 3.4. Spesifikasi data untuk deteksi wajah.

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, skenario uji coba dibagi menjadi 2 dan 1 hasil uji coba yaitu:

1. Skenario Uji Coba pada dataset diolah secara mandiri
2. Skenario Uji Coba pada dataset *The Extended Cohn-Kanade Dataset* (CK+)

5.3.1 Skenario Uji Coba Pada Data Diolah Secara Mandiri

Pada skenario ini, terdapat 3 macam skenario uji coba dan semuanya akan dicoba pada arsitektur yang telah dirancang. Data yang digunakan sebagai masukan awal adalah data yang telah digunakan pada saat mata kuliah mahasiswa S2 Tahun Ajaran 2019/2020. Dengan terdiri dari 2 kelas yang dilabeli secara mandiri yaitu, kelas positif dan kelas negatif.

Tabel 5.2. Spesifikasi awal parameter arsitektur

Keterangan	**Parameter**
Jumlah *epoch*	25
Ukuran *batch*	52
Learning Rate	0,001, 0,0001, 0,00001
Optimizer	Adam, RMSprop, Adagrad
Loss Compile	*Crossentropy*
Test Percentage	*20%*

Skenario uji coba yang akan dilakukan yaitu:

1. Uji Coba model VGG16
2. Uji Coba model Resnet50
3. Uji Coba model Senet50

Setiap skenario nantinya akan diuji pada data uji. Pada setiap skenario uji coba akan ditetapkan nilai parameter yang dapat meningkatkan kinerja arsitektur.

5.3.1.1 Skenario Uji Coba Model VGG16

Pada skenario ini, terdapat 3 macam skenario uji coba dengan perbedaan jenis *optimizer* yang digunakan. Pada skenario ini, arsitektur VGG16 dengan *optimizer* Adam *learning rate* sebesar 0,0001 mendapatkan hasil terbaik. Pertama menggunakan pengoptimalan Adam dapat dilihat pada Tabel 5.3. Evaluasi model VGG16 Adam.

Tabel 5.3. Evaluasi model VGG16 Adam

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	591 detik	96%	96%	96%	96%
0,0001	**597 detik**	**98%**	**98%**	**98%**	**98%**
0,001	594 detik	97%	97%	97%	97%

Kedua menggunakan pengoptimalan RMSprop dapat dilihat pada Tabel 5.4. Evaluasi model VGG16 RMSprop.

Tabel 5.4. Evaluasi model VGG16 RMSprop

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	597 detik	97%	95%	96%	95%
0,0001	597 detik	97%	96%	97%	96%
0,001	**597 detik**	**98%**	**97%**	**97%**	**97%**

Ketiga menggunakan pengoptimalan Adagrad dapat dilihat pada Tabel 5.5. Evaluasi model VGG16 Adagrad.

Tabel 5.5. Evaluasi model VGG16 Adagrad

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	594 detik	80%	80%	80%	80%
0,0001	591 detik	96%	95%	96%	95%
0,001	**594 detik**	**97%**	**96%**	**97%**	**96%**

5.3.1.2 Skenario Uji Coba Model Resnet50

Pada skenario ini, terdapat 3 macam skenario uji coba dengan perbedaan jenis *optimizer* yang digunakan. Pada skenario

ini, arsitektur Resnet50 dengan *optimizer* Adam *learning rate* sebesar 0,001 mendapatkan hasil terbaik. Pertama menggunakan pengoptimalan Adam dapat dilihat pada Tabel 5.6. Evaluasi model Resnet50 Adam.

Tabel 5.6. Evaluasi model Resnet50 Adam

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	723 detik	47%	55%	47%	48%
0,0001	663 detik	85%	88%	85%	84%
0,001	**663 detik**	**96%**	**96%**	**96%**	**96%**

Kedua menggunakan pengoptimalan RMSprop dapat dilihat pada Tabel 5.7. Evaluasi model Resnet50 RMSprop.

Tabel 5.7. Evaluasi model Resnet50 RMSprop

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	663 detik	80%	80%	80%	79%
0,0001	660 detik	82%	86%	82%	80%
0,001	**663 detik**	**95%**	**96%**	**95%**	**95%**

Ketiga menggunakan pengoptimalan Adagrad dapat dilihat pada Tabel 5.8. Evaluasi model Resnet50 Adagrad.

Tabel 5.8. Evaluasi model Resnet50 Adagrad

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	660 detik	59%	53%	59%	54%
0,0001	663 detik	68%	67%	68%	61%
0,001	**660 detik**	**87%**	**88%**	**87%**	**87%**

5.3.1.3 Skenario Uji Coba Model Senet50

Pada skenario ini, terdapat 3 macam skenario uji coba dengan perbedaan jenis *optimizer* yang digunakan. Pada skenario ini, arsitektur Senet50 dengan *optimizer* RMSprop *learning rate* sebesar 0,001 mendapatkan hasil terbaik. Pertama menggunakan pengoptimalan Adam dapat dilihat pada Tabel 5.9. Evaluasi model Senet50 Adam.

Tabel 5.9. Evaluasi model Senet50 Adam

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	837 detik	68%	67%	68%	68%
0,0001	792 detik	83%	85%	83%	81%
0,001	**789 detik**	**89%**	**89%**	**89%**	**88%**

Kedua menggunakan pengoptimalan RMSprop dapat dilihat pada Tabel 5.10. Evaluasi model Senet50 RMSprop.

Tabel 5.10. Evaluasi model Senet50 RMSprop

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	780 detik	61%	61%	61%	61%
0,0001	771 detik	89%	86%	83%	80%
0,001	**768 detik**	**91%**	**91%**	**91%**	**91%**

Ketiga menggunakan pengoptimalan Adagrad dapat dilihat pada Tabel 5.11. Evaluasi model Senet50 Adagrad.

Tabel 5.11. Evaluasi model Senet50 Adagrad

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	768 detik	66%	78%	66%	67%
0,0001	765 detik	74%	79%	74%	69%
0,001	**768 detik**	**74%**	**80%**	**74%**	**69%**

5.3.2 Skenario Uji Coba Pada Data *The Extended Cohn-Kanade Dataset* (CK+)

Pada skenario ini, terdapat 3 macam skenario uji coba dan semuanya akan dicoba pada arsitektur yang telah dirancang.

Tabel 5.12 Spesifikasi awal parameter arsitektur

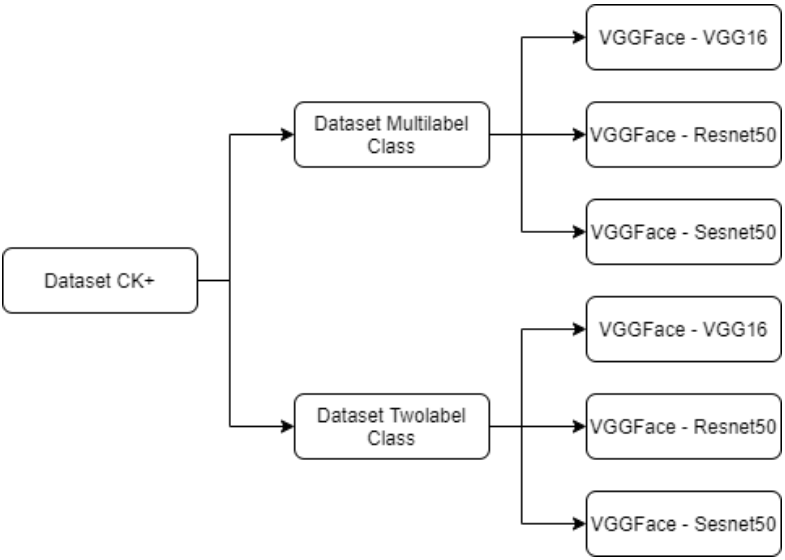
Keterangan	**Parameter**
Jumlah *epoch*	25
Ukuran *batch*	52
Learning Rate	0,001, 0,0001, 0,00001
Optimizer	Adam, RMSprop, Adagrad
Loss Compile	*Crossentropy*
Test Percentage	*20%*

Skenario uji coba yang akan dilakukan yaitu:

1. Uji Coba model VGG16 dengan 7 kelas
2. Uji Coba model Resnet50 dengan 7 kelas
3. Uji Coba model Sesnet50 dengan 7 kelas
4. Uji Coba model VGG16 dengan 2 kelas
5. Uji Coba model Resnet50 dengan 2 kelas
6. Uji Coba model Sesnet50 dengan 2 kelas

Setiap skenario nantinya akan diuji pada data uji. Pada setiap skenario uji coba akan ditetapkan nilai parameter yang dapat meningkatkan kinerja arsitektur. Diagram skenario menggunakan

data (CK+) dapat dilihat pada Gambar 5.3.1. Diagram skenario dataset CK+ Gambar 5.3.1.



Gambar 5.3.1. Diagram skenario dataset CK+

Data yang digunakan sebagai masukan awal dari tahap perancangan data CK+ adalah data yang digunakan pada beberapa referensi mengenai *facial expression* [7]. Untuk data CK+ 7 kelas terdiri atas kelas *anger* (marah), kelas *netral* (netral), kelas *disgust* (jijik), kelas *fear* (takut), kelas *happy* (senang), kelas *sadness* (sedih) dan kelas *surprise* (terkejut). Untuk data CK+ 2 kelas dengan pengelompokan ekspresi positif terdiri dari kelas senang, netral, terkejut, dan takut. Sedangkan untuk kelas negatif terdiri dari kelas marah, sedih dan jijik.

5.3.2.1 Skenario Uji Coba Model VGG16 - 7 Kelas

Pada skenario ini, terdapat 3 macam skenario uji coba dengan perbedaan jenis *optimizer* yang digunakan. Pada skenario ini, arsitektur VGG16 dengan *optimizer* Adam *learning rate* sebesar 0,001 mendapatkan hasil terbaik. Hasil menggunakan

pengoptimalan Adam dapat dilihat pada Tabel 5.13. Evaluasi model 7 kelas VGG16 Adam.

Tabel 5.13. Evaluasi model 7 kelas VGG16 Adam

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	615 detik	91%	92%	91%	91%
0,0001	576 detik	97%	97%	97%	97%
0,001	**570 detik**	**97%**	**97%**	**97%**	**97%**

Kedua menggunakan pengoptimalan RMSprop dapat dilihat pada Tabel 5.14. Evaluasi model 7 kelas VGG16 RMSprop.

Tabel 5.14. Evaluasi model 7 kelas VGG16 RMSprop

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	597 detik	93%	94%	93%	93%
0,0001	**570 detik**	**96%**	**97%**	**96%**	**96%**
0,001	567 detik	93%	95%	93%	94%

Ketiga menggunakan pengoptimalan Adagrad dapat dilihat pada Tabel 5.15. Evaluasi model 7 kelas VGG16 Adagrad.

Tabel 5.15. Evaluasi model 7 kelas VGG16 Adagrad

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	567 detik	45%	48%	45%	45%
0,0001	567 detik	89%	90%	89%	89%
0,001	**567 detik**	**97%**	**96%**	**97%**	**97%**

5.3.2.2 Skenario Uji Coba Model Resnet50 – 7 Kelas

Pada skenario ini, terdapat 3 macam skenario uji coba dengan perbedaan jenis *optimizer* yang digunakan. Pada skenario ini, arsitektur Resnet50 dengan *optimizer* Adam *learning rate* sebesar 0,001 mendapatkan hasil terbaik. Hasil menggunakan pengoptimalan Adam dapat dilihat pada Tabel 5.16. Evaluasi model 7 kelas Resnet50 Adam.

Tabel 5.16. Evaluasi model 7 kelas Resnet50 Adam

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	654 detik	31%	30%	31%	28%
0,0001	594 detik	74%	75%	74%	72%
0,001	**603 detik**	**90%**	**92%**	**90%**	**90%**

Kedua menggunakan pengoptimalan RMSprop dapat dilihat pada Tabel 5.17. Evaluasi model 7 kelas Resnet50 RMSprop.

Tabel 5.17. Evaluasi model 7 kelas Resnet50 RMSprop

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	609 detik	26%	24%	26%	25%
0,0001	591 detik	80%	82%	80%	79%
0,001	**588 detik**	**89%**	**92%**	**89%**	**89%**

Ketiga menggunakan pengoptimalan Adagrad dapat dilihat pada Tabel 5.18. Evaluasi model 7 kelas Resnet50 Adagrad.

Tabel 5.18. Evaluasi model 7 kelas Resnet50 Adagrad

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	585 detik	15%	17%	15%	15%
0,0001	594 detik	30%	34%	30%	30%
0,001	**582 detik**	**80%**	**81%**	**80%**	**78%**

5.3.2.3 Skenario Uji Coba Model Senet50 – 7 Kelas

Pada skenario ini, terdapat 3 macam skenario uji coba dengan perbedaan jenis *optimizer* yang digunakan. Pada skenario ini, arsitektur Resnet50 dengan *optimizer* Adam *learning rate* sebesar 0,001 mendapatkan hasil terbaik. Hasil menggunakan pengoptimalan Adam dapat dilihat pada Tabel 5.19. Evaluasi model 7 kelas Senet50 Adam.

Tabel 5.19. Evaluasi model 7 kelas Senet50 Adam

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	957 detik	26%	28%	26%	26%
0,0001	900 detik	80%	81%	80%	79%
0,001	**894 detik**	**87%**	**90%**	**87%**	**87%**

Kedua menggunakan pengoptimalan RMSprop dapat dilihat pada Tabel 5.20. Evaluasi model 7 kelas Senet50 RMSprop.

Tabel 5.20. Evaluasi model 7 kelas Senet50 RMSprop

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	891 detik	21%	25%	21%	22%
0,0001	**900 detik**	**83%**	**85%**	**85%**	**85%**
0,001	897 detik	72%	87%	72%	73%

Ketiga menggunakan pengoptimalan Adagrad dapat dilihat pada Tabel 5.21. Evaluasi model 7 kelas Senet50 Adagrad.

Tabel 5.21. Evaluasi model 7 kelas Senet50 Adagrad

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	891 detik	18%	24%	18%	29%
0,0001	897 detik	28%	28%	28%	28%
0,001	**894 detik**	**73%**	**74%**	**73%**	**71%**

5.3.2.4 Skenario Uji Coba Model VGG16 - 2 Kelas

Pada skenario ini, terdapat 3 macam skenario uji coba dengan perbedaan jenis *optimizer* yang digunakan. Pada skenario ini, arsitektur VGG16 dengan *optimizer* Adam *learning rate* sebesar 0,0001 mendapatkan hasil terbaik. Hasil menggunakan pengoptimalan Adam dapat dilihat pada Tabel 5.22. Evaluasi model 2 kelas VGG16 Adam.

Tabel 5.22. Evaluasi model 2 kelas VGG16 Adam

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	990 detik	97%	96%	97%	96%
0,0001	**957 detik**	**98%**	**98%**	**97%**	**98%**
0,001	945 detik	96%	95%	94%	94%

Kedua menggunakan pengoptimalan RMSprop dapat dilihat pada Tabel 5.23. Evaluasi model 2 kelas VGG16 RMSprop.

Tabel 5.23. Evaluasi model 2 kelas VGG16 RMSprop

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	936 detik	97%	97%	97%	97%
0,0001	**945 detik**	**98%**	**98%**	**97%**	**97%**
0,001	936 detik	97%	97%	97%	97%

Ketiga menggunakan pengoptimalan Adagrad dapat dilihat pada Tabel 5.24. Evaluasi model 2 kelas VGG16 Adagrad.

Tabel 5.24. Evaluasi model 2 kelas VGG16 Adagrad

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	939 detik	85%	85%	85%	85%
0,0001	930 detik	95%	95%	95%	95%
0,001	**945 detik**	**97%**	**98%**	**97%**	**96%**

5.3.2.5 Skenario Uji Coba Model Resnet50 – 2 Kelas

Pada skenario ini, terdapat 3 macam skenario uji coba dengan perbedaan jenis *optimizer* yang digunakan. Pada skenario ini, arsitektur Resnet50 dengan *optimizer* RMSprop *learning rate* sebesar 0,001 mendapatkan hasil terbaik. Pertama menggunakan pengoptimalan Adam dapat dilihat pada Tabel 5.25. Evaluasi model 2 kelas Resnet50 Adam.

Tabel 5.25. Evaluasi model 2 kelas Resnet50 Adam

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	954 detik	55%	53%	55%	53%
0,0001	894 detik	92%	92%	92%	92%
0,001	**894 detik**	**96%**	**96%**	**96%**	**96%**

Kedua menggunakan pengoptimalan RMSprop dapat dilihat pada Tabel 5.26. Evaluasi model 2 kelas Resnet50 RMSprop.

Tabel 5.26. Evaluasi model 2 kelas Resnet50 RMSprop

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	897 detik	68%	67%	68%	67%
0,0001	900 detik	88%	89%	88%	88%
0,001	**891 detik**	**98%**	**97%**	**98%**	**97%**

Ketiga menggunakan pengoptimalan Adagrad dapat dilihat pada Tabel 5.27. Evaluasi model 2 kelas Resnet50 Adagrad.

Tabel 5.27. Evaluasi model 2 kelas Resnet50 Adagrad

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	891 detik	48%	53%	48%	47%
0,0001	897 detik	70%	69%	67%	67%
0,001	**891 detik**	**88%**	**90%**	**88%**	**88%**

5.3.2.6 Skenario Uji Coba Model Senet50 – 2 Kelas

Pada skenario ini, terdapat 3 macam skenario uji coba dengan perbedaan jenis *optimizer* yang digunakan. Pada skenario ini, arsitektur Senet50 dengan *optimizer* Adam *learning rate* sebesar 0,001 mendapatkan hasil terbaik. Pertama menggunakan pengoptimalan Adam dapat dilihat pada Tabel 5.28. Evaluasi model 2 kelas Senet50 Adam.

Tabel 5.28. Evaluasi model 2 kelas Senet50 Adam

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	714 detik	51%	52%	51%	51%
0,0001	651 detik	81%	81%	81%	81%
0,001	**651 detik**	**93%**	**93%**	**93%**	**93%**

Kedua menggunakan pengoptimalan RMSprop dapat dilihat pada Tabel 5.29. Evaluasi model 2 kelas Senet50 RMSprop.

Tabel 5.29. Evaluasi model 2 kelas Senet50 RMSprop

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	651 detik	62%	63%	62%	62%
0,0001	**654 detik**	**90%**	**90%**	**90%**	**90%**
0,001	654 detik	87%	90%	87%	87%

Ketiga menggunakan pengoptimalan Adagrad dapat dilihat pada Tabel 5.30. Evaluasi model 2 kelas Senet50 Adagrad.

Tabel 5.30. Evaluasi model 2 kelas Senet50 Adagrad

Learning rate	**Lama waktu pelatihan**	**Akurasi**	***Precision***	***Recall***	***F1-score***
0,00001	648 detik	54%	47%	54%	47%
0,0001	648 detik	61%	90%	61%	61%
0,001	**648 detik**	**84%**	**84%**	**84%**	**84%**

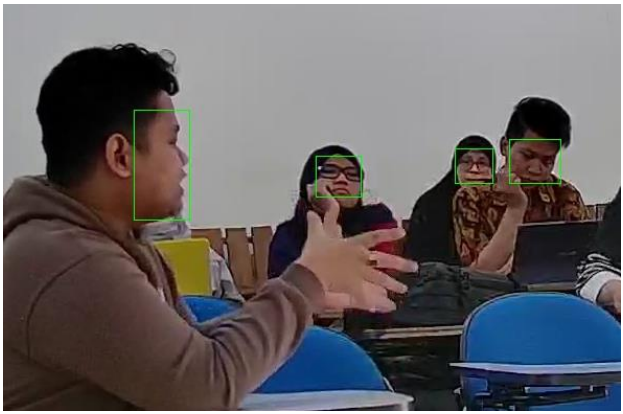
5.3.3 Skenario Uji Coba Pada Deteksi Wajah

Pada skenario ini, terdapat 4 macam skenario uji coba berdasarkan *confidence value* yang telah ditetapkan. *Confidence value* yang akan digunakan adalah 60%, 70%, 80%, dan 90%.

Setiap skenario nantinya akan diuji pada kumpulan *frame* berbasis video rekaman Mata Kuliah mahasiswa S2 Tahun Ajaran 2019/2020. Uji coba digunakan untuk mengukur *confidence value* yang sesuai dengan data yang digunakan. Berikut contoh hasil deteksi wajah pada rekaman kedua, *frame* ke-7 pada Tabel 5.32. Evaluasi pada rekaman kedua dapat dilihat pada gambar dibawah ini.

Gambar 5.3.2. Potongan hasil deteksi wajah (60%) rekaman kedua frame ke-7

Pada Gambar 5.3.2, terdapat kesalahan deteksi berupa telunjuk jari. Terdapat beberapa analisa yang dapat menyebabkan kesalahan deteksi ini. Pertama, telunjuk jari memiliki *color feature* yang menyerupai warna dari wajah. Kedua, tekstur wajah jika dilakukan *block normalization* akan menyerupai jari telunjuk. Dan analisa terakhir karena jari telunjuk tersebut memiliki *image gradient* hitam yang menyerupai sepasang mata pada wajah manusia.



Gambar 5.3.3. Potongan hasil deteksi wajah (70%) rekaman kedua *frame* ke-7

Pada Gambar 5.3.2, hasil deteksi dengan *confidence value* 60% didapatkan *bounding box* pada area telunjuk mahasiswa berjaket coklat. Sedangkan pada Gambar 5.3.3, hasil dengan *confidence value* 70% didapatkan *bounding box* yang sesuai. Hasil evaluasi pada rekaman pertama dapat dilihat pada Tabel 5.31. Evaluasi pada rekaman pertama.

Tabel 5.31. Evaluasi pada rekaman pertama

Frame	**Jumlah Wajah**	**Jumlah Wajah Terdeteksi**			
		Confidence Value			
		60%	**70%**	**80%**	**90%**
1	11	11	11	11	11
2	9	8	8	8	6
3	10	9	9	9	8
4	10	9	7	7	6
5	11	11	11	11	11
6	12	11	11	10	9
7	9	10	9	9	9
8	11	9	9	7	7
9	10	10	10	9	8

10	12	12	12	11	10
11	11	11	11	10	9
12	12	11	11	11	9
13	10	8	8	6	6
14	9	10	9	8	6
15	11	8	7	7	6
16	8	8	7	6	6
17	11	9	9	9	6
18	11	9	8	7	6
19	8	9	8	8	7
20	9	9	9	8	5

Hasil Evaluasi pada rekaman kedua dapat dilihat pada **Tabel 5.32. Evaluasi pada rekaman kedua**.

Tabel 5.32. Evaluasi pada rekaman kedua

Frame	**Jumlah Wajah**	**Jumlah Wajah Terdeteksi**			
		Confidence Value			
		60%	**70%**	**80%**	**90%**
1	9	9	9	9	6
2	9	9	9	9	8
3	7	9	7	6	4
4	9	10	9	7	5
5	7	8	7	5	5
6	10	11	10	10	8
7	8	10	8	8	6
8	10	11	10	9	8
9	11	11	11	11	10
10	11	11	11	10	6
11	9	10	9	9	8
12	9	10	9	9	8
13	10	10	10	10	10
14	10	10	10	8	5
15	11	11	11	10	6

16	9	9	9	8	7
17	9	9	9	8	4
18	11	11	11	11	11
19	11	11	11	11	11
20	9	9	9	9	8

Hasil Evaluasi pada rekaman ketiga dapat dilihat pada Tabel 5.33. Evaluasi pada rekaman ketiga**.**

Tabel 5.33. Evaluasi pada rekaman ketiga

Frame	**Jumlah Wajah**	**Jumlah Wajah Terdeteksi**			
		Confidence Value			
		60%	**70%**	**80%**	**90%**
1	10	10	10	9	8
2	10	10	10	10	8
3	10	9	9	8	7
4	9	7	7	7	5
5	9	9	9	8	8
6	9	9	9	8	8
7	8	8	8	8	6
8	7	8	7	6	6
9	8	9	8	6	5
10	7	8	7	7	4
11	9	8	6	6	5
12	9	9	8	6	6
13	10	10	10	9	5
14	9	9	9	8	6
15	8	9	8	8	6
16	10	8	8	8	8
17	6	6	5	4	3
18	9	8	6	5	5
19	8	8	8	7	7
20	12	12	11	11	10

Dilakukan juga uji coba terhadap model deteksi wajah yang telah dibuat pada tahap pelatihan deteksi wajah. Uji coba dilakukan dengan memberikan *IoU threshold* sebesar 50%, 60%, 70% dan 80%. Hasil uji coba dapat dilihat pada Tabel 5.34. Uji coba dengan IoU.

Tabel 5.34. Uji coba dengan IoU

Nilai	***IoU Threshold***			
	50%	**60%**	**70%**	**80%**
TP	845	827	713	405
FP	10	28	142	450
FN	1	19	133	441
Precision	98,83%	96,73%	83,39%	47,37%
Recall	99,88%	97,75%	84,28%	47,87%
F1-Score	99,35%	97,24%	83,83%	47,62%
AP	99,72%	97,17%	77,95%	28,46%

Nilai TP merupakan nilai *True Positive*, nilai FP merupakan nilai *False Positive,* nilai FN merupakan nilai *False Negative*

5.3.4 Skenario Performa Waktu Eksekusi Proses

Pada skenario ini akan dilakukan uji coba terhadap waktu eksekusi seluruh proses. Proses yang diuji coba terdiri dari proses deteksi wajah, dan proses pengenalan ekspresi wajah terhadap sebuah *frame* pada data video. *Frame* yang digunakan sebanyak 20 *frame* untuk 3 video rekaman. Skenario uji coba ini digunakan untuk mengukur performa yang didapatkan dalam melakukan implementasi proses. Performa proses pada rekaman pertama dapat dilihat pada Tabel 5.35. Performa proses pada rekaman pertama.

Tabel 5.35. Performa proses pada rekaman pertama

Frame	**Proses Deteksi Wajah (ms)**	**Proses Pengenalan Ekspresi Wajah (ms)**	**Proses Pelacakan Objek (ms)**
1	638,608	9,040	8,110
2	282,762	9,081	8,011
3	283,096	8,600	7,899
4	281,656	8,831	7,022
5	283,975	8,571	7,219
6	283,180	8,692	7,482
7	282,285	8,846	7,802
8	282,253	9,111	7,085
9	285,136	8,966	7,723
10	280,281	8,614	7,729
11	283,638	8,805	7,465
12	280,503	8,644	6,992
13	284,460	8,680	7,640
14	281,792	8,841	7,303
15	282,258	8,647	7,293
16	281,935	9,164	7,893
17	282,398	8,539	7,775
18	281,613	8,849	7,594
19	284,055	9,334	8,143
20	283,939	8,980	7,955
Rata-rata	300,491	8,842	7,607

Performa proses pada rekaman kedua dapat dilihat pada Tabel 5.36. Performa proses pada rekaman kedua.

Tabel 5.36. Performa proses pada rekaman kedua

Frame	**Proses Deteksi Wajah (ms)**	**Proses Pengenalan Ekspresi Wajah (ms)**	**Proses Pelacakan Objek (ms)**
1	610,850	8,562	10,510
2	279,159	8,500	10,586
3	275,947	8,370	10,278
4	279,281	9,112	10,528
5	277,587	8,827	10,542
6	276,844	8,487	10,712
7	276,080	9,145	10,420
8	279,682	8,483	9,162
9	276,004	9,685	11,977
10	281,787	8,631	10,610
11	275,803	8,611	10,362
12	279,530	8,544	10,780
13	276,747	12,021	10,599
14	278,654	8,730	10,583
15	277,256	10,152	10,520
16	277,884	8,530	10,623
17	280,413	8,638	10,493
18	278,258	8,582	10,430
19	280,115	8,781	10,528
20	276,330	8,644	10,535
Rata-rata	294,711	8,952	10,539

Performa proses pada rekaman ketiga dapat dilihat pada Tabel 5.37. Performa proses pada rekaman ketiga.

Tabel 5.37. Performa proses pada rekaman ketiga

Frame	**Proses Deteksi Wajah (ms)**	**Proses Pengenalan Ekspresi Wajah (ms)**	**Proses Pelacakan Objek (ms)**
1	694,315	8,877	10,697
2	278,177	8,720	11,341
3	278,579	8,906	10,710
4	279,375	8,636	10,486
5	276,580	8,677	10,479
6	278,054	8,892	10,750
7	279,545	8,970	2,286
8	281,545	8,646	10,924
9	278,784	8,753	10,997
10	277,934	8,654	10,830
11	279,172	8,640	10,541
12	278,350	8,623	10,458
13	278,984	8,652	10,534
14	278,027	8,702	11,192
15	278,629	8,932	10,884
16	286,149	8,495	10,956
17	281,843	8,625	11,396
18	277,565	8,536	11,007
19	279,543	9,631	10,967
20	278,862	8,544	10,186
Rata-rata	300,001	8,755	10,381

Dari ketiga performa diatas didapatkan waktu proses deteksi wajah *frame* pertama selama ± 600 ms, waktu rata-rata proses deteksi wajah selama ± 300 ms, waktu rata-rata proses pengenalan ekspresi wajah selama ± 8 ms, dan waktu rata-rata proses pelacakan objek selama ± 10ms. Dalam uji coba performa ini,

perangkat keras yang digunakan adalah Intel(R) Xeon(R) dengan kecepatan 2.00 GHz, *Random Access Memory* (RAM) sebesar 13 GB, dan mempunyai *Graphics Processing Unit* (GPU) yaitu NVIDIA Tesla K80 sebesar 16 GB.

5.3.5 Evaluasi Hasil Sistem

Pada evaluasi hasil sistem, akan ditampilkan informasi mengenai hasil keluaran sistem yang telah diimplementasikan. Data video yang digunakan adalah salah satu rekaman Mata Kuliah mahasiswa S2 Tahun Ajaran 2019/2020 dengan spesifikasi dapat dilihat pada Tabel 5.38. Spesifikasi data evaluasi hasil sistem.

Tabel 5.38. Spesifikasi data evaluasi hasil sistem

Keterangan	**Spesifikasi**
Ukuran resolusi asli	1920 x 1080
Ekstensi	.mp4
FPS	20
Durasi	1 menit 40 detik
Ukuran file	8219 KB
Kanal warna	3 (RGB)

Dengan menggunakan 2 *frame* tiap detik untuk dilakukan pengenalan, didapatkan waktu eksekusi sistem selama 1 menit 44 detik. Dengan waktu eksekusi tersebut, video rekaman dibagi menjadi ± 200 *frame* untuk dilakukan pengenalan. Dalam seluruh *frame* terdapat 1793 deteksi wajah sesuai dengan *threshold* yang digunakan. Hasil rata-rata confidence, rata-rata akurasi pengenalan ekspresi dan jumlah *frame* tiap subjek dapat dilihat pada Tabel 5.39. Hasil evaluasi rekaman.

Tabel 5.39. Hasil evaluasi rekaman

Subjek	**Rata-rata *Confidence***	**Rata-rata Akurasi Pengenalan Ekspresi**	**Jumlah *Frame***
ID 0	0,944511	0,981097	200
ID 1	0,804464	0,86632	198
ID 2	0,857359	0,935497	198
ID 3	0,983343	0,943215	200
ID 4	0,852467	1	155
ID 5	0,870866	1	173
ID 6	0,952148	0,999832	176
ID 7	0,906006	0,997241	176
ID 8	0,834946	0,882492	29
ID 9	0,867269	0,993137	54
ID 10	0,832298	0,849512	130

Potongan *frame* dengan subjek – ID 0, ID 1, dan ID 2 dapat dilihat pada Gambar 5.3.4.

Gambar 5.3.4. Potongan *frame* dengan subjek

Rekap hasil pengenalan ekspresi dengan menggunakan *query* pada bagian Implementasi Tahap Rekap Hasil Pengenalan dapat dilihat pada Tabel 5.40. Rekap hasil sistem.

Tabel 5.40. Rekap hasil sistem

Subjek	Jumlah Ekspresi Negatif	Jumlah Ekspresi Positif	Persentase Ekspresi Positif
ID 0	7	193	97%
ID 1	25	173	87%
ID 2	67	131	66%
ID 3	66	134	67%
ID 4	0	155	100%
ID 5	0	173	100%
ID 6	0	176	100%
ID 7	1	175	99%
ID 8	8	21	72%
ID 9	0	54	100%
ID 10	25	105	81%

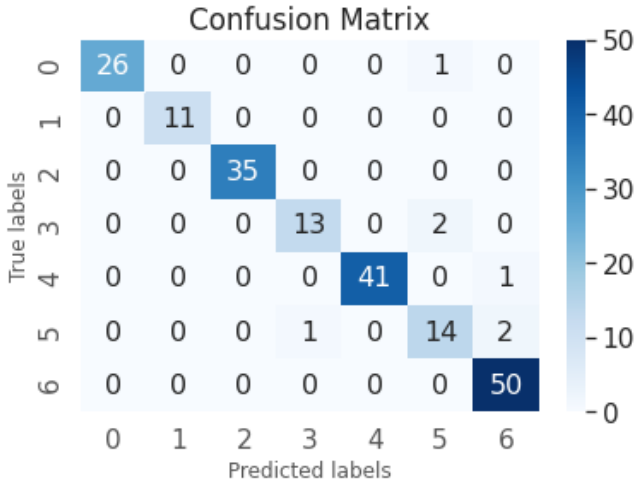
Untuk rekap hasil dengan menggunakan *win – loss chart* dapat dilihat pada Gambar 5.3.5. Keterangan hijau untuk ekspresi positif dan merah untuk ekspresi negatif.

Gambar 5.3.5. Rekap hasil setiap subjek

5.3.6 Evaluasi Kesalahan Klasifikasi

Pada evaluasi ini, terdapat beberapa kesalahan klasifikasi pada model yang telah dibangun. Dilakukan analisa pada setiap kesalahan klasifikasi yang ditemukan. Kesalahan klasifikasi pada tugas akhir ini terdapat pada model dengan dataset CK+ dengan 7 kelas.

Hasil klasifikasi pada data tes dapat dilihat pada Gambar 5.3.6. Dari hasil prediksi kelas oleh model yang dibangun, didapatkan persebaran hasil prediksi yang sesuai dengan kelas aslinya. Dari 197 data tes, hanya terdapat 7 citra yang tidak sesuai dengan kelas aslinya.



Gambar 5.3.6. Confusion matrix CK+ Adam 0,001

Berikut kode label yang terdapat pada Gambar 5.3.6. Confusion matrix CK+ Adam 0,001. 0 merupakan label marah, 1 merupakan label netral, 2 merupakan label jijik, 3 merupakan label takut, 4 merupakan label senang, 5 merupakan label sedih, dan 6 merupakan label terkejut.

Kesalahan prediksi terdapat pada prediksi kelas takut, sedih, dan terkejut. Sebagai contoh prediksi kelas sedih mendeteksi 2 gambar sebagai kelas takut. Hal ini dikarenakan terdapat kemiripan teksture gambar pada kedua kelas tersebut. Pada Gambar 5.3.7 terdapat teksture alis, mata dan lekukan pipi hampir sama, maka dari itu dianggap sebagai kelas yang sama.

Gambar 5.3.7. Ekspresi sedih dan takut

Selain itu, pada saat pengujian menggunakan model dengan 2 kelas, terdapat kesalahan klasifikasi terhadap kelompok ekspresi negatif. Pada Gambar 5.3.8, mahasiswa yang duduk paling kanan terdeteksi sebagai kelompok ekspresi positif. Mahasiswa tersebut sedang tertidur, namun prediksi dengan model dianggap sebagai kelas positif. Hal ini terjadi akibat area telapak tangan menjadi satu area yang akan diprediksi membuat terjadinya *noise* yang membuat kerja model tidak maksimal. Namun berdasarkan *confidence score* yang tertera, model tidak yakin dalam mengklasifikasikannya karena *confidence score* mendapatkan 81% sebagai ekspresi positif. Berbeda dengan prediksi lainnya yang mendapatkan *confidence score* mendekati 99%.

Gambar 5.3.8. Kesalahan klasifikasi negative

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas tentang kesimpulan yang didasari oleh hasil uji coba yang telah dilakukan terhadap *You Only Look Once* (YOLO) dan *Convolutional Neural Network* (CNN) pada bab sebelumnya. Kesimpulan nantinya sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut di masa depan.

6.1 Kesimpulan

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

1. Dengan menggunakan *framework* Darknet, dapat dilatih sebuah pendeteksi objek dengan kelas wajah. Dengan menggunakan *pretrained convolutional layer weights*, didapatkan pendeteksi yang dapat bekerja pada sistem monitoring.
2. Berikut Uji coba pada setiap model yang telah diimplementasikan :
    1. Berdasarkan uji coba model pada data diolah sendiri, model yang dibangun menghasilkan akurasi paling baik yaitu sebesar 98% untuk arsitektur VGG16 dengan menggunakan algoritma optimasi Adam dan *learning rate* sebesar 0,0001.
    2. Berdasarkan uji coba model pada data CK+ untuk 7 kelas, model yang dibangun menghasilkan akurasi paling baik yaitu sebesar 97% untuk arsitektur VGG16 dengan menggunakan algoritma optimasi Adam dan *learning rate* sebesar 0,001.

3. Berdasarkan uji coba model pada data CK+ untuk 2 kelas, model yang dibangun menghasilkan akurasi paling baik yaitu sebesar 98% untuk arsitektur VGG16 dengan menggunakan algoritma optimasi Adam dan *learning rate* sebesar 0,0001.
4. Berdasarkan 3 uji coba model, pembangunan model terbaik pada pengenalan wajah adalah arsitektur VGG16 dengan algoritma optimasi Adam dengan *learning rate* yang disesuaikan.
5. Berdasarkan evaluasi terhadap deteksi wajah, model yang dibangun menghasilkan ketepatan paling baik yaitu pada nilai kepercayaan diatas 70%.
6. Berdasarkan evaluasi terhadap deteksi wajah dengan menggunakan *IoU threshold*, model yang dibangun menghasilkan *precision* sebesar 98,83%, *recall* sebesar 99,88%, *F1-score* sebesar 99,35%, dan *Average Precision* sebesar 99,72% yaitu pada nilai *IoU threshold* sebesar 50%.

3. Berdasarkan evaluasi terhadap waktu proses sistem, didapatkan rata-rata waktu proses deteksi wajah selama ± 300 milidetik, rata-rata waktu proses pengenalan ekspresi wajah selama ± 8 milidetik, dan rata-rata waktu proses pelacakan wajah selama ± 10 milidetik. Dengan melakukan penyimpanan hasil pengenalan pada sebuah database MySQL, pengelolaan hasil rekapitulasi dapat diakses terhadap pengguna yang berbeda.

6.2 Saran

Saran yang diberikan untuk pengenalan ekspresi wajah pada pengembangan sistem monitoring perkuliahan, yaitu:

1. Melakukan survei atau korespondensi dalam menentukan kelompok ekspresi terhadap data yang diolah secara mandiri.
2. Membagi target responden menjadi responden dan pakar sebagai salah satu pertimbangan dalam menentukan kelompok ekspresi.
3. Menambahkan variasi kondisi ekspresi wajah pada data latih berdasarkan kondisi sesungguhnya pada proses pembelajaran di kelas, contoh: tertawa, senang, murung, dan lainnya.
4. Melakukan proses augmentasi menggunakan *Generative Adversarial Network* (GAN) untuk menambah jumlah ekspresi wajah data latih sesuai dengan koefisien yang diperlukan.
5. Melakukan eksplorasi terhadap *object tracking* seperti *Deep Sort.*
6. Meningkatkan waktu proses deteksi wajah dengan menggunakan perangkat keras GPU dengan *Compute Capability* diatas 5,3. Dalam tugas akhir ini GPU yang digunakan adalah NVIDIA Tesla K80 *Compute Capability* sebesar 3,7.
7. Menggunakan video dengan resolusi lebih tinggi untuk mendapatkan hasil deteksi yang lebih optimal.

(Halaman ini sengaja dikosongkan)

DAFTAR PUSTAKA

[1] N. K. Benamara, M. Val-Calvo, J. R. Alvarez-Sanchez, A. Diaz-Morcillo, J. M. F. Vicente, E. Fernandez-Jover dan T. B. Stambouli, "Real-Time Emotional Recognition for Sociable Robotics Based on Deep Neural Networks Ensemble," 2019.

[2] A. Karpathy, "Convolutional Neural Networks for Visual Recognition," Stanford University, [Online]. Available: http://cs231n.github.io/. [Diakses 30 March 2020].

[3] S. Sena, "Medium," 13 Novermber 2017. [Online]. Available: https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94. [Diakses 13 March 2020].

[4] S. Sena, "Pengenalan Deep Learning Neural Network," 28 October 207. [Online]. Available: https://medium.com/@samuelsena/pengenalan-deep-learning-8fbb7d8028ac. [Diakses 30 March 2020].

[5] K. H. a. X. Z. a. S. R. a. J. Sun, "Deep Residual Learning for Image Recognition," *he2015deep,* 2015.

[6] K. Simonyan, "Very Deep Convolutional Networks For Large-scale Image Recognition," *Very Deep Convolutional Networks For Large-scale Image Recognition,* vol. I, no. 12, p. 14, 2015.

[7] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar dan I. Matthews, "The Extended Cohn-Kanade Dataset (CK+): A complete facial expression dataset for action unit and emotion-specified expression," *3rd IEEE Workshop on CVPR for Human Communicative Behavior Analysis,* 2010.

[8] P. F. W. V. &. T. S. S. Ekman, "Facial Affect Scoring Technique: A First Validity Study," *Facial Affect Scoring Technique: A First Validity Study,* vol. 3, no. Semiotica, pp. 37-58, 1971.

[9] D. &. O. S. &. a. T. B. &. M. A. &. N. N. &. O. A. T. Beh Mei Yin, "Fusion of face recognition and facial expression detection for authentication: a proposed model," *Fusion of face recognition and facial expression detection for authentication: a proposed model,* pp. 1-8, 2017.

[10 Y. C. S. W. a. L. Z. X. Du, "Overview of deep learning," *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC),* pp. 159-164, 2016.

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

[12 S. Saha. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

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

[14 A. J. A. J. S. M. Jacinto C. Nascimento, "AN ALGORITHM FOR CENTROID-BASED TRACKING OF MOVING OBJECTS," *AN ALGORITHM FOR CENTROID-BASED TRACKING OF MOVING OBJECTS,* 1999.

[15 A. Rosebrock, "pyimagesearch," 23 July 2018. [Online]. Available: https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/. [Diakses Juni 2020].

[16 F. Dirfaux, “Key frame selection to represent a video,” *Proceedings 2000 International Conference on Image Processing (Cat. No.00CH37101),* vol. 2, pp. 275-278, 2002.

[17 rcmalli, “Github Keras VGGFace,” Github, [Online]. Available: https://github.com/rcmalli/keras-vggface. [Diakses 4 February 2020].

[18 L. S. S. A. G. S. E. W. Jie Hu, “Squeeze-and-Excitation Networks,” *Squeeze-and-Excitation Networks,* 2019.

[19 A. Deis, “Data Augmentation for Deep Learning,” [Online]. Available: https://towardsdatascience.com/data-augmentation-for-deep-learning-4fe21d1a4eb9.

[20 R. Irsyad, “Penggunaan Python Web Framework Flask Untuk Pemula,” *Penggunaan Python Web Framework Flask Untuk Pemula.*

[21 tzutalin, “Github LabelImg,” [Online]. Available: https://github.com/tzutalin/labelImg.

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

[23 N. Z. a. S. R. S. Mohseni, “Facial expression recognition using anatomy based facial graph,” *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC),* pp. 3715-3719, 2014.

[24 K. S. B. a. M. T. Eskil, “Anatomy based features for facial expression recognition,” *2014 22nd Signal Processing and Communications Applications Conference (SIU),* pp. 172-175, 2014.

[25 “Sci-Py.org,” Sci-Py.org, [Online]. Available: https://www.scipy.org/about.html. [Diakses 8 March 2020].

[26 G. Nishad, “Medium,” 2 March 2019. [Online]. Available: https://towardsdatascience.com/you-only-look-once-yolo-implementing-yolo-in-less-than-30-lines-of-python-code-97fb9835bfd2. [Diakses 13 March 2020].

[27 A. Rosebrock, “pyimagesearch,” 7 November 2016. [Online]. Available: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/. [Diakses 12 March 2020].

[28 Y. Yun, “Github,” 21 February 2019. [Online]. Available: https://github.com/YunYang1994/tensorflow-yolov3. [Diakses 13 March 2020].

[29 K. a. Z. Z. a. L. Z. a. Q. Y. Zhang, “Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks,” *IEEE Signal Processing Letters,* vol. 23, p. 10, 2016.

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

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

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

[33 “Deep learning for complete beginners: convolutional neural networks with keras,” Cambridgespark, 20 March 2017. [Online]. Available: https://cambridgespark.com/content/tutorials/convolu

tional-neural-networks-with-keras/index.html. [Diakses 29 November 2018].

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

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

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

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

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

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

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

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

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

[43 “Binary Image Analysis,” dalam *Computer Vision*, 2000, pp. 63-75.

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

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

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

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

[48 J. B. ANDREAS GIRGENSOHN, “Time-Constrained Keyframe Selection Technique,” *Time-Constrained Keyframe Selection Technique,* vol. 11, pp. 347-358, 2000.

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

BIODATA PENULIS

Marde Fasma, lahir di Surabayapada tanggal 6 Agustus 1998. Penulis menempuh pendidikan mulai dari TK Insan Mulya (2004 – 2006), SD Insan Mulya (2006 – 2012), SD Negeri 01 Winongo (2006 – 2012), SMP Negeri 1 Madiun (2012 – 2014), SMA Negeri 3 Madiun (2014 – 2016), dan sekarang sedang menjalani pendidikan S1 Teknik Informatika di ITS. Penulis aktif dalam organisasi dan kepanitiaan menjadi Sekretaris Umum Lembaga Dakwah Jurusan Keluarga Muslim Informatika, staff ahli Himpunan Mahasiswa Teknik Informatika, administrator laboratorium *Mobile Innovation* Studio, dan Schematics. Komunikasi dengan penulis dapat melalui telepon: +6289615125347 dan *email*: **mardefasma123up@gmail.com**.