



TUGAS AKHIR - IF184802

SINTESA CITRA WAJAH MENGGUNAKAN GENERATIVE ADVERSARIAL NETWORK UNTUK AUGMENTASI DATA PADA APLIKASI PENGENALAN WAJAH

**ALDINATA RIZKY REVANDA
NRP 05111640000023**

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

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

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



TUGAS AKHIR - IF184802

**SINTESA CITRA WAJAH MENGGUNAKAN
GENERATIVE ADVERSARIAL NETWORK
UNTUK AUGMENTASI DATA PADA APLIKASI
PENGENALAN WAJAH**

**ALDINATA RIZKY REVANDA
NRP 05111640000023**

**Dosen Pembimbing I
Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.**

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

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

(Halaman ini sengaja dikosongkan)



UNDERGRADUATE THESIS - IF184802

FACE IMAGE SYNTHESIS USING GENERATIVE ADVERSARIAL NETWORKS FOR DATA AUGMENTATION IN FACE RECOGNITION APPLICATIONS

**ALDINATA RIZKY REVANDA
NRP 05111640000023**

First Advisor

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

Second Advisor

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

Department of Informatics

Faculty of Intelligent Electrical and Informatics Technology

Institut Teknologi Sepuluh Nopember

Surabaya 2020

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

SINTESA CITRA WAJAH MENGGUNAKAN GENERATIVE ADVERSARIAL NETWORK UNTUK AUGMENTASI DATA PADA APLIKASI PENGENALAN WAJAH

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:

ALDINATA RIZKY REVANDA
NRP: 05111640000023

Disetujui oleh Pembimbing Tugas Akhir:

1. Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.
(NIP. 19751220 200112 2 002) (Pembimbing 1)
2. Dr. Eng. Nanik Suciati, S.Kom., M.Kom.
(NIP. 19710428 199412 2 001) (Pembimbing 2)

SURABAYA
JUNI, 2020

(Halaman ini sengaja dikosongkan)

SINTESA CITRA WAJAH MENGGUNAKAN GENERATIVE ADVERSARIAL NETWORK UNTUK AUGMENTASI DATA PADA APLIKASI PENGENALAN WAJAH

Nama : Aldinata Rizky Revanda
NRP : 05111640000023
Departemen : Teknik Informatika, FTEIC-ITS
Pembimbing I : Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.
Pembimbing II : Dr. Eng. Nanik Suciati, S.Kom., M.Kom.

ABSTRAK

Pengenalan wajah telah menjadi bidang penelitian yang populer dalam visi komputer dan banyak diaplikasikan di berbagai sektor. Tantangan dari aplikasi pengenalan wajah adalah jika dataset untuk pelatihan yang dimiliki terbatas maka tingkat pengenalan wajah akan menjadi kurang efektif. Generative Adversarial Network (GAN) merupakan salah satu metode pada Deep Learning yang mampu memodifikasi gambar (sintesa citra) dengan kualitas tinggi.

Tugas Akhir ini bertujuan untuk melakukan sintesa citra wajah menggunakan Generative Adversarial Network dan akan ditambahkan pada data latih pengenalan wajah sebagai bentuk augmentasi data. Metode GAN bertujuan untuk membuat representasi latent space dari citra wajah kemudian melakukan penyesuaian terhadap latent space untuk menambah gaya atau kondisi dari citra wajah awal. Kemudian variasi dari sintesa citra wajah tersebut ditambahkan pada dataset aplikasi pengenalan wajah untuk membantu pengumpulan dataset serta menambah efektivitas model classifier. Sistem yang dibangun memiliki dua proses utama yaitu Augmentasi Data dan Pengenalan Wajah. Pada proses Augmentasi Data memiliki 4 tahap yaitu praproses citra wajah, transfer learning dan implementasi GAN, penyesuaian latent space, serta pembentukan ulang citra wajah. Pada proses Pengenalan Wajah memiliki 4 tahap yaitu face detection and

extraction, transfer learning VGG Face Net dan face embedding, pembuatan model pengenalan wajah dan fitting model, serta face classification.

Berdasarkan hasil uji coba yang telah dilakukan, model GAN yang baik untuk membuat representasi latent space adalah jenis pre-trained dari StyleGAN dengan menggunakan data latih dari dataset FFHQ yang mampu menghasilkan nilai loss mencapai 0,15. Metode penyesuaian latent space yang baik untuk menambah gaya atau kondisi adalah metode latent direction. Kemudian hasil sintesa citra wajah yang ditambahkan pada data latih pengenalan wajah sebagai bentuk augmentasi data mampu menaikkan akurasi model classifier untuk pengenalan wajah dari 0,74 menjadi 0,89.

Kata kunci: *Sintesa Citra Wajah, Generative Adversarial Network, Augmentasi Data, Pengenalan Wajah.*

FACE IMAGE SYNTHESIS USING GENERATIVE ADVERSARIAL NETWORKS FOR DATA AUGMENTATION IN FACE RECOGNITION APPLICATIONS

Student's Name : Aldinata Rizky Revanda

Student's ID : 05111640000023

Department : Informatics, Faculty of ELECTICS-ITS

First Advisor : Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.

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

ABSTRACT

Face recognition has become a popular research field in computer vision and is widely applied in various sectors. The challenge with face recognition applications is that if the dataset for training is limited, the face recognition rate will be less effective. Generative Adversarial Network (GAN) is a method in deep learning that can modify images (image synthesis) with high quality.

This final project aims to synthesize face images using the Generative Adversarial Network and will be added to the face recognition training data as a form of data augmentation. The GAN method aims to create a representation of the latent space of the face image then make adjustments to the latent space to add the style or condition of the initial face image. Then the variations of the face image synthesis are added to the face recognition application dataset to help collect the dataset and increase the effectiveness of the classifier model. The system built has two main processes, namely Data Augmentation and Face Recognition. The Data Augmentation process has 4 stages: face image preprocessing, transfer learning and GAN implementation, latent space adjustment, and face image reconstruction. The Face Recognition process has 4 stages: face detection and extraction, transfer learning VGG Face Net and face embedding, making face recognition models and fitting models, and face classification.

Based on the results of the trials that have been carried out, a good GAN model for making latent space representations is the pre-trained type of StyleGAN using training data from the FFHQ dataset which can produce a loss value of 0.15. A good method of adjusting latent space to add a style or condition is the latent direction method. Then the results of the face image synthesis added to the face recognition training data as a form of data augmentation were able to increase the accuracy of the classifier model for face recognition from 0.74 to 0.89.

Keywords: *Face Image Synthesis, Generative Adversarial Networks, Data Augmentation, Face Recognition.*

KATA PENGANTAR

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

“SINTESA CITRA WAJAH MENGGUNAKAN GENERATIVE ADVERSARIAL NETWORK UNTUK AUGMENTASI DATA PADA APLIKASI PENGENALAN WAJAH”

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. Chastine Fatichah, S.Kom., M.Kom. dan Dr. Eng. Nanik Suciati, S.Kom., M.Kom. selaku pembimbing I dan II yang telah membimbing, memberikan motivasi, dan masukan dalam menyelesaikan Tugas Akhir ini.
3. Dr. Eng. Chastine Fatichah, S.Kom., M.Kom. selaku Ketua Departemen Teknik Informatika ITS, seluruh dosen dan karyawan Departemen Teknik Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa kuliah di Teknik Informatika ITS.
4. Marde Fasma, Steve Daniels, dan Rasyid Fajar yang membantu dan menemani penulis selama perkuliahan semester akhir dan pengerjaan Tugas Akhir ini.
5. Admin-admin Laboratorium Komputasi Cerdas & Visi (KCV) yang memberikan kesempatan penulis untuk fokus mengerjakan Tugas Akhir ini dan menyediakan tempat di laboratorium tersebut.

6. Seluruh mahasiswa *user* TA Teknik Informatika ITS angkatan 2016 yang telah menjadi teman penulis selama pengerjaan Tugas Akhir ini.
7. Seluruh mahasiswa Teknik Informatika ITS angkatan 2016 yang telah menjadi teman penulis selama menjalani masa kuliah di Teknik Informatika ITS.
8. Serta semua pihak yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini, termasuk semua teman yang telah bersedia menjadi sampel untuk mengumpulkan dataset citra wajah (Ardo, Fahmi, Fasma, Handika, Rasyid, Steve, Sugiarto, dan Yuda).

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

Surabaya, Juni 2020

DAFTAR ISI

LEMBAR PENGESAHAN.....	vii
ABSTRAK.....	iii
ABSTRACT	v
KATA PENGANTAR	vii
DAFTAR ISI.....	ix
DAFTAR TABEL.....	xiii
DAFTAR KODE SUMBER	xv
DAFTAR GAMBAR.....	xvii
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Batasan Permasalahan	3
1.4 Tujuan	3
1.5 Manfaat.....	3
1.6 Metodologi	4
1.6.1 Penyusunan Proposal Tugas Akhir	4
1.6.2 Studi Literatur	4
1.6.3 Implementasi Perangkat Lunak.....	4
1.6.4 Pengujian dan Evaluasi.....	5
1.6.5 Penyusunan Buku	5
1.7 Sistematika Penulisan Laporan	5
BAB II TINJAUAN PUSTAKA.....	7
2.1 Generative Adversarial Network (GAN)	7
2.1.1 Prinsip dari GAN	7
2.1.2 Model Pembelajaran dari GAN (Learning Model)...	8
2.1.3 Model-model yang Diturunkan dari GAN.....	9
2.1.4 Aplikasi dari GAN	10
2.2 Style-GAN Model	10
2.3 Representasi Latent Space.....	12
2.4 Mapping Network	14
2.5 Adaptive Instance Normalization (AdaIN)	15
2.6 Face Recognition atau Pengenalan Wajah	16
2.7 Convolutional Neural Network	17

2.7.1	Convolutional Layer	18
2.7.2	Pooling Layer.....	18
2.7.3	Activation Function	19
2.7.4	Loss Function.....	20
2.7.5	Dropout	20
2.8	Confusion Matrix.....	20
2.9	Dataset	22
2.10	Transfer Learning	22
2.11	Python.....	23
2.12	Library	23
2.12.1	Keras	23
2.12.2	TensorFlow	24
2.12.3	OpenCV	24
2.12.4	Numpy.....	24
2.12.5	Scikit-learn.....	24
2.12.6	Matplotlib.....	25
2.12.7	Scikit-image	25
2.12.8	SciPy	25
BAB III	PERANCANGAN SISTEM.....	27
3.1	Perancangan Data	27
3.2	Desain Umum Sistem	31
3.2.1	Proses Augmentasi Data	32
3.2.2	Proses Pengenalan Wajah	36
BAB IV	IMPLEMENTASI.....	39
4.1	Lingkungan Implementasi	39
4.1.1	Perangkat Keras	39
4.1.2	Perangkat Lunak	39
4.2	Implementasi Proses Augmentasi Data	40
4.2.1	Praproses Citra Wajah.....	40
4.2.2	Transfer Learning dan Implementasi GAN.....	45
4.2.3	Penyesuaian Latent Space.....	54
4.2.4	Pembentukan Ulang Citra Wajah.....	56
4.3	Implementasi Proses Pengenalan Wajah	57
4.3.1	<i>Face Detection and Extraction</i>	57

4.3.2	Transfer Learning VGG Face Net dan Face Embedding.....	59
4.3.3	Pembuatan Model Pengenalan Wajah dan Fitting Model.....	62
4.3.4	Face Classification.....	63
BAB V UJI COBA DAN EVALUASI.....		67
5.1	Lingkungan Uji Coba.....	67
5.2	Deskripsi Dataset.....	67
5.3	Hasil Praproses.....	68
5.3.1	Praproses Citra Pada Augmentasi Data	69
5.3.2	Praproses Citra Pada Pengenalan Wajah	70
5.4	Skenario Uji Coba	70
5.4.1	Skenario Uji Coba pada tahap implementasi Generative Adversarial Network (GAN)	71
5.4.2	Skenario Uji Coba Pada Tahap Penyesuaian Representasi Latent Space	75
5.4.3	Hasil Uji Coba Pada Proses Augmentasi Data	79
5.4.4	Hasil Uji Coba Pada Proses Pengenalan Wajah	82
5.5	Hasil dan Evaluasi.....	98
BAB VI KESIMPULAN DAN SARAN.....		105
6.1	Kesimpulan.....	105
6.2	Saran.....	106
DAFTAR PUSTAKA		109
LAMPIRAN.....		115
L.1	Dataset Sumber Citra Wajah Asli Sebagai Input Untuk Dilakukan Sintesa Citra Wajah Dengan GAN	115
L.2	Hasil Praproses Citra Wajah Untuk Siap Digunakan Sebagai Input Pada GAN	116
L.3	Hasil Semua Uji Coba pada tahap implementasi Generative Adversarial Network (GAN)	117
L.4	Hasil Sintesa Citra Wajah Setelah Diproses Melalui GAN, Penyesuaian Latent Space, dan Pembentukan Ulang Citra Wajah	119
L.5	Dataset Sumber Citra Wajah Asli Sebagai Data Uji Pada Uji Coba Pengenalan Wajah.....	128

L.6	Contoh Hasil Face Detection and Extraction Pada Proses Pengenalan Wajah	133
L.7	Contoh Hasil Face Classification Pada Proses Pengenalan Wajah.....	134
L.8	Hasil Lengkap Face Classification Pada Semua Skenario Uji Coba Pengenalan Wajah.....	135
BIODATA PENULIS		149

DAFTAR TABEL

Tabel 2.1 Klasifikasi dari model GAN [4]	9
Tabel 3.1 Spesifikasi dari dataset referensi CelebA-HQ	28
Tabel 3.2 Spesifikasi dari dataset referensi FFHQ	29
Tabel 3.3 Spesifikasi dari dataset sumber	30
Tabel 3.4 Waktu pelatihan dari StyleGAN pada FFHQ dataset yang telah dicoba oleh NVIDIA [29]	34
Tabel 5.1 Spesifikasi parameter awal arsitektur GAN	71
Tabel 5.2 Hasil uji coba penggantian jenis <i>pretrained</i> StyleGAN	72
Tabel 5.3 Hasil uji coba penggantian ukuran <i>batch</i>	73
Tabel 5.4 Hasil uji coba penggantian <i>learning rate</i>	73
Tabel 5.5 Hasil uji coba penggantian <i>random noise</i>	74
Tabel 5.6 Hasil uji coba penggantian jumlah iterasi	74
Tabel 5.7 Spesifikasi data uji untuk uji coba pengenalan wajah	84
Tabel 5.8 Spesifikasi data latih skenario A	84
Tabel 5.9 Hasil evaluasi pada skenario A	85
Tabel 5.10 Hasil prediksi pada salah satu identitas wajah Skenario A	86
Tabel 5.11 Spesifikasi data latih skenario B1	86
Tabel 5.12 Hasil evaluasi pada skenario B1	87
Tabel 5.13 Hasil prediksi pada salah satu identitas wajah Skenario B1	88
Tabel 5.14 Spesifikasi data latih skenario B2	88
Tabel 5.15 Hasil evaluasi pada skenario B2	89
Tabel 5.16 Hasil prediksi pada salah satu identitas wajah Skenario B2	90
Tabel 5.17 Spesifikasi data latih skenario B3	90
Tabel 5.18 Hasil evaluasi pada skenario B3	91
Tabel 5.19 Hasil prediksi pada salah satu identitas wajah Skenario B3	92
Tabel 5.20 Spesifikasi data latih skenario B4	92
Tabel 5.21 Hasil evaluasi pada skenario B4	93

Tabel 5.22 Hasil prediksi pada salah satu identitas wajah Skenario B4	94
Tabel 5.23 Spesifikasi data latih skenario B5.....	94
Tabel 5.24 Hasil evaluasi pada skenario B5.....	95
Tabel 5.25 Hasil prediksi pada salah satu identitas wajah Skenario B5	96
Tabel 5.26 Spesifikasi data latih skenario B6.....	96
Tabel 5.27 Hasil evaluasi pada skenario B6.....	97
Tabel 5.28 Hasil prediksi pada salah satu identitas wajah Skenario B6	98
Tabel 5.29 Hasil evaluasi pada semua skenario proses pengenalan wajah	101
Tabel 5.30 Parameter GAN optimal yang ditetapkan.....	104

DAFTAR KODE SUMBER

Kode Sumber 4.2.1 Implementasi Praproses Citra Wajah	41
Kode Sumber 4.2.2 Implementasi <i>class LandmarksDetector</i>	42
Kode Sumber 4.2.3 Implementasi <i>function image_align</i>	45
Kode Sumber 4.2.4 Implementasi <i>transfer learning pre-trained StyleGAN</i>	49
Kode Sumber 4.2.5 Implementasi model <i>Generator</i>	51
Kode Sumber 4.2.6 Implementasi model <i>Perceptual</i>	54
Kode Sumber 4.2.7 Implementasi <i>common average style mixing</i>	54
Kode Sumber 4.2.8 Implementasi <i>vector operation in latent space</i>	54
Kode Sumber 4.2.9 Implementasi <i>NVIDIA's style mixing technique</i>	56
Kode Sumber 4.2.10 Implementasi <i>latent direction</i>	56
Kode Sumber 4.2.11 Implementasi pembentukan ulang citra wajah	57
Kode Sumber 4.3.1 Implementasi <i>face detection and extraction</i>	58
Kode Sumber 4.3.2 Implementasi <i>transfer learning VGG Face Net</i>	60
Kode Sumber 4.3.3 Implementasi <i>face embedding</i>	62
Kode Sumber 4.3.4 Implementasi pembuatan model pengenalan wajah	63
Kode Sumber 4.3.5 Implementasi <i>fitting model</i>	63
Kode Sumber 4.3.6 Implementasi <i>face classification</i>	65

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

Gambar 2.1 Arsitektur dari <i>generative adversarial network</i> [4] ...	8
Gambar 2.2 Perbedaan arsitektur tradisional GAN dan StyleGAN [7]	12
Gambar 2.3 Contoh penggambaran pembuatan <i>latent space</i> [8].	13
Gambar 2.4 Generator dengan <i>Mapping Network</i> [10]	14
Gambar 2.5 Generator dengan <i>Adaptive Instance Normalization</i> (AdaIN) [10].....	15
Gambar 2.6 Contoh Arsitektur Sistem Pengenalan Wajah [2]....	16
Gambar 2.7 Contoh Arsitektur <i>Convolutional Neural Network</i> [12]	17
Gambar 2.8 Contoh proses <i>Convolution Layer</i> [11]	18
Gambar 2.9 Contoh proses <i>Pooling Layer (Max Pooling)</i> [11] ..	19
Gambar 2.10. Contoh <i>Non-Linear Activation Function, Sigmoid</i> dan <i>Tanh function</i> [13]	19
Gambar 2.11 Contoh <i>confusion matrix</i> dengan empat kombinasi nilai prediksi dan aktual yang berbeda [14]	21
Gambar 3.1 Contoh citra dari dataset referensi CelebA-HQ [26]	28
Gambar 3.2 Contoh citra dari dataset referensi FFHQ [27]	29
Gambar 3.3 Contoh citra dari dataset sumber untuk data uji GAN	30
Gambar 3.4 Diagram alir sistem yang dibangun	31
Gambar 3.5 Diagram alir proses augmentasi data	32
Gambar 3.6 Visualisasi <i>68-points facial landmark</i> [28]	33
Gambar 3.7 Contoh operasi aritmatika pada vektor <i>latent space</i> [9]	35
Gambar 3.8 Diagram alir proses pengenalan wajah	36
Gambar 3.9 Visualisasi arsitektur <i>VGG Face Net</i> [30]	37
Gambar 5.1 (a) Citra input; (b) Hasil praproses pada augmentasi data	69
Gambar 5.2 (a) Citra input; (b) Hasil praproses pada pengenalan wajah	70
Gambar 5.3 Contoh pembentukan ulang citra dengan parameter awal	75

Gambar 5.4 Contoh pembentukan ulang citra dengan parameter optimal.....	75
Gambar 5.5 Hasil uji coba metode <i>common average style mixing</i>	76
Gambar 5.6 Hasil uji coba metode <i>vector operation in latent space</i>	77
Gambar 5.7 Hasil uji coba metode <i>NVIDIA's style mixing technique</i>	78
Gambar 5.8 Hasil uji coba metode <i>latent direction</i>	79
Gambar 5.9 Contoh hasil proses augmentasi data untuk ekspresi senyum.....	80
Gambar 5.10 Contoh hasil proses augmentasi data untuk ekspresi sedih.....	80
Gambar 5.11 Contoh hasil proses augmentasi data untuk pose kanan	81
Gambar 5.12 Contoh hasil proses augmentasi data untuk pose kiri	81
Gambar 5.13 Contoh hasil proses augmentasi data untuk atribut kacamata.....	82
Gambar 5.14 Contoh data uji pada model <i>classifier</i> pengenalan wajah	83
Gambar 5.15 Plot hasil fitting model pengenalan wajah skenario A	85
Gambar 5.16 Plot hasil fitting model pengenalan wajah skenario B1	87
Gambar 5.17 Plot hasil fitting model pengenalan wajah skenario B2	89
Gambar 5.18 Plot hasil fitting model pengenalan wajah skenario B3	91
Gambar 5.19 Plot hasil fitting model pengenalan wajah skenario B4	93
Gambar 5.20 Plot hasil fitting model pengenalan wajah skenario B5	95
Gambar 5.21 Plot hasil fitting model pengenalan wajah skenario B6	97

Gambar 5.22 Contoh gambar hasil citra wajah yang tidak sesuai seperti yang diharapkan pada metode <i>NVIDIA's style mixing technique</i>	99
Gambar 5.23 Contoh hasil citra wajah pada metode <i>latent direction</i> dengan koefisien perkalian yang melebihi batas	100
Gambar 5.24 Contoh hasil citra wajah dengan metode <i>latent direction</i> untuk gaya atau kondisi yang kompleks	101

(Halaman ini sengaja dikosongkan)

BAB I

PENDAHULUAN

1.1 Latar Belakang

Selama sepuluh tahun terakhir, *face recognition* atau pengenalan wajah telah menjadi bidang penelitian yang populer dalam visi komputer serta merupakan salah satu aplikasi analisis dan pemahaman gambar yang paling sukses [1]. Hal tersebut dikarenakan pengaplikasian dari teknologi pengenalan wajah yang sangat luas di berbagai sektor. Misalnya dalam sektor keamanan, pengenalan wajah digunakan sebagai fitur untuk masuk ke dalam sebuah sistem atau perangkat lunak. Kemudian dalam sektor komersial, pengenalan wajah digunakan sebagai fitur pembayaran yang terintegrasi dengan akun keuangan seseorang. Bahkan untuk yang lebih maju, pengenalan wajah dapat diimplementasikan pada CCTV untuk mendeteksi secara otomatis siapa saja yang melalui area di sekitar CCTV tersebut, serta masih banyak lagi.

Pernyataan umum tentang pengenalan wajah (dalam visi komputer) dapat dirumuskan sebagai berikut: diberikan gambar video atau diam dari sebuah adegan, mengidentifikasi atau memverifikasi satu atau lebih orang dalam adegan tersebut menggunakan database yang tersimpan. Mayoritas penelitian sejauh ini berfokus pada wajah frontal dan pengenalan wajah ekspresi netral. Mengenali wajah dengan handal di seluruh perubahan pose dan ekspresi telah terbukti menjadi masalah yang jauh lebih sulit, misalnya tingkat pengenalan wajah akan berkurang secara drastis ketika sudut pose wajah lebih besar dari 30 derajat [2]. Hal tersebut erat juga kaitannya dengan ketersediaan dataset wajah yang digunakan ketika proses *learning* atau pembelajaran pada metode pengenalan wajah. Kurang lengkapnya citra wajah yang tersimpan pada dataset akan mengurangi efektivitas model yang digunakan untuk mengenali wajah. Selain itu, pengumpulan dataset untuk pengenalan wajah kebanyakan masih menggunakan cara manual dan kurang efisien, yaitu dengan mengambil lima

hingga sepuluh citra wajah dari setiap orang lalu akan dilakukan pelabelan pada dataset tersebut sesuai dengan identitas wajahnya.

Di sisi lain, saat ini sedang berkembang sebuah metode yang disebut *Generative Adversarial Network* (GAN) yang mampu memberikan revolusi besar terhadap *deep learning*. Beberapa potensi penggunaan GAN adalah dapat menghasilkan citra berkualitas tinggi, memperbaiki kualitas citra, menghasilkan citra dari teks, mengubah citra dari satu domain ke domain lain, mengubah tampilan citra wajah seiring bertambahnya usia dan banyak lagi [3]. Dari potensi tersebut, kita dapat melihat adanya peluang pemanfaatan metode GAN khususnya dalam membuat atau memodifikasi citra wajah. Contoh kasusnya adalah dari satu citra wajah kita bisa membuat citra wajah dengan identitas yang sama serta dengan penambahan beberapa gaya atau kondisi seperti pose wajah (menoleh ke kanan atau ke kiri, ke atas atau kebawah), ekspresi wajah (netral menjadi tersenyum), hingga penambahan atribut pada wajah (seperti kacamata).

Pada Tugas Akhir ini, penulis mengusulkan sintesa citra wajah menggunakan *Generative Adversarial Network* untuk augmentasi data pada aplikasi pengenalan wajah. Hasil sintesa citra wajah dari GAN dapat ditambahkan pada data latih pengenalan wajah sebagai bentuk augmentasi data sehingga mampu menambah efisiensi pengumpulan dataset serta mampu menambah efektivitas model pada aplikasi pengenalan wajah.

1.2 Rumusan Masalah

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

1. Bagaimana implementasi metode GAN dalam membuat representasi *latent space* dari citra wajah yang diinginkan serta membetuknya kembali?
2. Bagaimana metode penyesuaian representasi *latent space* yang digunakan dalam menambahkan gaya atau kondisi pada citra wajah awal?

3. Bagaimana dampak penambahan hasil sintesa citra wajah dari GAN terhadap akurasi model *classifier* pada aplikasi pengenalan wajah?

1.3 Batasan Permasalahan

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

1. Dataset referensi adalah dataset dari “*CelebA-HQ Dataset*” dan “*Flickr-Face-HQ Dataset*” yang telah digunakan untuk proses *learning* sebelumnya pada *pre-trained* model GAN. Dataset sumber adalah dataset yang dikumpulkan sendiri oleh pengusul.
2. Gaya atau kondisi sintesa citra wajah yang diharapkan hanya untuk perubahan ekspresi (senyum dan sedih), perubahan pose (pose kanan dan kiri), dan penambahan atribut (kacamata).
3. Implementasi dari semua program menggunakan bahasa pemrograman *Python*.
4. *Running* program menggunakan *Google Colaboratory*.
5. Model GAN dan model untuk pengenalan wajah menggunakan metode *Transfer Learning*.

1.4 Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah melakukan sintesa citra wajah menggunakan *Generative Adversial Network* dan akan ditambahkan pada data latih pengenalan wajah sebagai bentuk augmentasi data.

1.5 Manfaat

Manfaat yang diharapkan dari pengerjaan Tugas Akhir ini adalah:

- dapat mengimplementasikan metode GAN yang baik untuk melakukan sintesa citra wajah,
- dapat menambah efisiensi pengumpulan dataset pada aplikasi pengenalan wajah, serta

- dapat menambah efektivitas model *classifier* pada aplikasi pengenalan wajah.

1.6 Metodologi

Pembuatan Tugas Akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Proposal Tugas Akhir

Tahap awal yang dilakukan dalam proses pengerjaan Tugas Akhir ini adalah penyusunan proposal Tugas Akhir. Proposal Tugas Akhir ini berisi pendahuluan, tinjauan pustaka, dan metodologi dari Tugas Akhir yang akan dibuat. Pendahuluan terdiri atas latar belakang usulan Tugas Akhir, rumusan masalah yang diangkat dan batasan masalah yang ditentukan. Tinjauan pustaka menjadi referensi pendukung pembuatan Tugas Akhir. Metodologi berisi penjelasan tahapan penyusunan Tugas Akhir. Selain itu, terdapat jadwal kegiatan pengerjaan Tugas Akhir.

1.6.2 Studi Literatur

Tahap yang kedua adalah studi literatur dimana penulis mencari referensi yang terkait untuk menyelesaikan studi kasus, diantara lain adalah buku, *scientific paper*, artikel di internet, dan materi kuliah yang terkait dengan *Digital Image Processing*, *Image Classification*, *Deep Learning*, *Convolutional Neural Network*, *Generative Adversarial Network*, *Tensorflow*, dan *Keras*.

1.6.3 Implementasi Perangkat Lunak

Tahap implementasi meliputi implementasi algoritma pada perangkat lunak yang telah didukung oleh hasil analisis dan desain pada tahap sebelumnya. Implementasi ini dilakukan dengan menggunakan bahasa pemrograman *Python* serta *library TensorFlow* dan *Keras* yang mendukung *Deep Learning*, khususnya GAN dan CNN, serta pemrosesan *Graphics Processing Units* (GPU). Semua program dijalankan melalui laptop pribadi pengusul dan *Google Colaboratory*.

1.6.4 Pengujian dan Evaluasi

Tahap pengujian yang dilakukan untuk Tugas Akhir ini adalah untuk mengetahui seberapa baik kemampuan model GAN dalam melakukan sintesa citra wajah. Kemudian juga akan dilakukan uji coba pemanfaatan hasil sintesa citra wajah pada aplikasi pengenalan wajah dengan harapan dapat meningkatkan performa pengenalan wajah. Sehingga pada tahap evaluasi akan berfokus kepada seberapa besar peningkatan performa pengenalan wajah ketika tanpa ada penambahan data latih dan ketika ada penambahan data latih dengan hasil sintesa citra wajah dari GAN.

1.6.5 Penyusunan Buku

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam Tugas Akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat sebagai dokumentasi dari 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 laporan 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 *Generative Adversarial Network*, *Convolutional Neural Network*, *Tensorflow*, *Keras*, dan *library* lain yang digunakan.

Bab III Perancangan Sistem

Bab ini berisi pembahasan mengenai perancangan dari metode *Generative Adversarial Network* yang

digunakan untuk sintesa citra wajah serta metode *Convolutional Neural Network* yang digunakan untuk pengenalan wajah.

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 berisi kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada saat proses pengerjaan Tugas Akhir, serta 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 *Generative Adversarial Network*, *Convolutional Neural Network* dan beberapa teori lain yang mendukung pembuatan Tugas Akhir. Penjelasan ini bertujuan untuk memberikan gambaran umum dan diharapkan dapat mendukung sistem yang dibangun.

2.1 Generative Adversarial Network (GAN)

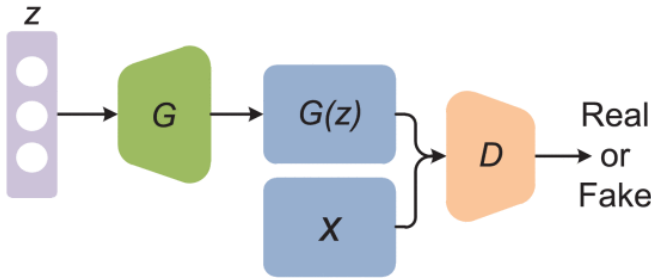
Pembuatan citra dari sebuah dataset menjadi bidang penelitian yang menarik di *Artificial Intelligence*. Sebuah model yang mampu membuat data baru dari distribusi data awal disebut dengan *generative model*. Alasan kenapa sekarang manusia mulai mengadopsi teknologi pembuatan gambar tersebut adalah karena dengan adanya teknologi ini mampu mengurangi kebutuhan tenaga kerja manusia dalam pembuatan konten industri seperti *game*. Karena pembuatan gambar secara manual akan menimbulkan biaya yang tinggi dan kelambatan dalam pekerjaan [3].

Pada tahun 2014, kerangka kerja *Generative Adversarial Network* (GAN) telah ditemukan oleh Goodfellow et al., kerangka kerja tersebut memperkirakan *generative model* melalui proses *adversarial* atau permusuhan antara dua *neural network* yang bersaing satu sama lain. Jaringan-jaringan itu disebut Generator dan Diskriminator, yang dapat dianalogikan dengan Generator adalah pembuat uang palsu sementara Diskriminator adalah polisi yang mencoba mendeteksi apakah uang itu palsu (dibuat oleh Generator) atau nyata (diproduksi oleh data latihan) [3].

2.1.1 Prinsip dari GAN

GAN terinspirasi oleh teori permainan yaitu generator G dan diskriminator D akan saling berkompetisi satu sama lain untuk mencapai kesetimbangan *Nash* pada proses pelatihan. Input dari Generetor adalah *random noise vector* z (biasanya sebuah

distribusi seragam atau normal). *Noise* dipetakan ke ruang data baru melalui generator G untuk mendapatkan sampel palsu, $G(z)$, yang merupakan vektor multi-dimensional. Dan, diskriminator D adalah sebuah *binary classifier*, dibutuhkan sampel nyata dari dataset dan sampel palsu yang dihasilkan oleh generator G sebagai input, dan output dari diskriminator D mewakili probabilitas bahwa sampel tersebut adalah nyata daripada sampel palsu. Ketika diskriminator D tidak dapat menentukan apakah data berasal dari dataset nyata atau generator, keadaan optimal tercapai. Pada titik ini, kita memperoleh model generator G , yang telah mempelajari distribusi data nyata [4]. Arsitektur GAN diilustrasikan pada Gambar 2.1.



Gambar 2.1 Arsitektur dari *generative adversarial network* [4]

2.1.2 Model Pembelajaran dari GAN (Learning Model)

Sebagai dua pemain dalam teori permainan, generator dan diskriminator memiliki fungsi kehilangan (*loss functions*) mereka sendiri. Dalam hal ini, kami menyebutnya $J(G)$ dan $J(D)$. Diskriminator D didefinisikan sebagai pengklasifikasi biner, dan fungsi kehilangannya diwakili oleh *cross entropy* [4],

$$J^{(D)} = -\frac{1}{2} \mathbb{E}_{x \sim p_{data}} \log D(x) - \frac{1}{2} \mathbb{E}_z \log (1 - D(G(z))) \quad (2.1)$$

di mana x mewakili sampel nyata, z mewakili *random noise vector*, $G(z)$ adalah data yang dihasilkan oleh generator, dan E mewakili ekspektasi. $D(x)$ menunjukkan probabilitas bahwa D membedakan

x sebagai data nyata, dan $D(G(z))$ menunjukkan probabilitas bahwa D menentukan data yang dihasilkan oleh G . Tujuan D adalah untuk menentukan sumber data dengan benar, sehingga ia ingin $D(G(z))$ mendekati 0, sedangkan tujuan G adalah untuk mendekatkannya ke 1. Berdasarkan ide ini, terdapat konflik antara dua model ini (yaitu, *zero-sum game*). Oleh karena itu, fungsi kehilangan dari generator dapat diturunkan oleh diskriminator [4]:

$$J^{(G)} = -J^{(D)} \quad (2.2)$$

Akibatnya, masalah optimasi GAN ditransformasikan ke dalam *game minimax* seperti yang ditunjukkan di bawah ini,

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))] \quad (2.3)$$

Dalam proses pelatihan, parameter dalam G diperbarui bersama dengan proses pembaruan parameter dalam D . Ketika $D(G(z)) = 0,5$, diskriminator tidak dapat menentukan perbedaan antara dua distribusi ini, dan dalam status ini, model akan mencapai solusi optimal global [4].

2.1.3 Model-model yang Diturunkan dari GAN

Karena kekurangan dari model GAN yang asli, berbagai model GAN turunan diusulkan oleh beberapa peneliti, dan model GAN turunan ini dapat diklasifikasikan ke dalam dua kelompok, GAN berbasis optimasi arsitektur, dan GAN berbasis optimasi fungsi objektif, seperti yang ditunjukkan pada Tabel 2.1.

Tabel 2.1 Klasifikasi dari model GAN [4]

Architecture Optimization Based GANs	Convolution based GANs	DCGAN [12]
	Condition based GANs	CGANs [13]; InfoGAN [14]; ACGAN [15]
	Autoencoder based GANs	AAE [16]; BiGAN [17]; ALI [18]; AGE [19]; VAE-GAN [20]
Objective Function Optimization Based GANs	unrolled GAN [21]; f-GAN [22]; Mode-Regularized GAN [23]; Least-Square GAN [24]; Loss-Sensitive GAN [25]; EBGAN [26]; WGAN [27]; WGAN-GP [28]; WGAN-LP [29]	

2.1.4 Aplikasi dari GAN

Sebagai salah satu model generatif, aplikasi GAN yang paling utama adalah untuk pembuatan data. Yaitu belajar dari distribusi sampel nyata, dan menghasilkan sampel yang konsisten dengan distribusi tersebut. GAN dapat diaplikasikan pada bidang visi komputer, *natural language processing* (NLP), dan bidang lainnya. Namun hingga saat ini, aplikasi GAN yang paling sukses adalah di bidang visi komputer, seperti contohnya *image translation*, *image super resolution*, *image synthesis*, *video generation*, dan lain-lain [4].

GAN telah terbukti menjadi sistem yang efisien untuk pembuatan data. Salah satu keuntungan utama GAN dibanding dengan metode *deep learning* yang lain adalah kemampuan mereka untuk menghasilkan data baru dari *noise* atau kebisingan, serta kemampuan mereka untuk secara virtual meniru distribusi data apa pun. Namun, menghasilkan data realistis menggunakan GAN tetap menjadi tantangan, terutama ketika fitur spesifik diperlukan, serta sensitif terhadap inisialisasi parameter dan hiper-parameter [5].

2.2 Style-GAN Model

Sebagian besar perbaikan telah dilakukan untuk model diskriminator dalam upaya untuk melatih model generator yang lebih efektif, namun masih sedikit upaya telah dilakukan untuk meningkatkan model generator. *Style Generative Adversarial Network*, atau singkatnya StyleGAN, adalah perluasan dari arsitektur GAN yang mengusulkan perubahan besar pada model generator, termasuk penggunaan jaringan pemetaan untuk memetakan titik-titik di ruang laten ke ruang laten menengah, penggunaan ruang laten antara untuk mengontrol gaya di setiap titik dalam model generator, dan pengenalan kebisingan sebagai sumber variasi di setiap titik dalam model generator. Model yang dihasilkan tidak hanya mampu menghasilkan citra wajah berkualitas tinggi yang mengesankan secara fotorealistik, tetapi juga menawarkan kontrol atas gaya gambar yang dihasilkan pada

berbagai tingkat detail dengan memvariasikan vektor gaya dan kebisingan [6].

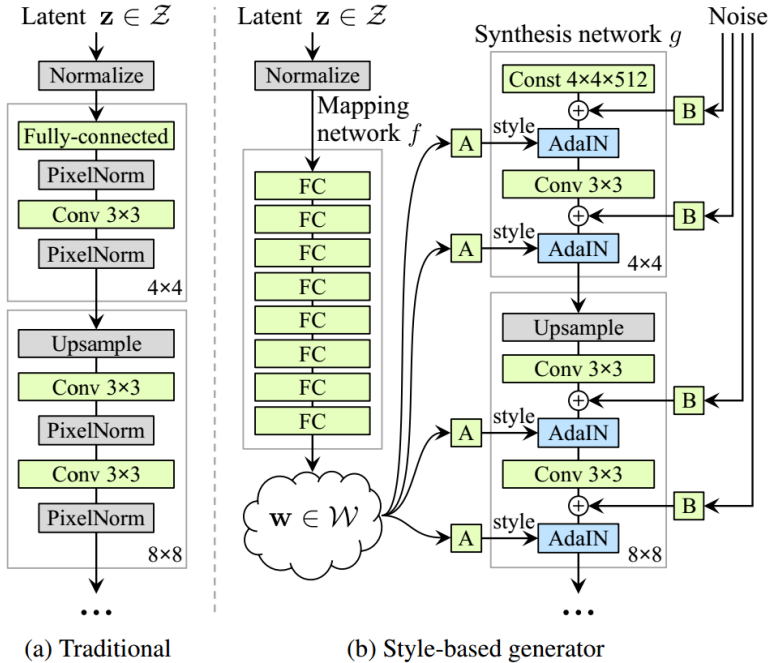
StyleGAN adalah pengembangan dari *progressive growing* GAN yang merupakan pendekatan untuk pelatihan model generator yang mampu mensintesis gambar berkualitas tinggi yang sangat besar melalui perluasan inkremental baik model diskriminator maupun generator dari gambar kecil ke besar selama proses pelatihan.

Selain semakin bertambahnya model selama pelatihan, StyleGAN mengubah arsitektur generator secara signifikan. Generator StyleGAN tidak lagi mengambil titik dari ruang laten sebagai input; sebaliknya, ada dua sumber acak yang digunakan untuk menghasilkan gambar sintetis: jaringan pemetaan mandiri dan lapisan noise. Output dari jaringan pemetaan adalah vektor yang mendefinisikan gaya yang terintegrasi pada setiap titik dalam model generator melalui lapisan baru yang disebut *adaptive instance normalization* (AdaIN) [6].

Penggunaan vektor gaya ini memberi kendali atas gaya gambar yang dihasilkan. Variasi stokastik diperkenalkan melalui kebisingan yang ditambahkan pada setiap titik dalam model generator. Kebisingan ditambahkan ke seluruh peta fitur yang memungkinkan model untuk menafsirkan gaya dengan cara yang halus, per-piksel. Penggabungan per-blok vektor gaya dan kebisingan ini memungkinkan setiap blok untuk melokalisasi interpretasi gaya dan variasi stokastik ke tingkat detail tertentu.

Arsitektur StyleGAN digambarkan sebagai arsitektur *progressive growing* GAN dengan lima modifikasi, masing-masing ditambahkan dan dievaluasi secara bertahap dalam studi ablatif. Modifikasi tersebut antara lain penambahan *tuning* dan *bilinear upsampling*, penambahan jaringan pemetaan (*mapping network*) dan AdaIN (*style*), penghapusan input vektor laten ke generator, penambahan *noise* untuk setiap blok, penambahan

mixing regularization [7]. Gambar 2.2 di bawah ini merangkum arsitektur generator StyleGAN.



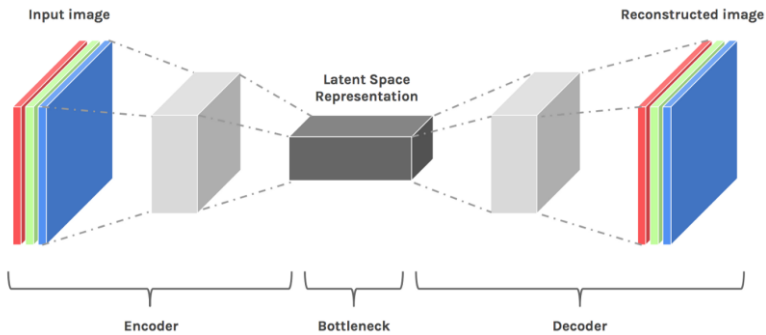
Gambar 2.2 Perbedaan arsitektur tradisional GAN dan StyleGAN [7]

2.3 Representasi Latent Space

Singkatnya *latent space* atau ruang laten merupakan representasi dari data yang dikompresi. Konsep ruang laten sangat penting karena utilitasnya merupakan inti dari *deep learning*, yaitu mempelajari fitur-fitur dari data dan menyederhanakan representasi data untuk tujuan menemukan pola [8].

Dalam *machine learning* sering kali kita mengompresi data untuk mempelajari informasi penting tentang titik data. Kompresi data adalah proses penyandian (*encoding*) informasi menggunakan bit yang lebih sedikit daripada representasi aslinya. Contoh

penggambaran pembuatan dari *latent space* dapat dilihat pada Gambar 2.3.



Gambar 2.3 Contoh penggambaran pembuatan *latent space* [8]

Kemudian, karena model diperlukan untuk merekonstruksi data terkompresi (*Decoding*), ia harus belajar untuk menyimpan semua informasi yang relevan dan mengabaikan kebisingan. Itulah kegunaan dari kompresi data, memungkinkan kita untuk menyingkirkan informasi asing, dan hanya fokus pada fitur yang paling penting. 'Kondisi terkompresi' ini adalah Representasi Ruang Laten dari data kita. Kita menyebutnya dengan ruang karena setiap titik data terkompresi secara unik dan ditentukan oleh titik-titik lain, sehingga dapat dibuat grafik datanya dengan bentuk ruang. Ruang laten tidak harus merupakan representasi dari vektor 2 atau 3 dimensi, oleh karenanya kita dapat menggunakan bantuan alat seperti *t-SNE* untuk mengubah representasi ruang laten dengan dimensi yang lebih tinggi ($n > 3$) dan tidak bisa dibayangkan biasa oleh manusia [8].

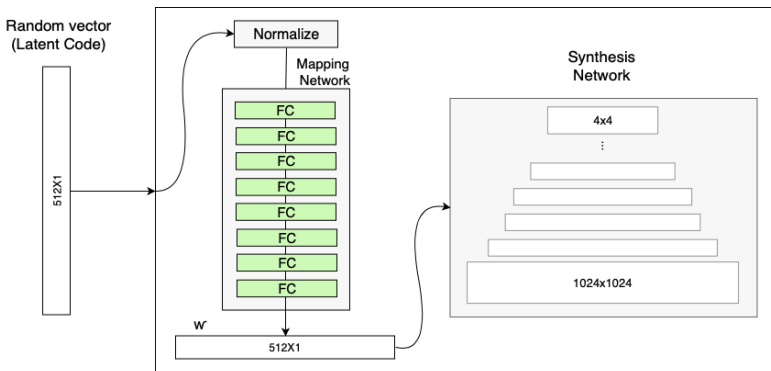
Contoh spesifik dari pembelajaran representasi ruang laten adalah *Manifold*, yaitu dalam *data science* dapat dipahami sebagai kelompok atau himpunan bagian data yang 'serupa' dalam beberapa hal. Kesamaan ini, biasanya tidak terlihat atau dikaburkan dalam ruang dimensi yang lebih tinggi, dapat ditemukan setelah data kami diwakili dalam ruang laten. Kemudian contoh lain di

dalam model *deep learning* adalah *Autoencoder* dan *Generative Model* dengan melakukan interpolasi pada ruang latent [9].

2.4 Mapping Network

Tujuan dari *Mapping Network* atau Jaringan Pemetaan adalah untuk menyandikan vektor (*latent space*) input ke dalam vektor perantara yang elemen-elemennya berbeda serta mengontrol fitur visual yang berbeda. Ini adalah proses yang tidak sepele karena kemampuan untuk mengontrol fitur visual dengan vektor input adalah terbatas, karena harus mengikuti kepadatan probabilitas data pelatihan. Misalnya, jika gambar orang dengan rambut hitam lebih umum atau banyak dalam dataset, maka lebih banyak nilai input akan dipetakan ke fitur tersebut. Akibatnya, model tidak mampu memetakan bagian dari input (elemen dalam vektor) ke dalam fitur, fenomena ini disebut *features entanglement*. Namun, dengan menggunakan *neural network* lain model dapat menghasilkan vektor yang tidak harus mengikuti distribusi data pelatihan dan dapat mengurangi korelasi antara fitur [10].

Mapping Network pada styleGAN terdiri dari 8 lapisan yang terhubung sepenuhnya (*fully connected layers*) dan outputnya w memiliki ukuran yang sama dengan lapisan input (512×1), seperti pada Gambar 2.4.

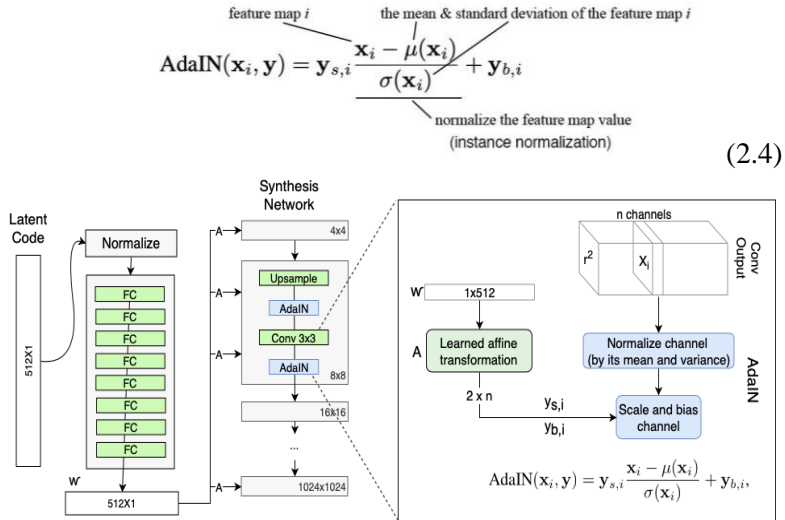


Gambar 2.4 Generator dengan *Mapping Network* [10]

2.5 Adaptive Instance Normalization (AdaIN)

Modul AdaIN mentransfer informasi yang disandikan w , yang dibuat oleh *Mapping Network*, ke dalam gambar yang dihasilkan. Modul ini ditambahkan ke setiap tingkat resolusi dari Jaringan Sintesis dan mendefinisikan ekspresi visual dari fitur-fitur di tingkat itu (Gambar 2.5):

- Langkah 1: Setiap saluran dari output lapisan konvolusi pertama dinormalisasi untuk memastikan penskalaan dan pergeseran dari langkah 3 memiliki efek yang diharapkan.
- Langkah 2: Vektor perantara w ditransformasikan menggunakan lapisan lain yang terhubung sepenuhnya (ditandai A) menjadi skala dan bias untuk setiap saluran.
- Langkah 3: Vektor skala dan bias menggeser setiap saluran dari output konvolusi, sehingga mendefinisikan pentingnya setiap filter dalam konvolusi. Penyesuaian ini menerjemahkan informasi dari w ke representasi visual.

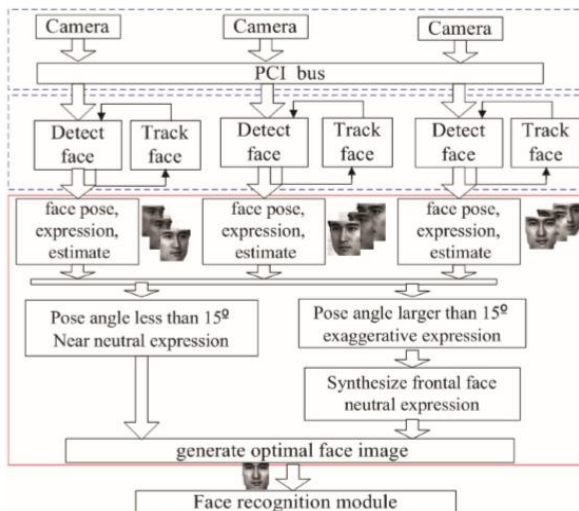


Gambar 2.5 Generator dengan *Adaptive Instance Normalization* (AdaIN) [10]

2.6 Face Recognition atau Pengenalan Wajah

Selama sepuluh tahun terakhir, *face recognition* atau pengenalan wajah telah menjadi bidang penelitian yang populer dalam visi komputer serta merupakan salah satu aplikasi analisis dan pemahaman gambar yang paling sukses [1]. Pernyataan umum tentang pengenalan wajah (dalam visi komputer) dapat dirumuskan sebagai berikut: Diberikan gambar video atau diam dari sebuah adegan, mengidentifikasi atau memverifikasi satu atau lebih orang dalam adegan tersebut menggunakan database yang tersimpan.

Biasanya ada tiga langkah yang ada dalam sistem pengenalan wajah; Akuisisi adalah pengenalan atau pengambilan deskripsi wajah dari berbagai pandangan; Normalisasi adalah segmentasi, pengurutan, dan konsistensi deskripsi wajah; *Recognition* atau pengenalan, yaitu melakukan ilustrasi, pemodelan deskripsi yang tidak dikenal dan menghubungkannya dengan model yang sudah dikenal untuk menawarkan identitas [1]. Contoh arsitektur yang digunakan untuk pengenalan wajah dapat dilihat pada Gambar 2.6.



Gambar 2.6 Contoh Arsitektur Sistem Pengenalan Wajah [2]

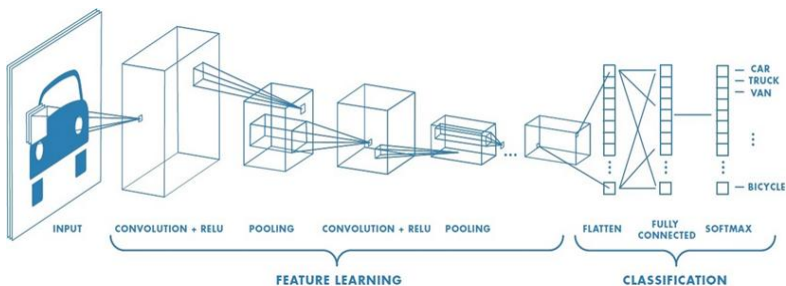
Untuk metode pengenalan wajah yang akan digunakan dalam pengerjaan Tugas Akhir ini adalah metode yang umum digunakan, yaitu *Convolutional Neural Network* (CNN) dengan menggunakan bahasa Pemrograman *Python*.

2.7 Convolutional Neural Network

Convolutional Neural Network (CNN) adalah salah satu jenis neural network yang biasa digunakan pada data citra. CNN bisa digunakan untuk mendeteksi dan mengenali objek pada citra.

Feature Learning / Extraction Layer adalah dimana terjadi proses penerjemahan dari sebuah citra menjadi *features*. *Features* ini berupa angka-angka yang merepresentasikan citra tersebut, yaitu berupa *feature map*. Proses ini disebut juga *Feature Extraction*. *Extraction Layer* terdiri dari dua yaitu, *Convolutional Layer* dan *Pooling Layer*.

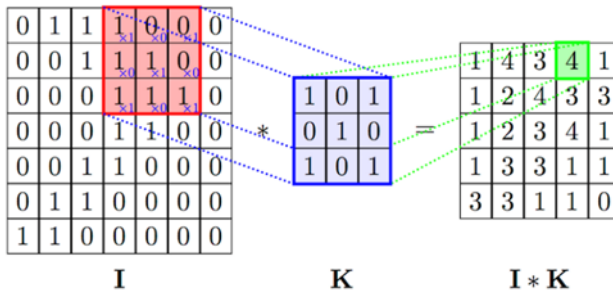
Classification Layer adalah dimana *feature map* yang dihasilkan dari *extraction layer* masih berbentuk array multidimensi sehingga harus dilakukan pengubahan *feature map* menjadi sebuah *feature vector* agar bisa digunakan sebagai masukan dari *fully connected layer*. *Fully connected layer* yang dimaksud disini adalah *Multilayer Perceptron* (MLP) yang memiliki beberapa *hidden layer*, *activation function*, *output layer* dan *loss function* [11]. Secara arsitektur, arsitektur CNN dapat dilihat pada Gambar 2.7.



Gambar 2.7 Contoh Arsitektur *Convolutional Neural Network* [12]

2.7.1 Convolutional Layer

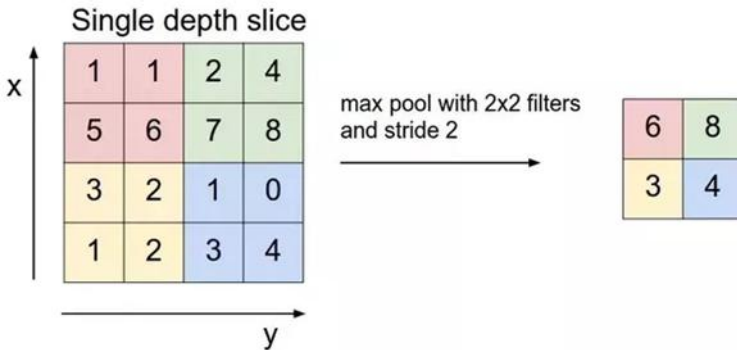
Convolution Layer melakukan operasi kovolusi pada output dari lapisan sebelumnya. Konvolusi adalah istilah matematis yang berarti mengaplikasikan sebuah fungsi pada output fungsi lain secara berulang. Tujuan dilakukannya konvolusi pada data citra adalah untuk mengekstraksi fitur dari citra input. Contoh cara kerja konvolusi bisa dilihat pada Gambar 2.8, dimana I adalah citra, K adalah *kernel*/filter yang digunakan. $I * K$ adalah hasil operasi konvolusi [11].



Gambar 2.8 Contoh proses *Convolution Layer* [11]

2.7.2 Pooling Layer

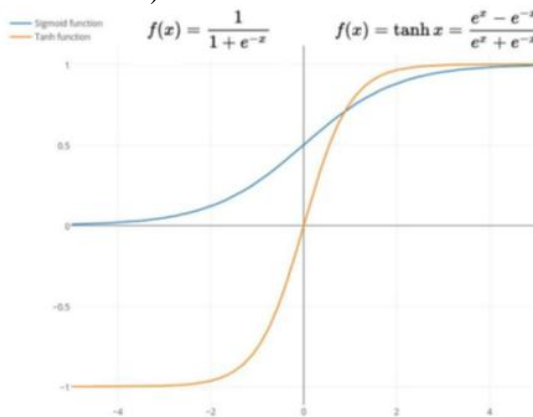
Fungsi dari *Pooling Layer* adalah mereduksi ukuran dari data. Terdapat beberapa tipe *Pooling Layer* diantaranya yaitu *max*, *average*, *sum* dan lainnya. Metode *Pooling* dalam CNN yang biasa digunakan adalah *Max Pooling*. *Max Pooling* membagi output dari *Convolutional Layer* menjadi beberapa matriks kecil lalu mengambil nilai maksimal dari tiap matriks untuk menyusun matriks citra yang telah direduksi. Proses tersebut memastikan fitur yang didapatkan akan sama meskipun objek citra mengalami translasi [11]. Contoh cara kerja *Max Pooling* bisa dilihat pada Gambar 2.9.



Gambar 2.9 Contoh proses *Pooling Layer (Max Pooling)* [11]

2.7.3 Activation Function

Fungsi aktivasi atau *Activation Function* berfungsi untuk menentukan apakah *neuron* tersebut harus aktif atau tidak berdasarkan dari jumlah bobot dari input. Secara umum terdapat dua jenis *activation function*, yaitu *Linear* dan *Non-Linear Activation Function*. Contoh dari *Non-Linear Activation Function* antara lain *Rectified Linear Unit (ReLU)*, *Softmax*, *Sigmoid*, dan *TanH*. (Gambar 2.10)



Gambar 2.10. Contoh *Non-Linear Activation Function*, *Sigmoid* dan *Tanh function* [13]

2.7.4 Loss Function

Loss Function dihitung oleh proses *forward pass*, yang tujuannya adalah memetakan pengaturan parameter ke nilai skalar, misalnya bobot *neural network* saat ini. Contoh *loss function* adalah *Softmax Loss* yang secara konsep sama dengan melakukan operasi *softmax* diikuti dengan menghitung *multinomial logistic loss*, namun menjamin nilai gradien yang lebih stabil. *Softmax Loss* digunakan untuk *K-class classification* [13].

2.7.5 Dropout

Dropout merupakan proses mencegah terjadinya *overfitting* dan juga mempercepat proses *learning*. *Dropout* mengacu kepada menghilangkan *neuron* yang berupa *hidden layer* maupun *visible layer* di dalam jaringan. Dengan menghilangkan suatu *neuron*, berarti menghilangkannya sementara dari jaringan yang ada. *Neuron* yang akan dihilangkan akan dipilih secara acak [13].

2.8 Confusion Matrix

Evaluasi terhadap kinerja suatu algoritma klasifikasi dilakukan untuk mengetahui seberapa baik algoritma dalam mengklasifikasikan data. *Confusion Matrix* merupakan salah satu metode yang dapat digunakan untuk mengukur kinerja suatu algoritma klasifikasi. Pada dasarnya *confusion matrix* mengandung informasi yang membandingkan hasil klasifikasi yang dilakukan oleh algoritma dengan hasil klasifikasi yang seharusnya, di mana outputnya bisa dua atau lebih kelas. Contoh *confusion matrix* dengan empat kombinasi nilai prediksi dan aktual yang berbeda dapat dilihat pada Gambar 2.11.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Gambar 2.11 Contoh *confusion matrix* dengan empat kombinasi nilai prediksi dan aktual yang berbeda [14]

Pada pengukuran kinerja menggunakan *confusion matrix*, terdapat empat istilah sebagai representasi hasil proses klasifikasi yaitu *True Positive (TP)*, *True Negative (TN)*, *False Positive (FP)* dan *False Negative (FN)*. Nilai *TP* merupakan jumlah data yang diprediksi positif dan itu adalah benar, sedangkan *TN* merupakan jumlah data yang diprediksi negatif dan itu adalah benar. Sementara *FP* merupakan jumlah data yang diprediksi positif padahal itu adalah salah, dan *FN* merupakan jumlah data yang diprediksi negatif padahal itu adalah salah. Nilai-nilai tersebut sangat berguna dalam pengukuran *Accuracy*, *Precision*, *Recall*, dan *F-Measure* [14].

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

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.5)$$

$$Precision = \frac{TP}{TP + FP} \quad (2.6)$$

$$Recall = \frac{TP}{TP + FN} \quad (2.7)$$

$$F - Measure = \frac{2 * Precision * Recall}{Precision + Recall} \quad (2.8)$$

2.9 Dataset

Dataset atau himpunan data merupakan kumpulan objek dan atributnya. Nama lain dari objek yang sering digunakan seperti *record*, *point*, *vector*, *pattern*, *event*, *observation*, *case*, *sample*, *instance*, dan entitas. Objek digambarkan dengan sejumlah atribut yang menerangkan sifat atau karakteristik dari objek tersebut. Sehingga atribut adalah sifat/properti/karakteristik objek yang nilainya bisa bermacam-macam dari satu objek dengan objek lainnya, dari satu waktu ke waktu yang lainnya.

Metode *machine learning* melakukan pembelajaran melalui contoh yang direpresentasikan melalui pengolahan dataset. Sehingga pada proses *learning* kita mengenal istilah *train* dataset dan *test* dataset. Pengertian dari *train* dataset adalah kumpulan data yang kita masukan ke dalam algoritma *machine learning* untuk membuat dan melatih model. Sedangkan, pengertian dari *test* dataset adalah kumpulan data yang kita gunakan untuk memvalidasi keakuratan suatu model tetapi hasilnya tidak digunakan untuk melatih model, *test* dataset juga biasa disebut dengan dataset validasi [15].

2.10 Transfer Learning

Transfer Learning adalah metode yang populer dalam visi komputer karena memungkinkan kita untuk membangun model yang akurat dengan cara yang menghemat waktu. Dengan *transfer learning*, kita tidak memulai proses pembelajaran dari awal, kita memulai dari pola yang telah dipelajari saat memecahkan masalah yang berbeda. Dengan cara ini kita memanfaatkan pembelajaran sebelumnya dan menghindari memulai dari awal [16].

Dalam visi komputer, *transfer learning* biasanya diekspresikan melalui penggunaan model pra-terlatih atau *pre-trained*. Model pra-terlatih adalah model yang dilatih tentang dataset dalam skala besar untuk menyelesaikan masalah yang serupa dengan yang ingin kita pecahkan. Karenanya, dengan biaya komputasi untuk pelatihan model-model tersebut yang besar, maka merupakan praktik umum untuk mengimpor dan menggunakan model dari literatur yang diterbitkan (misalnya StyleGAN, BIGGAN, DCGAN, VGG, *Inception*, *MobileNet*).

2.11 Python

Python adalah bahasa pemrograman yang populer. *Python* sering dimanfaatkan dalam pengembangan web, perangkat lunak, penelitian, dan *system scripting*. *Python* dapat digunakan untuk menangani data besar dan melakukan operasi matematika yang kompleks. *Python* bekerja di berbagai *platform* seperti *Windows*, *Mac*, *Linux*, *Raspberry Pi*, dan lain-lain. *Python* dirancang untuk mudah dibaca, yaitu memiliki *syntax* yang sederhana dan menggunakan bahasa inggris [17].

2.12 Library

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

2.12.1 Keras

Keras adalah *high-level neural networks API*, yang ditulis dalam bahasa pemrograman *Python* dan mampu berjalan di atas *TensorFlow* dan *Theano*. *Keras* dikembangkan dalam rangka memungkinkan eksperimen dilakukan dengan cepat. *Keras* dapat berjalan baik di CPU dan GPU. *Keras* berisi banyak implementasi *neural network* yang umum digunakan seperti, fungsi aktivasi,

optimizer, dan *tool* lain yang memudahkan dalam pengolahan citra dan data teks [18].

2.12.2 TensorFlow

TensorFlow adalah *library open source* untuk pembuatan program yang membutuhkan komputasi numerik berkinerja tinggi. *TensorFlow* dikembangkan oleh tim *Google Brain*. *TensorFlow* menyediakan fungsi-fungsi *machine learning* dan *deep learning*, dan dapat dijalankan dalam CPU atau GPU [19].

2.12.3 OpenCV

OpenCV (Open Source Computer Vision) adalah *library* yang dimanfaatkan dalam pengolahan citra dinamis secara *real-time*. *OpenCV* dapat digunakan dalam berbagai bahasa pemrograman seperti *Python*, *C++*, *Java*, atau *MATLAB*. *OpenCV* memiliki fitur seperti *Feature and Object Detection*, *Motion Analysis and Object Tracking*, *Image Filtering*, *Image Processing*, dan lain-lain [20].

2.12.4 Numpy

Numpy adalah *library Python* yang mendukung pengolahan data pada *array* dan matriks multidimensi yang besar. *Numpy* menyediakan kumpulan fungsi matematika, seperti aljabar linear, transformasi *Fourier*, pembuatan angka acak, dan lain-lain. *Numpy* bersifat *open source* sehingga banyak dimanfaatkan dalam pengolahan data penelitian [21].

2.12.5 Scikit-learn

Scikit-learn adalah *open source machine learning library* untuk bahasa pemrograman *Python*. *Scikit-learn* menyediakan fitur seperti *classification*, *regression*, *clustering*, termasuk juga di dalamnya algoritma *support vector machines*, *random forest*, *gradient boosting*, dan lain-lain [22].

2.12.6 Matplotlib

Matplotlib adalah *library Python* yang mendukung pembuatan grafik dua dimensi dalam berbagai format dan dari berbagai jenis data. *Matplotlib* bersifat *open source* dan banyak digunakan untuk pengolahan data dalam penelitian. *Matplotlib* dapat membuat plot, histogram, spektrum daya, diagram batang, diagram kesalahan, plot pencar, dan lain-lain [23].

2.12.7 Scikit-image

Scikit-image adalah *library Python* yang berisi kumpulan algoritma untuk pemrosesan gambar dan visi komputer. *Package* utama *Scikit-image* adalah menyediakan beberapa utilitas untuk mengkonversi antar tipe data gambar [24].

2.12.8 SciPy

SciPy adalah *library Python* gratis dan *open-source* yang digunakan untuk komputasi ilmiah dan komputasi teknis. *SciPy* berisi modul untuk optimasi, aljabar linear, integrasi, interpolasi, fungsi khusus, FFT, pemrosesan sinyal dan gambar, pemecah ODE, dan tugas-tugas lain yang umum dalam sains dan teknik [25].

(Halaman ini sengaja dikosongkan)

BAB III

PERANCANGAN SISTEM

Bab ini menjelaskan tentang perancangan data dan sistem sintesa citra wajah menggunakan metode *Generative Adversarial Network* (GAN) dan sistem pengenalan wajah menggunakan metode *Convolutional Neural Network* (CNN). Bab ini juga akan menjelaskan gambaran umum sistem dalam bentuk diagram alir.

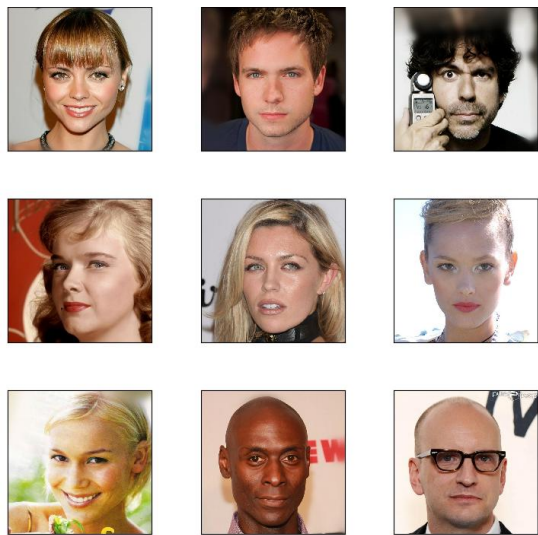
3.1 Perancangan Data

Dataset yang digunakan untuk proses pembuatan sintesa citra wajah menggunakan *Generative Adversarial Network* dibagi menjadi dua. Pertama adalah dataset referensi (diambil dari dataset CelebA-HQ dan FFHQ) yang telah digunakan untuk proses *learning* sebelumnya pada *pre-trained* model GAN agar mampu menghasilkan representasi *latent space* serta membentuknya kembali dengan kualitas tinggi. Kedua adalah sebagian dari dataset sumber dengan kondisi wajah netral pose lurus ke depan yang akan digunakan sebagai data uji pada implementasi *pre-trained* model GAN. Dari dataset sumber ini akan dilakukan penambahan gaya atau kondisi terhadap citra wajah dan hasilnya dapat ditambahkan pada data latih pengenalan wajah sebagai bentuk augmentasi data.

Dataset yang digunakan untuk uji coba pengenalan wajah adalah dari sebagian dataset sumber selain yang telah digunakan sebagai data uji pada implementasi *pre-trained* model GAN, dengan 6 variasi kondisi untuk setiap identitas wajah (ekspresi netral, ekspresi senyum, ekspresi sedih, pose kanan, pose kiri, dan memakai atribut kacamata).

Dataset *CelebFaces-Attributes* (CelebA) adalah dataset atribut wajah skala besar dari beberapa selebritas di dunia secara khusus yang disediakan oleh *Multimedia Laboratory* (MMLAB), *The Chinese University of Hong Kong*, yaitu berisi 202.599 citra wajah, 10.177 identitas, dan 5 lokasi *landmark* serta 40 anotasi atribut biner per gambarnya [26]. Kemudian untuk keperluan proses *learning* pada *pre-trained* model GAN maka dibuatlah versi

kualitas tinggi dari dataset CelebA tersebut serta diproses terlebih dahulu seperti disejajarkan dan dipotong secara otomatis oleh *NVIDIA Corporation* lalu disebut dengan dataset CelebA-HQ, yaitu terdiri dari 30.000 citra PNG dalam resolusi 1024×1024 serta dengan kanal warna RGB. Untuk contoh citra wajah pada dataset referensi CelebA-HQ dapat dilihat pada Gambar 3.1. Spesifikasi dari dataset referensi CelebA-HQ telah dirangkum pada Tabel 3.1.



Gambar 3.1 Contoh citra dari dataset referensi CelebA-HQ [26]

Tabel 3.1 Spesifikasi dari dataset referensi CelebA-HQ

Keterangan	Spesifikasi
Ukuran resolusi citra	1024×1024
Ekstensi	PNG
Kanal warna	RGB
Jumlah citra	30.000
Karakteristik	Untuk citra wajah selebritas dunia (Hollywood) secara khusus, variasi terbatas dalam hal usia dan etnis.

Dataset *Flickr-Faces-HQ* (FFHQ) adalah dataset citra wajah berkualitas tinggi dari beberapa orang di dunia secara umum yang disediakan oleh *NVIDIA Corporation*. Dataset ini didapatkan dari hasil menyaring atau *crawled* pada website *Flickr* serta diproses terlebih dahulu seperti disejajarkan dan dipotong secara otomatis oleh *NVIDIA Corporation*. Dataset FFHQ terdiri dari 70.000 citra PNG dalam resolusi 1024×1024 serta dengan kanal warna RGB, dataset ini juga memiliki variasi yang cukup besar dalam hal usia, etnis, dan latar belakang gambar. Dataset ini juga memiliki jangkauan yang baik dari aksesoris seperti kacamata, kacamata hitam, topi, dan lain-lain [27]. Untuk contoh citra wajah pada dataset referensi FFHQ dapat dilihat pada Gambar 3.2. Spesifikasi dari dataset referensi CelebA-HQ telah dirangkum pada Tabel 3.2.



Gambar 3.2 Contoh citra dari dataset referensi FFHQ [27]

Tabel 3.2 Spesifikasi dari dataset referensi FFHQ

Keterangan	Spesifikasi
Ukuran resolusi citra	1024×1024
Ekstensi	PNG
Kanal warna	RGB
Jumlah citra	70.000
Karakteristik	Untuk citra wajah orang di dunia secara umum, memiliki variasi yang besar dalam hal usia dan etnis.

Dataset sumber merupakan citra wajah yang dikumpulkan sendiri oleh pengusul (dari beberapa teman pengusul sebagai sampel). Data sumber ini terdiri dari 9 identitas wajah (termasuk gambar wajah pengusul dan teman-temannya) dengan resolusi dan tipe file yang berbeda (JPG, JPEG, PNG). Sehingga sebelum digunakan pada implementasi model GAN maupun pada pengenalan wajah maka pada data sumber tersebut perlu dilakukan praproses citra wajah untuk dipersiapkan dan disesuaikan dengan standar masukan dari model serta agar menunjang keberhasilan dari model. Untuk contoh citra wajah pada dataset sumber dapat dilihat pada Gambar 3.3. Dataset sumber secara lengkap dapat dilihat pada lampiran. Spesifikasi dari dataset sumber telah dirangkum pada Tabel 3.3.

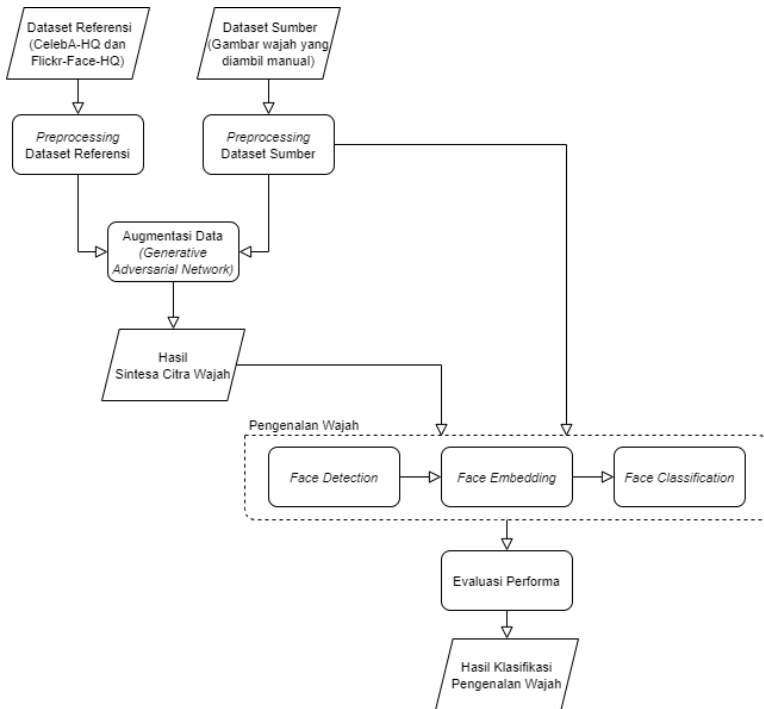


Gambar 3.3 Contoh citra dari dataset sumber untuk data uji GAN

Tabel 3.3 Spesifikasi dari dataset sumber

Keterangan	Spesifikasi
Ukuran resolusi citra	Bervariasi
Ekstensi	Bervariasi (JPG, JPEG, PNG)
Kanal warna	RGB
Jumlah citra	63
Karakteristik	Untuk citra wajah dari responden yang dikumpulkan sendiri oleh penulis, yaitu 9 identitas masing-masing 7 citra wajah dengan variasi kondisi netral, senyum, sedih, pose kanan, pose kiri, atribut kacamata.

3.2 Desain Umum Sistem



Gambar 3.4 Diagram alir sistem yang dibangun

Sistem yang dibangun memiliki dua proses utama yaitu Augmentasi Data dengan metode *Generative Adversarial Network* dan Pengenalan Wajah dengan metode *Convolutional Neural Network*. Diagram alir dari sistem ditunjukkan pada Gambar 3.4.

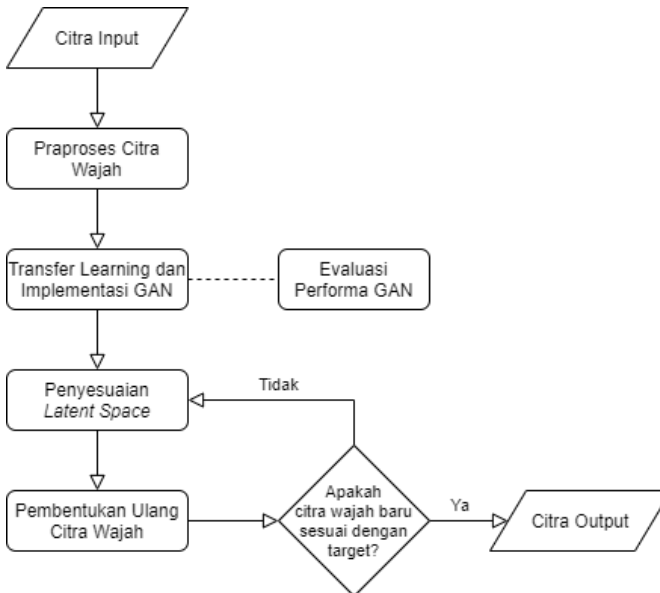
Proses Augmentasi Data terdiri dari beberapa tahap, yaitu pertama kita perlu melakukan praproses pada citra wajah dari input, kedua melakukan *transfer learning* dari *pre-trained* model GAN dan mengimplementasikannya untuk mendapatkan representasi *latent space* dari citra wajah, ketiga melakukan

penyesuaian *latent space*, kemudian keempat membentuk ulang citra wajah menggunakan *latent space* yang baru.

Proses Pengenalan Wajah terdiri dari beberapa tahap, yaitu melakukan *face detection and extraction*, *transfer learning* dari model *VGG Face Net* dan *face embedding*, pembuatan model untuk pengenalan wajah dan *fitting model*, kemudian terakhir *face classification* atau membuat prediksi dari citra wajah.

3.2.1 Proses Augmentasi Data

Pada tugas akhir ini, akan dilakukan proses Augmentasi Data yang berfungsi untuk melakukan sintesa citra wajah dari citra input agar menjadi citra output yang diharapkan. Tahap ini secara garis besar dibagi menjadi 4 tahapan, yaitu praproses citra wajah, *transfer learning* dan implementasi GAN, penyesuaian *latent space*, serta pembentukan ulang citra wajah. Diagram alir dari proses augmentasi data dapat dilihat pada Gambar 3.5.

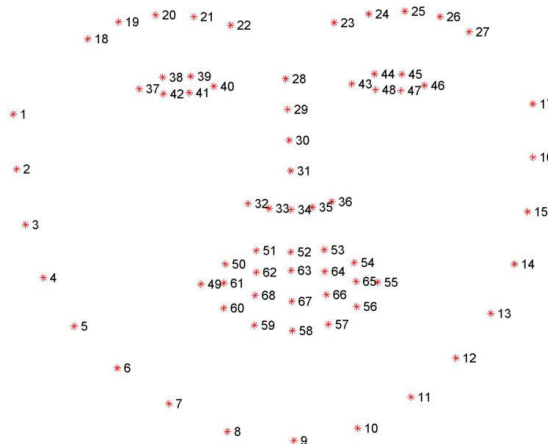


Gambar 3.5 Diagram alir proses augmentasi data

3.2.1.1 Praproses Citra Wajah

Pada tahap ini akan dilakukan praproses data citra input terlebih dahulu agar citra wajah siap untuk digunakan pada tahap berikutnya. Praproses yang dilakukan antara lain *face detection*, *face landmark*, *face alignment*, dan *resize*.

Untuk *face detection* dan *face landmark* menggunakan library dan model dari *DLib* (*histogram of oriented gradients* (HOG) and *linear SVM* untuk *face detector*, lalu *68-points pre-trained facial landmark detector* untuk *shape detector*) [28]. Contoh visualisasi *68-points facial landmark* dapat dilihat pada Gambar 3.6. Kemudian *face alignment* menggunakan fungsi dari *original FFHQ dataset preparation step* dan sekaligus akan di *resize* dengan ukuran 1024×1024 menyesuaikan standar input dari model GAN.



Gambar 3.6 Visualisasi *68-points facial landmark* [28]

3.2.1.2 Transfer Learning dan Implementasi GAN

Untuk jenis model GAN yang digunakan adalah StyleGAN, yaitu merupakan pengembangan GAN terbaru yang terkenal handal untuk melakukan *generate face*. Pada tahap ini dilakukan

transfer learning dari *pre-trained* dari StyleGAN pada dua dataset (CelebA-HQ dan FFHQ) dan akan dipilih *generator model* yang terbaik untuk digunakan dalam implementasi tahap ini (berdasarkan nilai *loss* pada uji coba kedua model *pre-trained*). Serta juga akan dilakukan uji coba dengan melakukan *fine tuning* pada parameter permukaan (*learning rate*, *batch size*, *iteration numbers*, dan *randomize noise*) pada model yang dipilih untuk menghasilkan output yang terbaik dan sesuai harapan.

Nilai *loss* yang dimaksud adalah hasil dari *Perceptual Model* yang bisa dikatakan adalah pengganti dari *Discriminator Model* pada GAN yang asli. Perbedaan utama antara keduanya adalah pada output yang dihasilkan. Jika pada *Discriminator Model* menghasilkan output biner (0/1) yang akan digunakan untuk melatih ulang *Generator Model* pada GAN, maka pada *Perceptual Model* menghasilkan nilai *loss* (menggunakan *mean squared error*) dalam membandingkan perbedaan (*latent space feature*) antara gambar asli dengan gambar yang dihasilkan oleh *Generator Model* dan digunakan untuk melatihnya kembali. *Perceptual Model* dibangun menggunakan model VGG16 yang disediakan pada *library keras* dan dioptimasi menggunakan *Gradient Descent Optimizer* dari *library tensorflow* [29].

Waktu pelatihan dari StyleGAN yang diharapkan untuk konfigurasi default menggunakan Tesla V100 GPU yang telah dicoba oleh NVIDIA dapat dilihat pada Tabel 3.4.

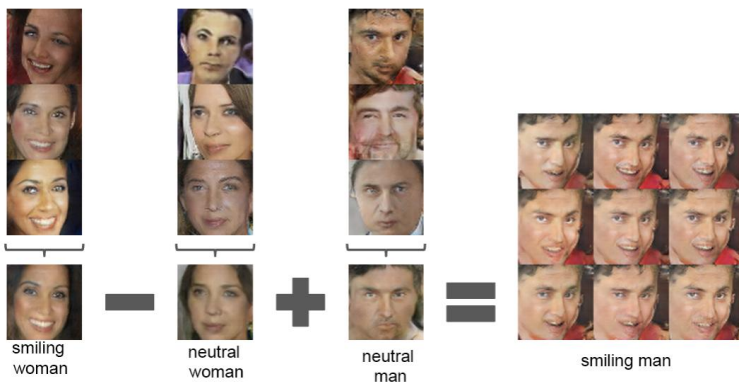
Tabel 3.4 Waktu pelatihan dari StyleGAN pada FFHQ dataset yang telah dicoba oleh NVIDIA [29]

GPUs	1024×1024	512×512	256×256
1	41 days 4 hours	24 days 21 hours	14 days 22 hours
2	21 days 22 hours	13 days 7 hours	9 days 5 hours
4	11 days 8 hours	7 days 0 hours	4 days 21 hours
8	6 days 14 hours	4 days 10 hours	3 days 8 hours

3.2.1.3 Penyesuaian Latent Space

Representasi *latent space* dari citra wajah awal dapat dirubah atau dilakukan penyesuaian sehingga ketika nantinya dibentuk ulang menggunakan *generator model* bisa menghasilkan citra wajah target atau seperti yang diharapkan (misal dengan perubahan ekspresi, pose rotasi wajah, dan penambahan atribut).

Terdapat beberapa metode yang digunakan pada tugas akhir ini yaitu *common average style mixing* (menggunakan koefisien *alpha* berupa bilangan desimal [0 sampai 1] untuk perkalian skalar terhadap *latent space* dan dapat diatur dengan mengubah nilai *alpha*), *vector operation in latent space* (menggunakan vektor untuk perkalian skalar terhadap *latent space* dan dapat diatur dengan mengubah nilai koefisien pangkat pada pembuatan vektor), *NVIDIA's style mixing technique* (menggunakan fungsi yang telah dibuat oleh NVIDIA untuk melakukan penggabungan gaya terhadap *latent space* dan dapat diatur dengan mengubah nilai *style range* [1-18]), dan *latent direction* (menggunakan *latent space* target sebagai arah untuk ditambahkan pada *latent space* awal dan dapat diatur dengan mengubah nilai koefisien perkalian terhadap *latent space* target).

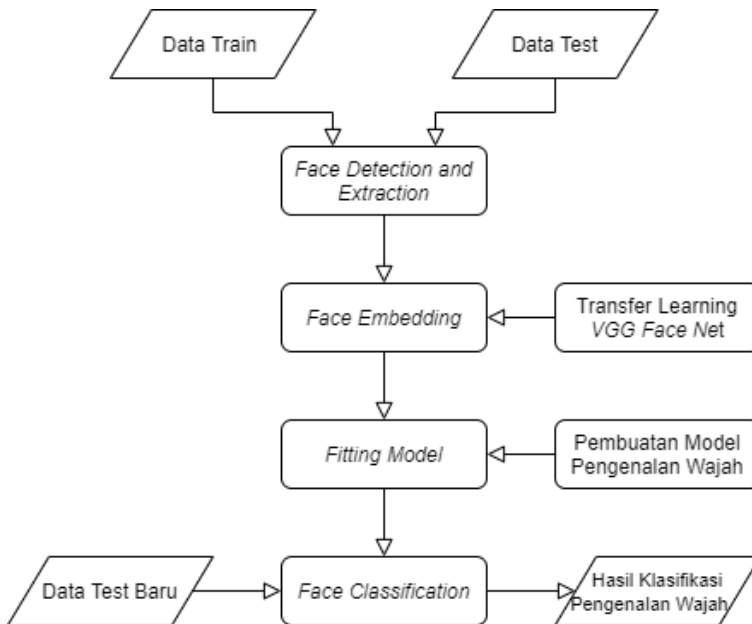


Gambar 3.7 Contoh operasi aritmatika pada vektor *latent space* [9]

3.2.1.4 Pembentukan Ulang Citra Wajah

Dengan memiliki representasi *latent space* tertentu dari sebuah citra wajah maka kita dapat membentuk ulang citra wajah tersebut menggunakan *generator model* yang dimiliki GAN. Model tersebut akan mentransformasi *latent space* tersebut kembali menjadi *matrix array* dari *pixel* citra wajah yang kemudian dengan bantuan *library PIL.Image* dapat menampilkannya menjadi sebuah citra wajah wajah dan menyimpannya.

3.2.2 Proses Pengenalan Wajah



Gambar 3.8 Diagram alir proses pengenalan wajah

Pada tugas akhir ini, akan dilakukan proses Pengenalan Wajah yang berfungsi untuk melakukan klasifikasi citra wajah berdasarkan identitasnya. Tahap ini secara garis besar dibagi menjadi 4 tahapan, yaitu *face detection and extraction*, *transfer learning VGG Face Net* dan *face embedding*, pembuatan model

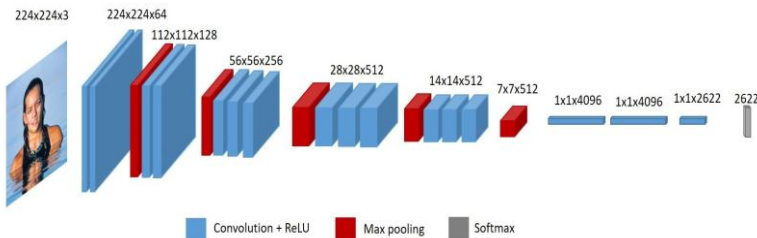
pengenalan wajah dan *fitting model*, kemudian terakhir *face classification*. Diagram alir dari proses pengenalan wajah dapat dilihat pada Gambar 3.8.

3.2.2.1 Face Detection and Extraction

Pada tahap ini dilakukan *face detection* menggunakan *Dlib CNN face detector* untuk menemukan bagian wajah dari sebuah citra. Berdasarkan hasil deteksi wajah tersebut selanjutnya akan dilakukan *face extraction* dengan cara mendapatkan titik-titik koordinat (*top, bottom, left, right*) yang kemudian digunakan untuk mengukur bagian citra yang akan dipotong.

3.2.2.2 Transfer Learning VGG Face Net dan Face Embedding

Transfer Learning VGG Face Net dilakukan dengan cara membuat arsitektur model dari *VGG Face Net* lalu memasukan *weight* dari *pre-trained vgg-face-model*. Visualisasi arsitektur dari *VGG Face Net* dapat dilihat pada Gambar 3.9.



Gambar 3.9 Visualisasi arsitektur *VGG Face Net* [30]

Kemudian akan dihapus *layer* Softmax terakhir dari arsitektur *VGG Face Net* untuk mendapatkan model hanya sampai lapisan rata (*flatten layer*) dengan output 2622 unit [31]. Sehingga model yang dihasilkan bisa digunakan untuk melakukan *Face Embedding* yaitu untuk mendapatkan ekstraksi fitur dari citra dengan bantuan *library tensorflow.keras.backend*.

3.2.2.3 Pembuatan Model Pengenalan Wajah dan *Fitting Model*

Model yang akan dibuat untuk melakukan pengenalan wajah adalah *simple softmax classifier* yang akan melakukan klasifikasi dari hasil *face embeddings* terhadap label identitas wajah. *Layer* terakhir dari model akan disesuaikan dengan jumlah identitas wajah yang akan dikenali. Kemudian model akan dilatih dengan jumlah *epoch*: 300, fungsi *activation*: *tanh* dan *softmax*, fungsi *loss*: *Sparse Categorical Crossentropy*, dan fungsi optimasi: *n-adam* [31].

3.2.2.4 *Face Classification*

Setelah selesai membuat model *classifier* pengenalan wajah dan melatihnya sesuai dengan dataset awal maka selanjutnya model *classifier* siap digunakan untuk mencoba pengenalan wajah terhadap data uji baru. Alur yang digunakan kurang lebih sama dengan tahap-tahap sebelumnya yaitu dengan mendeteksi dan mengekstrak citra wajah, kemudian mendapatkan fitur dari wajah atau *face embedding* dengan bantuan model *VGG Face Net*, lalu memasukkannya ke dalam model *classifier* pengenalan wajah, sehingga akan didapatkan hasil klasifikasi label identitas wajah. Selanjutnya akan ditambahkan kotak pembatas pada area wajah dengan dilengkapi label identitas wajah beserta nilai akurasi.

BAB IV IMPLEMENTASI

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

4.1 Lingkungan Implementasi

Dalam mengimplementasikan sistem yang akan dibangun, yaitu sintesa citra wajah menggunakan GAN untuk augmentasi data pada aplikasi pengenalan wajah, maka diperlukan beberapa perangkat pendukung sebagai berikut.

4.1.1 Perangkat Keras

Implementasi tugas akhir ini menggunakan Laptop *ASUS K401LB*. Sistem operasi yang digunakan adalah *Windows 10 64-bit*. Laptop yang digunakan memiliki spesifikasi *Intel® Core™ i5-5200* dengan kecepatan 2,2 GHz, *Random Access Memory* (RAM) sebesar 8 GB, dan mempunyai *Graphics Processing Unit* (GPU) yaitu *NVIDIA GeForce 940M* sebesar 6 GB.

Selain itu, tugas akhir ini juga menggunakan *Google Collaboratory*. Sistem operasi yang digunakan adalah *Ubuntu 18.04.3 LTS*. Cloud yang digunakan memiliki spesifikasi *Intel® Xeon®* 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.

4.1.2 Perangkat Lunak

Dari sisi perangkat lunak memiliki spesifikasi antara lain menggunakan bahasa pemrograman *Python 3.6*, dilengkapi dengan *library* antara lain *OpenCV*, *Tensorflow-GPU*, *Keras-GPU*, *PyTorch-GPU*, *Numpy*, *Pickle*, *Pandas*, *Matplotlib*, *DLib*, *DNNLib*, *Scikit-learn*, *PIL-image*, dan *SciPy*.

4.2 Implementasi Proses Augmentasi Data

Pada subbab ini akan dijelaskan proses augmentasi data menggunakan metode GAN untuk menghasilkan sintesa citra wajah sesuai target yang diharapkan.

4.2.1 Praproses Citra Wajah

Sebelum masuk ke dalam GAN, setiap citra pada dataset akan melalui praproses terlebih dahulu agar citra wajah siap untuk digunakan pada tahap berikutnya sesuai dengan standar masukan dari model GAN. Pada tahap praproses ini terdiri dari *face detection*, *face landmark*, *face alignment*, dan *resize*.

Untuk praproses citra wajah, implementasi dapat dilihat pada Kode Sumber 4.2.1. Untuk rincian setiap *class* atau *function* yang digunakan pada praproses citra wajah dapat dilihat berturut-turut pada Kode Sumber 4.2.2 dan Kode Sumber 4.2.3.

```

1. LANDMARKS_MODEL_URL =
   'http://dlib.net/files/shape_predictor_68_face_land
   marks.dat.bz2'
2.
3.
4. def unpack_bz2(src_path):
5.     data = bz2.BZ2File(src_path).read()
6.     dst_path = src_path[:-4]
7.     with open(dst_path, 'wb') as fp:
8.         fp.write(data)
9.     return dst_path
10.
11.
12. if __name__ == "__main__":
13.     """
14.     Extracts and aligns all faces from images using
       DLib and a function from original FFHQ dataset
       preparation step
15.     python align_images.py /raw_images
       /aligned_images
16.     """
17.

```

```

18.     landmarks_model_path =
        unpack_bz2(get_file('shape_predictor_68_face_landma
        rks.dat.bz2'),
19.
        LANDMARKS_MODEL_URL, cache_subdir='temp'))
20.     RAW_IMAGES_DIR = sys.argv[1]
21.     ALIGNED_IMAGES_DIR = sys.argv[2]
22.
23.     landmarks_detector =
        LandmarksDetector(landmarks_model_path)
24.     for img_name in os.listdir(RAW_IMAGES_DIR):
25.         raw_img_path = os.path.join(RAW_IMAGES_DIR,
            img_name)
26.         for i, face_landmarks in
            enumerate(landmarks_detector.get_landmarks(raw_img_
            path), start=1):
27.             face_img_name = '%s.png' %
                (os.path.splitext(img_name)[0])
28.             aligned_face_path =
                os.path.join(ALIGNED_IMAGES_DIR, face_img_name)
29.
30.             image_align(raw_img_path,
                aligned_face_path, face_landmarks)

```

Kode Sumber 4.2.1 Implementasi Praproses Citra Wajah

Untuk *class LandmarksDetector*, implementasi dapat dilihat pada Kode Sumber 4.2.2.

```

1. class LandmarksDetector:
2.     def __init__(self, predictor_model_path):
3.         """
4.         :param predictor_model_path: path to
        shape_predictor_68_face_landmarks.dat file
5.         """
6.         self.detector =
            dlib.get_frontal_face_detector() #
            cnn_face_detection_model_v1 also can be used
7.         self.shape_predictor =
            dlib.shape_predictor(predictor_model_path)
8.
9.     def get_landmarks(self, image):

```

```

10.         img = dlib.load_rgb_image(image)
11.         dets = self.detector(img, 1)
12.
13.         for detection in dets:
14.             face_landmarks = [(item.x, item.y) for
item in self.shape_predictor(img,
detection).parts()]
15.         yield face_landmarks

```

Kode Sumber 4.2.2 Implementasi *class LandmarksDetector*

Untuk *function image_align*, implementasi dapat dilihat pada Kode Sumber 4.2.3.

```

1. def image_align(src_file, dst_file, face_landmarks,
output_size=1024, transform_size=4096,
enable_padding=True):
2.     # Align function from FFHQ dataset pre-
processing step
3.     # https://github.com/NVlabs/ffhq-
dataset/blob/master/download\_ffhq.py
4.
5.     lm = np.array(face_landmarks)
6.     lm_chin          = lm[0 : 17] # left-
right
7.     lm_eyebrow_left  = lm[17 : 22] # left-
right
8.     lm_eyebrow_right = lm[22 : 27] # left-
right
9.     lm_nose          = lm[27 : 31] # top-down
10.    lm_nostrils       = lm[31 : 36] # top-down
11.    lm_eye_left       = lm[36 : 42] # left-
clockwise
12.    lm_eye_right      = lm[42 : 48] # left-
clockwise
13.    lm_mouth_outer    = lm[48 : 60] # left-
clockwise
14.    lm_mouth_inner    = lm[60 : 68] # left-
clockwise
15.
16.    # Calculate auxiliary vectors.
17.    eye_left          = np.mean(lm_eye_left, axis=0)

```

```

18.         eye_right    = np.mean(lm_eye_right,
axis=0)
19.         eye_avg       = (eye_left + eye_right) * 0.5
20.         eye_to_eye    = eye_right - eye_left
21.         mouth_left    = lm_mouth_outer[0]
22.         mouth_right   = lm_mouth_outer[6]
23.         mouth_avg     = (mouth_left + mouth_right) *
0.5
24.         eye_to_mouth = mouth_avg - eye_avg
25.
26.         # Choose oriented crop rectangle.
27.         x = eye_to_eye - np.flipud(eye_to_mouth) *
[-1, 1]
28.         x /= np.hypot(*x)
29.         x *= max(np.hypot(*eye_to_eye) * 2.0,
np.hypot(*eye_to_mouth) * 1.8)
30.         y = np.flipud(x) * [-1, 1]
31.         c = eye_avg + eye_to_mouth * 0.1
32.         quad = np.stack([c - x - y, c - x + y, c +
x + y, c + x - y])
33.         qsize = np.hypot(*x) * 2
34.
35.         # Load in-the-wild image.
36.         if not os.path.isfile(src_file):
37.             print('\nCannot find source image.
Please run "--wilds" before "--align".')
38.             return
39.         img = PIL.Image.open(src_file)
40.
41.         # Shrink.
42.         shrink = int(np.floor(qsize / output_size *
0.5))
43.         if shrink > 1:
44.             rsize = (int(np rint(float(img.size[0])
/ shrink)), int(np rint(float(img.size[1]) /
shrink)))
45.             img = img.resize(rsize,
PIL.Image.ANTIALIAS)
46.             quad /= shrink
47.             qsize /= shrink
48.
49.         # Crop.
50.         border = max(int(np rint(qsize * 0.1)), 3)

```

```

51.         crop = (int(np.floor(min(quad[:,0]))),
int(np.floor(min(quad[:,1]))),
int(np.ceil(max(quad[:,0]))),
int(np.ceil(max(quad[:,1]))))
52.         crop = (max(crop[0] - border, 0),
max(crop[1] - border, 0), min(crop[2] + border,
img.size[0]), min(crop[3] + border, img.size[1]))
53.         if crop[2] - crop[0] < img.size[0] or
crop[3] - crop[1] < img.size[1]:
54.             img = img.crop(crop)
55.             quad -= crop[0:2]
56.
57.         # Pad.
58.         pad = (int(np.floor(min(quad[:,0]))),
int(np.floor(min(quad[:,1]))),
int(np.ceil(max(quad[:,0]))),
int(np.ceil(max(quad[:,1]))))
59.         pad = (max(-pad[0] + border, 0), max(-
pad[1] + border, 0), max(pad[2] - img.size[0] +
border, 0), max(pad[3] - img.size[1] + border, 0))
60.         if enable_padding and max(pad) > border -
4:
61.             pad = np.maximum(pad, int(np rint(qsize
* 0.3)))
62.             img = np.pad(np.float32(img), ((pad[1],
pad[3]), (pad[0], pad[2])), (0, 0)), 'reflect')
63.             h, w, _ = img.shape
64.             y, x, _ = np.ogrid[:h, :w, :1]
65.             mask = np.maximum(1.0 -
np.minimum(np.float32(x) / pad[0], np.float32(w-1-
x) / pad[2]), 1.0 - np.minimum(np.float32(y) /
pad[1], np.float32(h-1-y) / pad[3]))
66.             blur = qsize * 0.02
67.             img +=
(scipy.ndimage.gaussian_filter(img, [blur, blur,
0]) - img) * np.clip(mask * 3.0 + 1.0, 0.0, 1.0)
68.             img += (np.median(img, axis=(0,1)) -
img) * np.clip(mask, 0.0, 1.0)
69.             img =
PIL.Image.fromarray(np.uint8(np.clip(np rint(img),
0, 255))), 'RGB')
70.             quad += pad[:2]
71.

```

```

72.         # Transform.
73.         img = img.transform((transform_size,
                               transform_size), PIL.Image.QUAD, (quad +
                               0.5).flatten(), PIL.Image.BILINEAR)
74.         if output_size < transform_size:
75.             img = img.resize((output_size,
                                output_size), PIL.Image.ANTIALIAS)
76.
77.         # Save aligned image.
78.         img.save(dst_file, 'PNG')

```

Kode Sumber 4.2.3 Implementasi *function image_align*

4.2.2 Transfer Learning dan Implementasi GAN

Pada bagian ini akan dijabarkan mengenai metode *transfer learning* dari *pre-trained* StyleGAN pada CelebA-HQ dataset dan FFHQ dataset. Implementasi *transfer learning pre-trained* StyleGAN tercantum pada Kode Sumber 4.2.4. Untuk rincian setiap model *Generator* dan *Perceptual* dapat dilihat berturut-turut pada Kode Sumber 4.2.5 dan Kode Sumber 4.2.6.

```

1. # URL_FFHQ =
   'https://drive.google.com/uc?id=1MEGjdvVpUsu1jB4zrX
   ZN7Y4kBB0zizDQ' # karras2019stylegan-ffhq-
   1024x1024.pkl
2. # URL_CelebAHQ =
   'https://drive.google.com/uc?id=1MGqJl28pN4t7SAtSrP
   dSRJSQJqahkzUf' # karras2019stylegan-celebahq-
   1024x1024.pkl
3.
4. def split_to_batches(l, n):
5.     for i in range(0, len(l), n):
6.         yield l[i:i + n]
7.
8.
9. def main():
10.     parser =
        argparse.ArgumentParser(description='Find latent
        representation of reference images using perceptual
        loss')

```

```

11.     parser.add_argument('src_dir', help='Directory
    with images for encoding')
12.     parser.add_argument('generated_images_dir',
    help='Directory for storing generated images')
13.     parser.add_argument('dlatent_dir',
    help='Directory for storing dlatent
    representations')
14.
15.     # for now it's unclear if larger batch leads to
    better performance/quality
16.     parser.add_argument('--batch_size', default=1,
    help='Batch size for generator and perceptual
    model', type=int)
17.
18.     # Perceptual model params
19.     parser.add_argument('--image_size',
    default=256, help='Size of images for perceptual
    model', type=int)
20.     parser.add_argument('--lr', default=1.,
    help='Learning rate for perceptual model',
    type=float)
21.     parser.add_argument('--iterations',
    default=1000, help='Number of optimization steps
    for each batch', type=int)
22.
23.     # Generator params
24.     parser.add_argument('--randomize_noise',
    default=False, help='Add noise to dlatents during
    optimization', type=bool)
25.     args, other_args = parser.parse_known_args()
26.
27.     ref_images = [os.path.join(args.src_dir, x) for
    x in os.listdir(args.src_dir)]
28.     ref_images = list(filter(os.path.isfile,
    ref_images))
29.
30.     if len(ref_images) == 0:
31.         raise Exception('%s is empty' %
    args.src_dir)
32.
33.     os.makedirs(args.generated_images_dir,
    exist_ok=True)
34.     os.makedirs(args.dlatent_dir, exist_ok=True)

```



```

35.
36.     ## Initialize generator and perceptual model
37.     tflib.init_tf()
38.     # with dnnlib.util.open_url(URL_FFHQ,
cache_dir=config.cache_dir) as f:
39.         # generator_network, discriminator_network,
Gs_network = pickle.load(f)
40.         print(os.path.isfile("karras2019stylegan-ffhq-
1024x1024.pkl")) #FFHQ dataset
41.         generator_network, discriminator_network,
Gs_network = pickle.load(open( "karras2019stylegan-
ffhq-1024x1024.pkl", "rb"))
42.         # print(os.path.isfile("karras2019stylegan-
celebahq-1024x1024.pkl")) #celebaHQ dataset
43.         # generator_network, discriminator_network,
Gs_network = pickle.load(open( "karras2019stylegan-
celebahq-1024x1024.pkl", "rb"))
44.
45.         generator = Generator(Gs_network,
args.batch_size,
randomize_noise=args.randomize_noise)
46.         perceptual_model =
PerceptualModel(args.image_size, layer=9,
batch_size=args.batch_size)
47.
perceptual_model.build_perceptual_model(generator.g
enerated_image)
48.
49.         # Optimize (only) dlatents by minimizing
perceptual loss between reference and generated
images in feature space
50.         for images_batch in
tqdm(split_to_batches(ref_images, args.batch_size),
total=len(ref_images)//args.batch_size):
51.             names =
[os.path.splitext(os.path.basename(x))[0] for x in
images_batch]
52.
53.
perceptual_model.set_reference_images(images_batch)
54.             op =
perceptual_model.optimize(generator.dlatent_variabl

```

```

    e, iterations=args.iterations,
    learning_rate=args.lr)
55.     # log = open(' '.join(names)+'.txt',"a")
56.     log = []
57.     np.array(log)
58.     pbar = tqdm(op, leave=False,
    total=args.iterations)
59.     for loss in pbar:
60.         pbar.set_description(' '.join(names)+'
    Loss: %.2f' % loss)
61.         # print('\nloss:', loss)
62.         # log.write(str(loss)+'\n')
63.         log = np.append(log, loss)
64.         print(' '.join(names), ' loss:', loss)
65.         # print(type(loss))
66.         # log.close()
67.         logs = pd.DataFrame(log, columns=['loss'])
68.         logs.to_csv('loss_history/'+
    ' '.join(names)+'.txt', header=True, index=True,
    sep='\t', mode='w')
69.         # print(logs.dtypes)
70.         # print(logs)
71.         # # Plot the data
72.         # plt.plot(log, label='LOSS')
73.         # # Add a legend
74.         # plt.legend()
75.         # # Show the plot
76.         # print(plt.show())
77.
78.         # Generate images from found dlatents and
    save them
79.         generated_images =
    generator.generate_images()
80.         generated_dlatents =
    generator.get_dlatents()
81.         for img_array, dlatent, img_name in
    zip(generated_images, generated_dlatents, names):
82.             img = PIL.Image.fromarray(img_array,
    'RGB')
83.
    img.save(os.path.join(args.generated_images_dir,
    f'{img_name}.png'), 'PNG')

```

```

84.         np.save(os.path.join(args.dlatent_dir,
    f'{img_name}.npy'), dlatent)
85.
86.         generator.reset_dlatents()
87.
88.
89.
90. if __name__ == "__main__":
91.     main()

```

Kode Sumber 4.2.4 Implementasi *transfer learning pre-trained* StyleGAN

Untuk model *Generator*, implementasi dapat dilihat pada Kode Sumber 4.2.5.

```

1. def create_stub(name, batch_size):
2.     return tf.constant(0, dtype='float32',
    shape=(batch_size, 0))
3.
4.
5. def create_variable_for_generator(name,
    batch_size):
6.     return tf.get_variable('learnable_dlatents',
7.                             shape=(batch_size, 18,
    512),
8.                             dtype='float32',
9.                             initializer=tf.initializers.random_normal())
10.
11.
12. class Generator:
13.     def __init__(self, model, batch_size,
    randomize_noise=False):
14.         self.batch_size = batch_size
15.
16.         self.initial_dlatents =
    np.zeros((self.batch_size, 18, 512))
17.
    model.components.synthesis.run(self.initial_dlatent
    s,

```

```

18.         randomize_noise=randomize_noise,
           minibatch_size=self.batch_size,
19.         custom_inputs=[partial(create_variable_for_generato
                                r, batch_size=batch_size),
20.         partial(create_stub, batch_size=batch_size)],
21.         structure='fixed')
22.
23.         self.sess = tf.get_default_session()
24.         self.graph = tf.get_default_graph()
25.
26.         self.dlatent_variable = next(v for v in
tf.global_variables() if 'learnable_dlatents' in
v.name)
27.         self.set_dlatents(self.initial_dlatents)
28.
29.         self.generator_output =
self.graph.get_tensor_by_name('G_synthesis_1/Run/c
oncat/concat:0')
30.         self.generated_image =
tf.image.convert_images_to_uint8(self.generator_output
, nchw_to_nhwc=True, uint8_cast=False)
31.         self.generated_image_uint8 =
tf.image.saturate_cast(self.generated_image, tf.uint8)
32.
33.         def reset_dlatents(self):
34.             self.set_dlatents(self.initial_dlatents)
35.
36.         def set_dlatents(self, dlatents):
37.             assert (dlatents.shape == (self.batch_size,
18, 512))
38.
self.sess.run(tf.assign(self.dlatent_variable,
dlatents))
39.
40.         def get_dlatents(self):
41.             return self.sess.run(self.dlatent_variable)
42.
43.         def generate_images(self, dlatents=None):
44.             if dlatents:

```

```

45.         self.set_dlatents(dlatents)
46.         return
        self.sess.run(self.generated_image_uint8)

```

Kode Sumber 4.2.5 Implementasi model *Generator*

Untuk model *Perceptual*, implementasi dapat dilihat pada Kode Sumber 4.2.6.

```

1. def load_images(images_list, img_size):
2.     loaded_images = list()
3.     for img_path in images_list:
4.         img = image.load_img(img_path,
5.                               target_size=(img_size, img_size))
6.         img = np.expand_dims(img, 0)
7.         loaded_images.append(img)
8.     loaded_images = np.vstack(loaded_images)
9.     preprocessed_images =
10.     preprocess_input(loaded_images)
11.     return preprocessed_images
12.
13. class PerceptualModel:
14.     def __init__(self, img_size, layer=9,
15.                  batch_size=1, sess=None):
16.         self.sess = tf.get_default_session() if
17.         sess is None else sess
18.         K.set_session(self.sess)
19.         self.img_size = img_size
20.         self.layer = layer
21.         self.batch_size = batch_size
22.
23.         self.perceptual_model = None
24.         self.ref_img_features = None
25.         self.features_weight = None
26.         self.loss = None
27.
28.     def build_perceptual_model(self,
29.                               generated_image_tensor):
30.         vgg16 = VGG16(include_top=False,
31.                        input_shape=(self.img_size, self.img_size, 3))

```

```

27.         self.perceptual_model = Model(vgg16.input,
vgg16.layers[self.layer].output)
28.         generated_image =
preprocess_input(tf.image.resize_images(generated_i
image_tensor,
29. (self.img_size, self.img_size), method=1))
30.         generated_img_features =
self.perceptual_model(generated_image)
31.
32.         self.ref_img_features =
tf.get_variable('ref_img_features',
shape=generated_img_features.shape,
33. dtype='float32',
initializer=tf.initializers.zeros())
34.         self.features_weight =
tf.get_variable('features_weight',
shape=generated_img_features.shape,
35. dtype='float32',
initializer=tf.initializers.zeros())
36.
self.sess.run([self.features_weight.initializer,
self.features_weight.initializer])
37.
38.         self.loss =
tf.losses.mean_squared_error(self.features_weight *
self.ref_img_features,
39. self.features_weight * generated_img_features) /
82890.0
40.
41.     def set_reference_images(self, images_list):
42.         assert(len(images_list) != 0 and
len(images_list) <= self.batch_size)
43.         loaded_image = load_images(images_list,
self.img_size)
44.         image_features =
self.perceptual_model.predict_on_batch(loaded_image
)
45.

```

```

46.         # in case if number of images less than
        actual batch size
47.         # can be optimized further
48.         weight_mask =
        np.ones(self.features_weight.shape)
49.         if len(images_list) != self.batch_size:
50.             features_space =
        list(self.features_weight.shape[1:])
51.             existing_features_shape =
        [len(images_list)] + features_space
52.             empty_features_shape = [self.batch_size
        - len(images_list)] + features_space
53.
54.             existing_examples =
        np.ones(shape=existing_features_shape)
55.             empty_examples =
        np.zeros(shape=empty_features_shape)
56.             weight_mask =
        np.vstack([existing_examples, empty_examples])
57.
58.             image_features =
        np.vstack([image_features,
        np.zeros(empty_features_shape)])
59.
60.
        self.sess.run(tf.assign(self.features_weight,
        weight_mask))
61.
        self.sess.run(tf.assign(self.ref_img_features,
        image_features))
62.
63.         def optimize(self, vars_to_optimize,
        iterations=500, learning_rate=1.):
64.             vars_to_optimize = vars_to_optimize if
        isinstance(vars_to_optimize, list) else
        [vars_to_optimize]
65.             optimizer =
        tf.train.GradientDescentOptimizer(learning_rate=lea
        rning_rate)
66.             min_op = optimizer.minimize(self.loss,
        var_list=[vars_to_optimize])
67.             for _ in range(iterations):

```

```

68.         _, loss = self.sess.run([min_op,
        self.loss])
69.         yield loss

```

Kode Sumber 4.2.6 Implementasi model *Perceptual*

4.2.3 Penyesuaian Latent Space

Setelah didapatkan representasi *latent space* dari citra wajah, maka selanjutnya dilakukan penyesuaian *latent space* untuk menghasilkan citra wajah target atau seperti yang diharapkan. Ada beberapa metode yang digunakan pada tugas akhir ini (seperti yang sudah dijelaskan pada Bab III) yaitu *common average style mixing*, *vector operation in latent space*, *NVIDIA's style mixing technique*, dan *latent direction*.

Implementasi untuk *common average style mixing* dapat dilihat pada Kode Sumber 4.2.7.

```

1. alpha = 0.5
2. mix_latent = (((alpha)*latent_source)+((1-
    alpha)*latent_target))

```

Kode Sumber 4.2.7 Implementasi *common average style mixing*

Implementasi untuk *vector operation in latent space* dapat dilihat pada Kode Sumber 4.2.8.

```

1. model_res = 1024
2. model_scale = int(2*(math.log(model_res,2)-1))
3. lr = ((np.arange(1,model_scale+1)/model_scale)**0.70).re
    shape((model_scale,1))
4. r1 = 1-lr
5. generate_image(lr*latent_source+r1*latent_target)
6. generate_image(lr*latent_target+r1*latent_source)

```

Kode Sumber 4.2.8 Implementasi *vector operation in latent space*

Implementasi untuk *NVIDIA's style mixing technique* dapat dilihat pada Kode Sumber 4.2.9.

```

1. def draw_style_mixing_figure(png, Gs, w, h,
   src_dlatents, dst_dlatents, style_ranges):
2.     print(png)
3.     #src_dlatents =
   Gs.components.mapping.run(src_latents, None) #
   [seed, layer, component]
4.     #dst_dlatents =
   Gs.components.mapping.run(dst_latents, None)
5.     src_images =
   Gs.components.synthesis.run(src_dlatents,
   randomize_noise=False, **synthesis_kwargs)
6.     dst_images =
   Gs.components.synthesis.run(dst_dlatents,
   randomize_noise=False, **synthesis_kwargs)
7.
8.     canvas = PIL.Image.new('RGB', (w *
   (len(src_dlatents) + 1), h * (len(dst_dlatents) +
   1)), 'white')
9.     for col, src_image in
   enumerate(list(src_images)):
10.        canvas.paste(PIL.Image.fromarray(src_image,
   'RGB'), ((col + 1) * w, 0))
11.    for row, dst_image in
   enumerate(list(dst_images)):
12.        canvas.paste(PIL.Image.fromarray(dst_image,
   'RGB'), (0, (row + 1) * h))
13.        row_dlatents = np.stack([dst_dlatents[row]]
   * len(src_dlatents))
14.        row_dlatents[:, style_ranges[row]] =
   src_dlatents[:, style_ranges[row]]
15.        row_images =
   Gs.components.synthesis.run(row_dlatents,
   randomize_noise=False, **synthesis_kwargs)
16.    for col, image in
   enumerate(list(row_images)):
17.        canvas.paste(PIL.Image.fromarray(image,
   'RGB'), ((col + 1) * w, (row + 1) * h))
18.    canvas.save(png)
19.    return canvas.resize((512,512))

```

```

20. synthesis_kwargs = dict(output_transform=dict(func=tflib.convert_image
    s_to_uint8, nchw_to_nhwc=True), minibatch_size=1)
    #minibatch_size=8
21. _Gs_cache = dict()
22. draw_style_mixing_figure(os.path.join(config.result
    _dir, 'style-mixing.png'), Gs, w=1024, h=1024,
    src_dlatents=laten_source.reshape((1, 18, 512)),
    dst_dlatents=latent_target.reshape((1, 18, 512)),
    style_ranges=[range(6,14)])

```

Kode Sumber 4.2.9 Implementasi *NVIDIA's style mixing technique*

Implementasi untuk *latent direction* dapat dilihat pada Kode Sumber 4.2.10.

```

1. def move_and_show(latent_vector, direction, coeffs):
2.     fig, ax = plt.subplots(1, len(coeffs),
    figsize=(15, 10), dpi=80)
3.     for i, coeff in enumerate(coeffs):
4.         new_latent_vector = latent_vector.copy()
5.         new_latent_vector[:8] = (latent_vector +
    coeff*direction)[:8]
6.
7.         ax[i].imshow(generate_image(new_latent_vector))
8.         ax[i].set_title('Coeff: %0.1f' % coeff)
9.         [x.axis('off') for x in ax]
10.        plt.show()
11. move_and_show(latent_resource, latent_direction,
    np.arange(0.0, 3.0, 0.5))

```

Kode Sumber 4.2.10 Implementasi *latent direction*

4.2.4 Pembentukan Ulang Citra Wajah

Pada tahap ini, representasi *latent space* yang sudah disesuaikan akan dibentuk ulang menjadi citra wajah menggunakan model *generator* yang dimiliki GAN. Model tersebut akan mentransformasi *latent space* tersebut kembali menjadi *matrix array* dari *pixel* citra wajah yang kemudian dengan

bantuan *library PIL.Image* dapat menampilkannya menjadi sebuah gambar wajah dan menyimpannya, implementasinya dapat dilihat pada Kode Sumber 4.2.11.

```

1. def generate_image(latent_vector):
2.     latent_vector = latent_vector.reshape((1, 18,
3.         512))
4.     generator.set_dlatents(latent_vector)
5.     img_array = generator.generate_images()[0]
6.     img = PIL.Image.fromarray(img_array, 'RGB')
7.     return img.resize((256, 256))
8. generate_image(latent_result)

```

Kode Sumber 4.2.11 Implementasi pembentukan ulang citra wajah

4.3 Implementasi Proses Pengenalan Wajah

Pada subbab ini akan dijabarkan implementasi pada tahap pengenalan wajah menggunakan metode *Convolutional Neural Network* (CNN) untuk memproses citra input dan melakukan klasifikasi identitas wajah yang terdapat pada citra tersebut.

4.3.1 *Face Detection and Extraction*

Pada bagian ini akan dijabarkan implementasi *face detection* menggunakan *Dlib CNN face detector* untuk menemukan bagian wajah dari sebuah citra dan juga *face extraction* dengan cara mendapatkan titik-titik koordinat (*top*, *bottom*, *left*, *right*) yang kemudian digunakan untuk mengukur bagian citra yang akan dipotong. Implementasi tercantum pada Kode Sumber 4.3.1.

```

1. # Download Dlib CNN face detector
2. !                                                                    wget
   http://dlib.net/files/mmod\_human\_face\_detector.dat.
   bz2
3.
4. # Load CNN face detector into dlib
5. dnnFaceDetector=dlib.cnn_face_detection_model_v1("m
   mod_human_face_detector.dat")

```

```

6.
7. # For data train
8. for file_name in image_path_names:
9.     img=cv2.imread(file_name)
10.    gray=cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
11.    rects=dnnFaceDetector(gray,1)
12.    left,top,right,bottom=0,0,0,0
13.    for (i,rect) in enumerate(rects):
14.        left=rect.rect.left() #x1
15.        top=rect.rect.top() #y1
16.        right=rect.rect.right() #x2
17.        bottom=rect.rect.bottom() #y2
18.        width=right-left
19.        height=bottom-top
20.        img_crop=img[top:top+height,left:left+width]
21.
22.        img_path=path+'/ImagesTrainA_crop/'+file_name.split
23.        ('/')[ -1].split('_')[0]+'/' +file_name.split('/')[ -
24.        1]
25.        cv2.imwrite(img_path,img_crop)
26.
27. # For data ujit
28. for file_name in test_image_path_names:
29.     img=cv2.imread(file_name)
30.     gray=cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
31.     rects=dnnFaceDetector(gray,1)
32.     left,top,right,bottom=0,0,0,0
33.     for (i,rect) in enumerate(rects):
34.         left=rect.rect.left() #x1
35.         top=rect.rect.top() #y1
36.         right=rect.rect.right() #x2
37.         bottom=rect.rect.bottom() #y2
38.         width=right-left
39.         height=bottom-top
40.         img_crop=img[top:top+height,left:left+width]
41.
42.         img_path=path+'/ImagesTest_crop/'+file_name.split('
43.         /')[ -1].split('_')[0]+'/' +file_name.split('/')[ -1]
44.         cv2.imwrite(img_path,img_crop)

```

Kode Sumber 4.3.1 Implementasi *face detection and extraction*

4.3.2 Transfer Learning VGG Face Net dan Face Embedding

Pada tahap ini *transfer learning VGG Face Net* dilakukan dengan cara membuat arsitektur model dari *VGG Face Net* lalu memasukan *weight* dari *pre-trained vgg-face-model*. Kemudian model akan disesuaikan sehingga dapat digunakan untuk melakukan *face embedding* yaitu untuk mendapatkan ekstraksi fitur dari citra. Implementasi *transfer learning VGG Face Net* dapat dilihat pada Kode Sumber 4.3.2.

```

1. #Download pre-trained vgg-face-model-weights as .h5
   file
2. !                                                                    gdown
   https://drive.google.com/uc?id=1CPSeum3HpopfomUEK1g
   ybeuIVoeJT\_Eo
3.
4. #Define VGG_FACE_MODEL architecture
5. model = Sequential()
6. model.add(ZeroPadding2D((1,1),input_shape=(224,224,
   3)))
7. model.add(Convolution2D(64,          (3,          3),
   activation='relu'))
8. model.add(ZeroPadding2D((1,1)))
9. model.add(Convolution2D(64,          (3,          3),
   activation='relu'))
10. model.add(MaxPooling2D((2,2), strides=(2,2)))
11. model.add(ZeroPadding2D((1,1)))
12. model.add(Convolution2D(128,        (3,          3),
   activation='relu'))
13. model.add(ZeroPadding2D((1,1)))
14. model.add(Convolution2D(128,        (3,          3),
   activation='relu'))
15. model.add(MaxPooling2D((2,2), strides=(2,2)))
16. model.add(ZeroPadding2D((1,1)))
17. model.add(Convolution2D(256,        (3,          3),
   activation='relu'))
18. model.add(ZeroPadding2D((1,1)))
19. model.add(Convolution2D(256,        (3,          3),
   activation='relu'))
20. model.add(ZeroPadding2D((1,1)))

```

```

21. model.add(Convolution2D(256,          (3,          3),
    activation='relu'))
22. model.add(MaxPooling2D((2,2), strides=(2,2)))
23. model.add(ZeroPadding2D((1,1)))
24. model.add(Convolution2D(512,          (3,          3),
    activation='relu'))
25. model.add(ZeroPadding2D((1,1)))
26. model.add(Convolution2D(512,          (3,          3),
    activation='relu'))
27. model.add(ZeroPadding2D((1,1)))
28. model.add(Convolution2D(512,          (3,          3),
    activation='relu'))
29. model.add(MaxPooling2D((2,2), strides=(2,2)))
30. model.add(ZeroPadding2D((1,1)))
31. model.add(Convolution2D(512,          (3,          3),
    activation='relu'))
32. model.add(ZeroPadding2D((1,1)))
33. model.add(Convolution2D(512,          (3,          3),
    activation='relu'))
34. model.add(ZeroPadding2D((1,1)))
35. model.add(Convolution2D(512,          (3,          3),
    activation='relu'))
36. model.add(MaxPooling2D((2,2), strides=(2,2)))
37. model.add(Convolution2D(4096,          (7,          7),
    activation='relu'))
38. model.add(Dropout(0.5))
39. model.add(Convolution2D(4096,          (1,          1),
    activation='relu'))
40. model.add(Dropout(0.5))
41. model.add(Convolution2D(2622, (1, 1)))
42. model.add(Flatten())
43. model.add(Activation('softmax'))
44.
45. # Load VGG Face model weights
46. model.load_weights('vgg_face_weights.h5')
47.
48. # Remove Last Softmax layer and get model upto last
    flatten layer with outputs 2622 units
49. vgg_face=Model(inputs=model.layers[0].input,outputs
    =model.layers[-2].output)

```

Kode Sumber 4.3.2 Implementasi *transfer learning VGG Face Net*

Kemudian, implementasi *face embedding* dapat dilihat pada Kode Sumber 4.3.3.

```

1. #Prepare for embedding data train
2. x_train=[]
3. y_train=[]
4. person_folders=os.listdir(path+'/ImagesTrainA_crop/
   ')
5. person_rep=dict()
6. for i,person in enumerate(person_folders):
7.     person_rep[i]=person
8.
9.     image_names=os.listdir('ImagesTrainA_crop/'+person+
   '/')
10.    for image_name in image_names:
11.        img=load_img(path+'/ImagesTrainA_crop/'+person+'/'+
   image_name,target_size=(224,224))
12.        img=img_to_array(img)
13.        img=np.expand_dims(img,axis=0)
14.        img_encode=vgg_face(img)
15.
16.        x_train.append(np.squeeze(K.eval(img_encode)).tolist())
17.        y_train.append(i)
18. x_train=np.array(x_train)
19. y_train=np.array(y_train)
20.
21.
22. #Prepare for embedding data ujit
23. x_test=[]
24. y_test=[]
25. person_folders=os.listdir(path+'/ImagesTest_crop/')
26. for i,person in enumerate(person_folders):
27.
28.     image_names=os.listdir('ImagesTest_crop/'+person+'/'
   ')
29.     for image_name in image_names:

```

```

29.     img=load_img(path+'/ImagesTest_crop/'+person+'/'+image_name,target_size=(224,224))
30.     img=img_to_array(img)
31.     img=np.expand_dims(img,axis=0)
32.     img=preprocess_input(img)
33.     img_encode=vgg_face(img)
34.
    x_test.append(np.squeeze(K.eval(img_encode)).tolist())
35.     y_test.append(i)
36.
37. x_test=np.array(x_test)
38. y_test=np.array(y_test)

```

Kode Sumber 4.3.3 Implementasi *face embedding*

4.3.3 Pembuatan Model Pengenalan Wajah dan Fitting Model

Model yang akan dibuat untuk melakukan pengenalan wajah adalah *simple softmax classifier* yang akan melakukan klasifikasi dari hasil *face embeddings* terhadap label identitas wajah. Kemudian akan dilakukan *fitting model* terhadap data train dan data uji dengan jumlah *epoch*: 300, fungsi *activation*: *tanh* dan *softmax*, fungsi *loss*: *Sparse Categorical Crossentropy*, dan fungsi optimasi: *n-adam*. Implementasi pembuatan model pengenalan wajah dapat dilihat pada Kode Sumber 4.3.4.

```

1. # Softmax regressor to classify images based on encoding
2. classifier_model=Sequential()
3. classifier_model.add(Dense(units=100,input_dim=x_train.shape[1]))
4. classifier_model.add(Activation('tanh'))
5. classifier_model.add(Dense(units=40))
6. classifier_model.add(Activation('tanh'))
7. classifier_model.add(Dense(units=9))
8. classifier_model.add(Activation('softmax'))

```



```
9. classifier_model.compile(loss=tf.keras.losses.SparseCategoricalCrossentropy(),optimizer='nadam',metrics=['accuracy'])
```

Kode Sumber 4.3.4 Implementasi pembuatan model pengenalan wajah

Untuk implementasi *fitting model* dapat dilihat pada Kode Sumber 4.3.5.

```
1. H =
   classifier_model.fit(x_train,y_train,epochs=300,validation_data=(x_test,y_test),
   batch_size=len(x_train))
```

Kode Sumber 4.3.5 Implementasi *fitting model*

4.3.4 Face Classification

Setelah selesai membuat model pengenalan wajah dan melatihnya sesuai dengan dataset awal maka selanjutnya model siap digunakan untuk mencoba pengenalan wajah terhadap data uji baru. Implementasinya dapat dilihat pada Kode Sumber 4.3.6.

```
1. # Path to folder which contains new images to be
   tested and predicted
2. test_images_path=path+'/TestForPredict/'
3.
4. dnnFaceDetector=dlib.cnn_face_detection_model_v1("mod_human_face_detector.dat")
5.
6. def plot(img):
7.     plt.figure(figsize=(8,4))
8.     plt.imshow(img[:,:,:-1])
9.     plt.show()
10.
11. # Label names for class numbers
12. person_rep = {0: 'Handika',
13. 1: 'Rasyid',
14. 2: 'Aldinata',
15. 3: 'Fahmi',
16. 4: 'Steve',
```

```

17. 5: 'Ardo',
18. 6: 'Sugiarto',
19. 7: 'Yuda',
20. 8: 'Fasma'}
21.
22. for img_name in os.listdir('TestForPredict/'):
23.     if img_name=='crop_img.jpg':
24.         continue
25.     # Load Image
26.     img=cv2.imread(path+'/TestForPredict/'+img_name)
27.     gray=cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
28.     # Detect Faces
29.     rects=dnnFaceDetector(gray,1)
30.     left,top,right,bottom=0,0,0,0
31.     for (i,rect) in enumerate(rects):
32.         # Extract Each Face
33.         left=rect.rect.left() #x1
34.         top=rect.rect.top() #y1
35.         right=rect.rect.right() #x2
36.         bottom=rect.rect.bottom() #y2
37.         width=right-left
38.         height=bottom-top
39.         img_crop=img[top:top+height,left:left+width]
40.
41.         cv2.imwrite(path+'/TestForPredict/crop_img.jpg',img
42.             _crop)
43.
44.         # Get Embeddings
45.         crop_img=load_img(path+'/TestForPredict/crop_img.jp
46.             g',target_size=(224,224))
47.         crop_img=img_to_array(crop_img)
48.         crop_img=np.expand_dims(crop_img,axis=0)
49.         crop_img=preprocess_input(crop_img)
50.         img_encode=vgg_face(crop_img)
51.
52.         # Make Predictions
53.         embed=K.eval(img_encode)
54.         person=classifier_model.predict(embed)
55.         name=person_rep[np.argmax(person)]
56.         os.remove(path+'/TestForPredict/crop_img.jpg')

```

```

54.     cv2.rectangle(img,(left,top),(right,bottom),(0,255,
        0), 2)
55.     img=cv2.putText(img,name,(left,top-
        10),cv2.FONT_HERSHEY_SIMPLEX,1,(255,0,255),2,cv2.LI
        NE_AA)
56.     img=cv2.putText(img,str(np.max(person))),(right,bott
        om+10),cv2.FONT_HERSHEY_SIMPLEX,0.5,(0,0,255),1,cv2
        .LINE_AA)
57.
58.     # Save images with bounding box,name and accuracy
59.     cv2.imwrite(path+'/PredictionsResults/'+img_name,img)
60.     plot(img)

```

Kode Sumber 4.3.6 Implementasi *face classification*

(Halaman ini sengaja dikosongkan)

BAB V

UJI COBA DAN EVALUASI

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

5.1 Lingkungan Uji Coba

Lingkungan uji coba pada tugas akhir ini adalah *Google Collaboratory*. Sistem operasi yang digunakan adalah *Ubuntu 18.04.3 LTS*. *Cloud* yang digunakan memiliki spesifikasi *Intel® Xeon®* 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. Pada sisi perangkat lunak, uji coba pada tugas akhir ini dilakukan dengan menggunakan bahasa pemrograman *Python 3.6*, dilengkapi dengan *library* antara lain *OpenCV*, *Tensorflow-GPU*, *Keras-GPU*, *PyTorch-GPU*, *Numpy*, *Pickle*, *Pandas*, *Matplotlib*, *DLib*, *DNNLib*, *Scikit-learn*, *PIL-image*, dan *SciPy*.

5.2 Deskripsi Dataset

Dataset yang digunakan untuk proses pembuatan sintesa citra wajah menggunakan *Generative Adversarial Network* dibagi menjadi dua. Pertama adalah dataset referensi (diambil dari dataset *CelebA-HQ* dan *FFHQ*) yang telah digunakan untuk proses *learning* sebelumnya pada *pre-trained* model GAN. Selanjutnya yang kedua adalah sebagian dari dataset sumber dengan kondisi wajah pose lurus ke depan yang akan digunakan sebagai data uji pada implementasi *pre-trained* model GAN yang telah dipilih.

Dataset yang digunakan untuk uji coba pengenalan wajah adalah dari sebagian dataset sumber selain yang telah digunakan sebagai data uji pada implementasi *pre-trained* model GAN, dengan 6 variasi kondisi untuk setiap identitas wajah (ekspresi netral, ekspresi senyum, ekspresi sedih, pose kanan, pose kiri, dan memakai atribut kacamata).

Dataset *CelebFaces-Attributes* (CelebA) adalah dataset atribut wajah skala besar dari beberapa selebritas di dunia secara khusus yang disediakan oleh *Multimedia Laboratory* (MMLAB), *The Chinese University of Hong Kong*, yaitu berisi 202.599 citra wajah, 10.177 identitas, dan 5 lokasi *landmark* serta 40 anotasi atribut biner per gambarnya [26]. Kemudian untuk keperluan proses *learning* pada *pre-trained* model GAN maka dibuatlah versi kualitas tinggi dari dataset CelebA tersebut oleh *NVIDIA Corporation* dan disebut dengan dataset CelebA-HQ, yaitu terdiri dari 30.000 citra PNG dalam resolusi 1024×1024 .

Dataset *Flickr-Faces-HQ* (FFHQ) adalah dataset citra wajah berkualitas tinggi dari beberapa orang di dunia secara umum yang disediakan oleh *NVIDIA Corporation*. Dataset ini didapatkan dari hasil menyaring atau *crawled* pada website *Flickr* serta diproses terlebih dahulu seperti disejajarkan dan dipotong secara otomatis oleh *NVIDIA Corporation*. Dataset FFHQ terdiri dari 70.000 citra PNG dalam resolusi 1024×1024 dan berisi variasi yang cukup besar dalam hal usia, etnis, dan latar belakang gambar. Dataset ini juga memiliki jangkauan yang baik dari aksesoris seperti kacamata, kacamata hitam, topi, dan lain-lain [27].

Dataset sumber merupakan citra wajah yang dikumpulkan sendiri oleh pengusul (dari beberapa teman pengusul sebagai sampel). Data sumber ini terdiri dari 9 identitas wajah (termasuk gambar wajah pengusul dan teman-temannya) dengan resolusi dan tipe file yang berbeda (JPG, JPEG, PNG). Sehingga sebelum digunakan pada implementasi model GAN maupun pada pengenalan wajah maka pada data sumber tersebut perlu dilakukan praproses citra wajah untuk dipersiapkan dan disesuaikan dengan standar masukan dari model serta agar menunjang keberhasilan dari model.

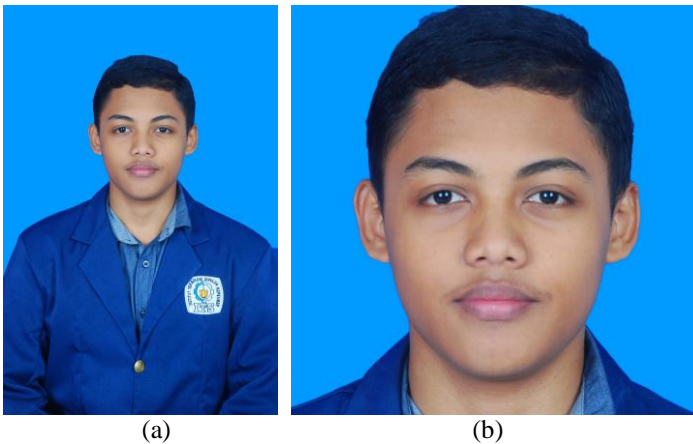
5.3 Hasil Praproses

Sebelum memasuki proses augmentasi data dan pengenalan wajah yang telah dirancang, akan dilakukan praproses terhadap

citra input. Terdapat perbedaan antara praproses citra yang dilakukan pada setiap tahapan. Perbedaan ini terletak pada tujuan praproses citra itu sendiri. Pada proses augmentasi data, tujuan praproses citra adalah untuk mendeteksi wajah, mensejajarkan, dan merubah ukuran dari citra input agar sesuai dengan standar masukan dari model *Generative Adversarial Networks* (GAN). Pada proses pengenalan wajah, praproses citra bertujuan untuk mendeteksi dan mengambil citra bagian wajah saja dari citra input agar dapat digunakan di tahap selanjutnya yaitu *face embedding*.

5.3.1 Praproses Citra Pada Augmentasi Data

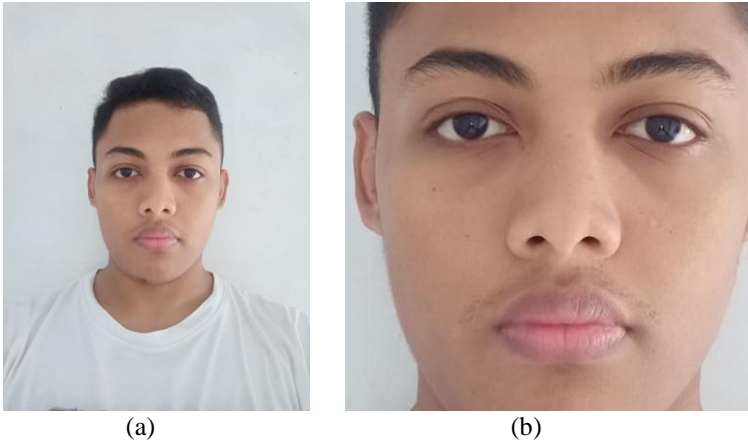
Tahap praproses pada augmentasi data dimulai dengan citra input akan dilakukan *face detection* untuk mendeteksi wajah yang terdapat pada citra input. Setelah dideteksinya wajah, citra kemudian dilakukan *face landmark* untuk mendapatkan titik-titik koordinat pada wajah yang terdeteksi. Kemudian, pada citra dilakukan *face alignment* dan *resize* untuk menyejajarkan ulang wajah berdasarkan titik *landmark* lalu memotong bagian tersebut dan menyimpannya dengan ukuran 1024×1024 , menyesuaikan dengan standar masukan dari model GAN. Contoh citra hasil praproses pada augmentasi data dapat dilihat pada Gambar 5.1.



Gambar 5.1 (a) Citra input; (b) Hasil praproses pada augmentasi data

5.3.2 Praproses Citra Pada Pengenalan Wajah

Tahap praproses pada pengenalan wajah dimulai dengan melakukan *face detection* untuk mendeteksi wajah yang terdapat pada citra input. Setelah dideteksinya wajah, citra kemudian dilakukan *face extraction* untuk mendapatkan potongan citra yang hanya berisi wajah yang terdeteksi. Contoh citra hasil praproses pada pengenalan wajah dapat dilihat pada Gambar 5.2.



Gambar 5.2 (a) Citra input; (b) Hasil praproses pada pengenalan wajah

5.4 Skenario Uji Coba

Proses uji coba berguna untuk menemukan parameter-parameter yang menghasilkan performa model yang paling optimal. Parameter yang tepat akan memberikan hasil yang lebih baik pada saat proses uji coba.

Hasil terbaik dari suatu skenario uji coba akan digunakan untuk skenario uji coba berikutnya. Pada subbab ini, skenario uji coba dibagi menjadi 2 serta ditambah dengan 2 hasil uji coba yaitu:

1. Skenario uji coba pada tahap implementasi *Generative Adversarial Network* (GAN)

2. Skenario uji coba pada tahap penyesuaian representasi *latent space*
3. Hasil uji coba pada proses augmentasi data
4. Hasil uji coba pada proses pengenalan wajah

5.4.1 Skenario Uji Coba pada tahap implementasi Generative Adversarial Network (GAN)

Pada GAN, skenario uji coba akan dicoba pada arsitektur GAN yang telah dirancang. Skenario uji coba yang akan dilakukan dengan mencoba 2 jenis *pretrained* StyleGAN, serta dengan mencoba merubah parameter GAN.

Tabel 5.1 Spesifikasi parameter awal arsitektur GAN

Keterangan	Parameter
Ukuran <i>batch</i>	1
<i>Learning Rate</i>	1
<i>Random Noise</i>	<i>False</i>
Jumlah iterasi	1000

Setiap skenario nantinya akan diuji pada sebuah citra input untuk dibandingkan hasil performanya berdasarkan nilai *loss*. Tabel 5.1 berisi parameter-parameter awal arsitektur GAN yang digunakan dan dapat berubah di setiap uji coba yang dilakukan. Pada setiap skenario uji coba akan ditetapkan nilai parameter yang dapat meningkatkan kinerja arsitektur.

Uji coba jenis *pre-trained* StyleGAN adalah berdasarkan data latih yang digunakan untuk proses *learning*-nya, yaitu StyleGAN-CelebAHQ dan StyleGAN-FFHQ. Percobaan dilakukan terhadap sebuah citra wajah tanpa merubah parameter lain pada GAN (seperti parameter awal).

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

1. Ukuran *Batch*

Uji coba pertama adalah uji coba ukuran *batch* pada model yang akan dibuat. Ukuran *batch* yang akan digunakan pada uji coba ini antara lain 1 dan 2. Percobaan dilakukan dengan menggunakan jenis *pre-trained* GAN yang paling optimal dan tanpa merubah parameter lain (masih sama seperti parameter awal).

2. *Learning rate*

Uji coba *learning rate* divariasikan dengan harapan mendapatkan performa model yang lebih baik dari sebelumnya. Variasi *learning rate* yang akan dicoba yaitu 0,01; 0,1; 0,5; dan 1.

3. *Random Noise*

Uji coba adanya atau tidak adanya penambahan *random noise* pada saat optimasi model akan dilakukan setelah ditemukan 2 nilai parameter lain yang menghasilkan performa baik. Parameter yang akan digunakan pada uji coba ini antara lain *False* dan *True*.

4. Jumlah Iterasi

Uji coba yang terakhir adalah dengan mencoba merubah jumlah iterasi untuk mengoptimasi model GAN. Parameter yang akan digunakan pada uji coba ini antara lain 500, 1000, 2000, 4000, dan 8000 iterasi.

Uji coba penggantian jenis *pretrained* StyleGAN menghasilkan perbandingan nilai *loss* yang dapat dilihat pada Tabel 5.2.

Tabel 5.2 Hasil uji coba penggantian jenis *pretrained* StyleGAN

Jenis <i>Pretrained</i>	<i>Loss</i>
StyleGAN-CelebAHQ	0,28
StyleGAN-FFHQ	0,25

Pada pengujian jenis *pretrained* StyleGAN, didapatkan StyleGAN-FFHQ lebih optimal bila dibandingkan dengan StyleGAN-CelebAHQ. Selain berdasarkan nilai *loss* tersebut

kedua jenis *pretrained* tersebut juga dievaluasi pada kualitas hasil proses pembentukan ulang citra wajah secara visual. Maka selanjutnya, pada uji coba penggantian parameter, jenis *pretrained* yang digunakan adalah StyleGAN-FFHQ.

Uji coba penggantian ukuran *batch* pada arsitektur GAN menghasilkan perbandingan nilai *loss* yang dapat dilihat pada Tabel 5.3.

Tabel 5.3 Hasil uji coba penggantian ukuran *batch*

Ukuran <i>Batch</i>	<i>Loss</i>
1	0,25
2	0,32

Pada pengujian ukuran *batch*, didapatkan ukuran *batch* 1 merupakan ukuran *batch* yang lebih optimal.

Uji coba penggantian *learning rate* pada arsitektur GAN menghasilkan perbandingan nilai *loss* yang dapat dilihat pada Tabel 5.4.

Tabel 5.4 Hasil uji coba penggantian *learning rate*

<i>Learning Rate</i>	<i>Loss</i>
1	0,25
0,5	0,28
0,1	0,39
0,01	0,67

Pada hasil uji coba, didapatkan bahwa nilai *learning rate* 1 menghasilkan *loss* terendah dibandingkan percobaan yang lainnya.

Uji coba penggantian *random noise* pada model GAN menghasilkan perbandingan nilai *loss* yang dapat dilihat pada Tabel 5.5.

Tabel 5.5 Hasil uji coba penggantian *random noise*

<i>Random Noise</i>	<i>Loss</i>
<i>False</i>	0,25
<i>True</i>	0,26

Pada pengujian penggantian *random noise*, didapatkan bahwa adanya penambahan *random noise* pada saat optimasi model justru menambah nilai *loss* dibandingkan dengan percobaan dengan tidak adanya penambahan *random noise*.

Uji coba penggantian jumlah iterasi pada model GAN menghasilkan perbandingan nilai *loss* yang dapat dilihat pada Tabel 5.6.

Tabel 5.6 Hasil uji coba penggantian jumlah iterasi

Jumlah Iterasi	<i>Loss</i>
500	0,42
1000	0,25
2000	0,21
4000	0,15
8000	0,15

Pada pengujian penggantian jumlah iterasi, didapatkan bahwa semakin bertambahnya iterasi maka model bisa mendapatkan nilai *loss* yang lebih rendah namun setelah lebih dari 4000 iterasi penurunan nilai *loss* sudah tidak terjadi lagi atau sangat kecil. Sehingga dapat disimpulkan bahwa jumlah iterasi yang optimal adalah 4000 iterasi.

Setelah ditemukan parameter optimal pada model GAN berdasarkan hasil nilai *loss* maka juga akan dilakukan evaluasi berdasarkan hasil pembentukan ulang citra dari representasi *latent space* oleh model GAN, yang dapat dilihat pada Gambar 5.3 dan Gambar 5.4. Dengan berturut-turut citra wajah asli pada sebelah kiri dan hasil sintesa citra wajah pada sebelah kanan.



Gambar 5.3 Contoh pembentukan ulang citra dengan parameter awal



Gambar 5.4 Contoh pembentukan ulang citra dengan parameter optimal

5.4.2 Skenario Uji Coba Pada Tahap Penyesuaian Representasi Latent Space

Pada tahap penyesuaian representasi *latent space* ada beberapa metode yang akan dicoba pada tugas akhir ini yaitu *common average style mixing*, *vector operation in latent space*, *NVIDIA's style mixing technique*, dan *latent direction*. Uji coba yang dilakukan adalah untuk target sintesa wajah dengan ekspresi senyum, sedih, pose kanan, pose kiri, dan pemakaian atribut kacamata. Tapi yang akan ditampilkan sebagai contoh dalam buku ini adalah untuk target ekspresi tersenyum. Evaluasi yang dilakukan terhadap setiap metode adalah melalui kualitas visual pada hasil pembentukan ulang dari citra wajah.

5.4.2.1 Uji Coba Metode *Common Average Style Mixing*

Pada Gambar 5.5 dapat dilihat hasil uji coba pembentukan ulang citra wajah dari hasil penyesuaian dengan metode *common average style mixing*. Dua gambar di bagian atas secara berturut-turut adalah untuk latent awal dan latent target. Kemudian tiga gambar di bagian bawah secara berturut-turut adalah hasil dari metode *common average style mixing* dengan variasi koefisien α yang digunakan yaitu 0,25; 0,50; dan 0,75.



Gambar 5.5 Hasil uji coba metode *common average style mixing*

Berdasarkan hasil uji coba metode *common average style mixing*, ditemukan bahwa jika semakin kecil nilai koefisien α maka identitas wajah dari citra awal akan semakin berkurang dan lebih cenderung kepada identitas wajah dari citra target. Sehingga apabila metode ini digunakan untuk augmentasi data pada aplikasi pengenalan wajah ada kecenderungan untuk merubah identitas dari wajah awal dan mengurangi performa dari pengenalan wajah.

5.4.2.2 Uji Coba Metode *Vector Operation In Latent Space*

Pada Gambar 5.6 dapat dilihat hasil uji coba pembentukan ulang citra wajah dari hasil penyesuaian dengan metode *vector*

operation in latent space. Dua gambar di bagian atas secara berturut-turut adalah untuk latent awal dan latent target. Kemudian tiga gambar di bagian bawah secara berturut-turut adalah hasil dari metode *vector operation in latent space* dengan koefisien pangkat yang digunakan yaitu 0,25; 0,50; dan 0,75.



Gambar 5.6 Hasil uji coba metode *vector operation in latent space*

Berdasarkan hasil uji coba metode *vector operation in latent space*, ditemukan bahwa jika semakin besar nilai koefisien perkalian maka identitas wajah dari citra awal akan semakin berkurang dan lebih cenderung kepada identitas wajah dari citra target. Namun metode ini masih lebih bagus jika dibandingkan dengan metode *common average style mixing* karena masih bisa mempertahankan sebagian dari identitas wajah awal. Namun metode ini belum bisa digunakan untuk augmentasi data pada aplikasi pengenalan wajah karena masih belum begitu baik dalam mempertahankan identitas wajah awal.

5.4.2.3 Uji Coba Metode *NVIDIA's Style Mixing Technique*

Pada Gambar 5.7 dapat dilihat hasil uji coba pembentukan ulang citra wajah dari hasil penyesuaian dengan metode *NVIDIA's*

style mixing technique. Dua gambar di bagian atas dan kiri secara berturut-turut adalah untuk latent awal dan latent target. Kemudian gambar yang ada di bagian kanan bawah adalah hasil dari metode *NVIDIA's style mixing technique* dengan *style range*: 1-11.



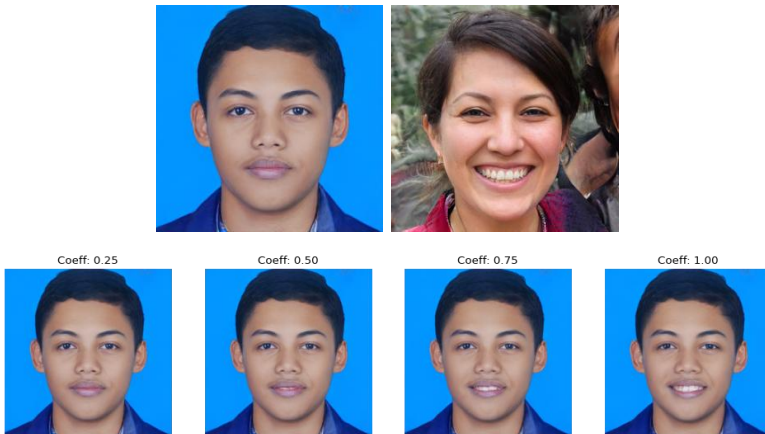
Gambar 5.7 Hasil uji coba metode *NVIDIA's style mixing technique*

Berdasarkan hasil uji coba metode *NVIDIA's style mixing technique*, ditemukan bahwa *style range* menyimpan nilai penting untuk menentukan bagian mana saja dari citra awal yang tetap dipertahankan dan bagian mana saja dari citra target yang akan digabung. Namun metode ini tidak menjelaskan lebih detail mengenai *style range* yang digunakan pada operasi vektor *latent space*. Sehingga akan susah jika hendak menentukan style khusus yang ingin diambil dari citra target untuk digabungkan pada citra awal. Jadi metode ini dirasa masih belum bisa dipilih untuk digunakan untuk augmentasi data pada aplikasi pengenalan wajah.

5.4.2.4 Uji Coba Metode *Latent Direction*

Pada Gambar 5.8. dapat dilihat hasil uji coba pembentukan ulang citra wajah dari hasil penyesuaian dengan metode *latent direction*. Dua gambar di bagian atas secara berturut-turut adalah untuk latent awal dan latent target. Kemudian 5 gambar di bagian

bawah secara berturut-turut adalah hasil dari metode *latent direction* dengan koefisien perkalian yang digunakan yaitu 0,25; 0,50; 0,75; dan 1,00.



Gambar 5.8 Hasil uji coba metode *latent direction*

Berdasarkan hasil uji coba metode *latent direction*, ditemukan bahwa jika pergeseran *style* sampai pada koefisien 1,00 tidak merubah identitas wajah dari citra. Dengan melihat hasil yang sesuai seperti yang diharapkan maka metode ini akan digunakan untuk augmentasi data pada aplikasi pengenalan wajah karena cukup baik dalam mempertahankan identitas wajah awal.

5.4.3 Hasil Uji Coba Pada Proses Augmentasi Data

Uji coba augmentasi data dilakukan terhadap 9 citra wajah dengan identitas wajah berbeda yang sudah dikumpulkan penulis pada dataset sumber. Pada setiap citra wajah akan diberikan 5 jenis kondisi untuk ditambahkan, yaitu ekspresi senyum, ekspresi sedih, pose kanan, pose kiri, dan atribut kacamata.

Metode yang digunakan untuk melakukan penyesuaian latent space adalah menggunakan *latent direction* dan uji coba yang dilakukan adalah dengan merubah nilai koefisien perkalian yang digunakan pada *latent direction*.

5.4.3.1 Ekspresi Senyum

Contoh hasil dari proses augmentasi data untuk ekspresi senyum dapat dilihat pada Gambar 5.9. Koefisien perkalian pada *latent direction* yang digunakan adalah 0,50; 0,75; dan 1,00. Pemilihan nilai koefisien ini adalah dengan mempertimbangkan perubahan identitas dari wajah karena jika terlalu besar dapat merubah identitas dari wajah.



Gambar 5.9 Contoh hasil proses augmentasi data untuk ekspresi senyum

5.4.3.2 Ekspresi Sedih

Contoh hasil dari proses augmentasi data untuk ekspresi sedih dapat dilihat pada Gambar 5.10. Koefisien perkalian pada *latent direction* yang digunakan adalah 1,00; 1,25; dan 1,50. Pemilihan nilai koefisien ini adalah dengan mempertimbangkan perubahan identitas dari wajah karena jika terlalu besar dapat merubah identitas dari wajah.



Gambar 5.10 Contoh hasil proses augmentasi data untuk ekspresi sedih

5.4.3.3 Pose Kanan

Contoh hasil dari proses augmentasi data untuk pose kanan dapat dilihat pada Gambar 5.11. Koefisien perkalian pada *latent direction* yang digunakan adalah 1,50; 2,00; dan 2,50. Pemilihan nilai koefisien ini adalah dengan mempertimbangkan perubahan identitas dari wajah karena jika terlalu besar dapat merubah identitas dari wajah selain itu koefisien yang terlalu besar juga dapat merusak hasil pembentukan ulang citra wajah.



Gambar 5.11 Contoh hasil proses augmentasi data untuk pose kanan

5.4.3.4 Pose Kiri

Contoh hasil dari proses augmentasi data untuk pose kiri dapat dilihat pada Gambar 5.12. Koefisien perkalian pada *latent direction* yang digunakan adalah 2,00; 3,00; dan 4,00. Pemilihan nilai koefisien ini adalah dengan mempertimbangkan perubahan identitas dari wajah karena jika terlalu besar dapat merubah identitas dari wajah selain itu koefisien terlalu kecil tidak akan memberikan efek yang terlihat untuk pose kiri.



Gambar 5.12 Contoh hasil proses augmentasi data untuk pose kiri

5.4.3.5 Atribut Kacamata

Contoh hasil dari proses augmentasi data untuk atribut kacamata dapat dilihat pada Gambar 5.13. Koefisien perkalian pada *latent direction* yang digunakan adalah antara 2,3 hingga 2,8. Pemilihan nilai koefisien ini adalah dengan mempertimbangkan perubahan identitas dari wajah karena jika terlalu besar dapat merubah identitas dari wajah bahkan merusak hasil pembentukan ulang citra wajah, sedangkan jika terlalu kecil maka efek atribut kacamata tidak akan muncul.



Gambar 5.13 Contoh hasil proses augmentasi data untuk atribut kacamata

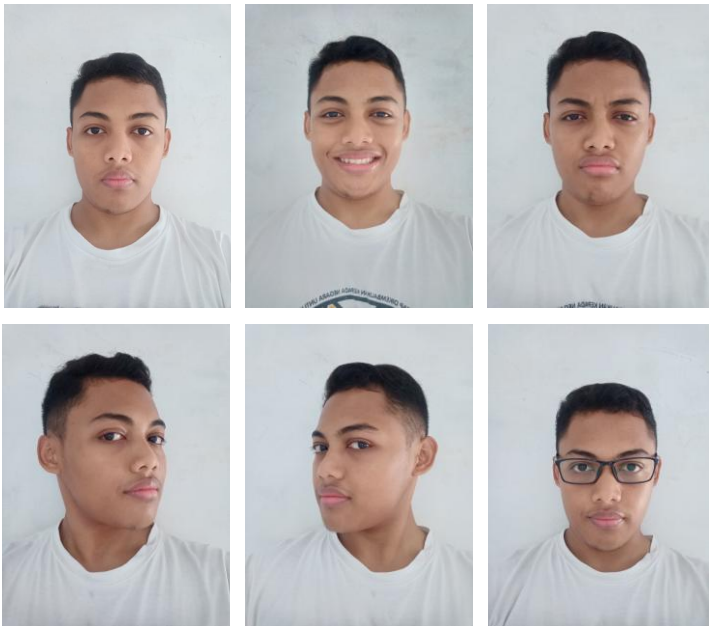
5.4.4 Hasil Uji Coba Pada Proses Pengenalan Wajah

Uji coba pada proses pengenalan wajah akan dibagi menjadi 7 kondisi skenario pada data latih terhadap model *classifier* serta akan dilihat dan dibandingkan performa model pada setiap kondisi skenario yang dicoba.

- Skenario A: menggunakan data latih awal tanpa penambahan hasil augmentasi data
- Skenario B1: data latih awal ditambah hasil augmentasi data untuk ekspresi senyum
- Skenario B2: data latih awal ditambah hasil augmentasi data untuk ekspresi sedih
- Skenario B3: data latih awal ditambah hasil augmentasi data untuk pose kanan dan pose kiri

- Skenario B4: data latih awal ditambah hasil augmentasi data untuk atribut kacamata
- Skenario B5: data latih awal ditambah hasil augmentasi data untuk ekspresi senyum, sedih, pose kanan, dan kiri
- Skenario B6: data latih awal ditambah hasil augmentasi data untuk seluruh hasil augmentasi data (ekspresi senyum, ekspresi sedih, pose kanan, pose kiri, dan atribut kacamata)

Data uji yang digunakan untuk evaluasi performa model *classifier* adalah 54 citra wajah asli yang baru dengan 9 identitas wajah dan setiap identitas dengan 6 kondisi citra wajah (ekspresi netral, ekspresi senyum, ekspresi sedih, pose kanan, pose kiri, dan atribut kacamata). Contoh citra wajah yang digunakan sebagai data uji dapat dilihat pada Gambar 5.14 dan spesifikasi dari data uji untuk uji coba pengenalan wajah dapat dilihat pada Tabel 5.7.



Gambar 5.14 Contoh data uji pada model *classifier* pengenalan wajah

Tabel 5.7 Spesifikasi data uji untuk uji coba pengenalan wajah

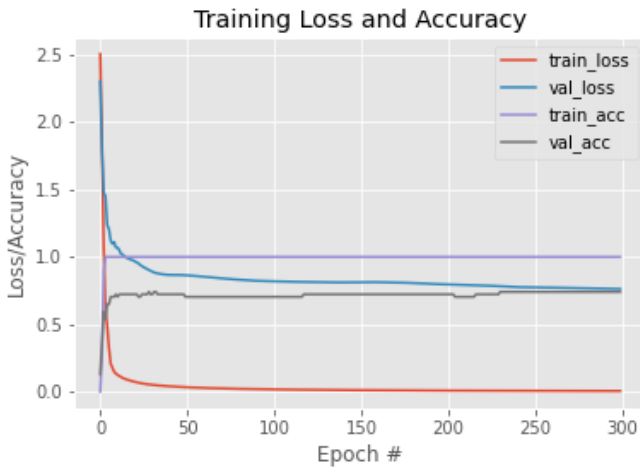
Identitas Wajah	Data uji	Keterangan
Aldinata	6 citra asli	6 kondisi
Ardo	6 citra asli	6 kondisi
Fahmi	6 citra asli	6 kondisi
Fasma	6 citra asli	6 kondisi
Handika	6 citra asli	6 kondisi
Rasyid	6 citra asli	6 kondisi
Steve	6 citra asli	6 kondisi
Sugiarto	6 citra asli	6 kondisi
Yuda	6 citra asli	6 kondisi

5.4.4.1 Uji Coba Skenario A (tanpa penambahan data latih)

Uji coba yang dilakukan pada skenario A menggunakan data latih awal yaitu 18 citra wajah dari data sumber dengan 9 identitas wajah, 2 citra wajah asli untuk setiap identitas, dan 1 kondisi (netral). Spesifikasi data latih pada skenario A dapat dilihat pada Tabel 5.8. Gambar plot hasil fitting model pengenalan wajah skenario A dapat dilihat pada Gambar 5.15.

Tabel 5.8 Spesifikasi data latih skenario A

Identitas Wajah	Data Latih	Keterangan
Aldinata	2 citra asli	1 kondisi
Ardo	2 citra asli	1 kondisi
Fahmi	2 citra asli	1 kondisi
Fasma	2 citra asli	1 kondisi
Handika	2 citra asli	1 kondisi
Rasyid	2 citra asli	1 kondisi
Steve	2 citra asli	1 kondisi
Sugiarto	2 citra asli	1 kondisi
Yuda	2 citra asli	1 kondisi



Gambar 5.15 Plot hasil fitting model pengenalan wajah skenario A

Evaluasi performa model *classifier* menggunakan nilai *Precision*, *Recall*, *F1-Score*, dan *Accuracy*. Hasil evaluasi pada skenario A dapat dilihat pada Tabel 5.9.

Tabel 5.9 Hasil evaluasi pada skenario A

	Precision	Recall	F1-Score	Support
Aldinata	1,00	0,67	0,80	6
Ardo	1,00	0,67	0,80	6
Fahmi	0,50	0,50	0,50	6
Fasma	0,60	1,00	0,75	6
Handika	1,00	0,67	0,80	6
Rasyid	1,00	1,00	1,00	6
Steve	0,50	1,00	0,67	6
Sugiarto	0,50	0,17	0,25	6
Yuda	1,00	1,00	1,00	6
accuracy			0,74	54
macro avg	0,79	0,74	0,73	54
weighted avg	0,79	0,74	0,73	54

Dari hasil evaluasi pada skenario A maka didapatkan nilai rata-rata *Precision* 0,79; *Recall* 0,74; *F1-Score* 0,73; dan *Accuracy* 0,74. Contoh hasil prediksi pada salah satu identitas wajah skenario A dapat dilihat pada Tabel 5.10. Skor yang dimaksud adalah nilai output prediksi tertinggi dari model *classifier* pengenalan wajah.

Tabel 5.10 Hasil prediksi pada salah satu identitas wajah Skenario A

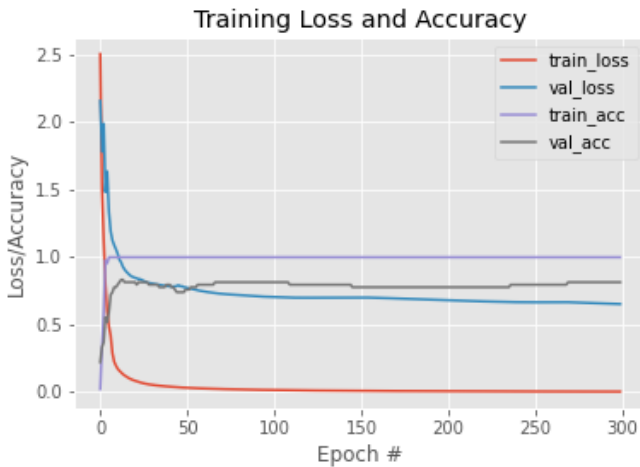
Nama File Citra Wajah	Hasil Prediksi	Skor
Aldinata_Kacamata.jpeg	Fasma	0,6058979
Aldinata_Netral.jpeg	Aldinata	0,9568714
Aldinata_PoseKanan.jpeg	Aldinata	0,8509558
Aldinata_PoseKiri.jpeg	Sugiarto	0,68165606
Aldinata_Sedih.jpeg	Aldinata	0,6553414
Aldinata_Senyum.jpeg	Aldinata	0,495758

5.4.4.2 Uji Coba Skenario B1 (penambahan data latihan ekspresi senyum)

Uji coba yang dilakukan pada skenario B1 menggunakan data latihan yaitu 18 citra wajah asli dengan 9 identitas wajah dan 1 kondisi (netral) ditambah 27 citra wajah dari hasil augmentasi data GAN dengan 9 identitas wajah dan 1 kondisi (ekspresi senyum). Spesifikasi data latihan pada skenario B1 dapat dilihat pada Tabel 5.11. Gambar plot hasil fitting model pengenalan wajah skenario B1 dapat dilihat pada Gambar 5.16.

Tabel 5.11 Spesifikasi data latihan skenario B1

Identitas Wajah	Data Latihan	Keterangan
Aldinata	2 citra asli, 3 citra GAN	2 kondisi
Ardo	2 citra asli, 3 citra GAN	2 kondisi
Fahmi	2 citra asli, 3 citra GAN	2 kondisi
Fasma	2 citra asli, 3 citra GAN	2 kondisi
Handika	2 citra asli, 3 citra GAN	2 kondisi
Rasyid	2 citra asli, 3 citra GAN	2 kondisi
Steve	2 citra asli, 3 citra GAN	2 kondisi
Sugiarto	2 citra asli, 3 citra GAN	2 kondisi
Yuda	2 citra asli, 3 citra GAN	2 kondisi



Gambar 5.16 Plot hasil fitting model pengenalan wajah skenario B1

Evaluasi performa model *classifier* menggunakan nilai *Precision*, *Recall*, *F1-Score*, dan *Accuracy*. Hasil evaluasi pada skenario B1 dapat dilihat pada Tabel 5.12.

Tabel 5.12 Hasil evaluasi pada skenario B1

	Precision	Recall	F1-Score	Support
Aldinata	1,00	0,83	0,91	6
Ardo	1,00	0,67	0,80	6
Fahmi	0,42	0,83	0,56	6
Fasma	1,00	0,83	0,91	6
Handika	1,00	0,67	0,80	6
Rasyid	1,00	0,83	0,91	6
Steve	0,75	1,00	0,86	6
Sugiarto	0,80	0,67	0,73	6
Yuda	1,00	1,00	1,00	6
accuracy			0,81	54
macro avg	0,89	0,81	0,83	54
weighted avg	0,89	0,81	0,83	54

Dari hasil evaluasi pada skenario B1 maka didapatkan nilai rata-rata *Precision* 0,89; *Recall* 0,81; *F1-Score* 0,83; dan *Accuracy* 0,81. Contoh hasil prediksi pada salah satu identitas wajah skenario B1 dapat dilihat pada Tabel 5.13. Skor yang dimaksud adalah nilai output prediksi tertinggi dari model *classifier* pengenalan wajah.

Tabel 5.13 Hasil prediksi pada salah satu identitas wajah Skenario B1

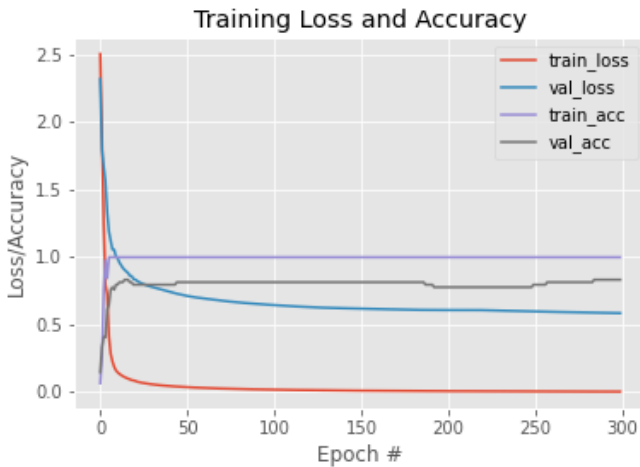
Nama File Citra Wajah	Hasil Prediksi	Skor
Aldinata_Kacamata.jpeg	Fahmi	0,50613165
Aldinata_Netral.jpeg	Aldinata	0,76727897
Aldinata_PoseKanan.jpeg	Aldinata	0,69936275
Aldinata_PoseKiri.jpeg	Aldinata	0,25011402
Aldinata_Sedih.jpeg	Aldinata	0,29790172
Aldinata_Senyum.jpeg	Aldinata	0,74366385

5.4.4.3 Uji Coba Skenario B2 (penambahan data latihan ekspresi sedih)

Uji coba yang dilakukan pada skenario B2 menggunakan data latihan yaitu 18 citra wajah asli dengan 9 identitas wajah dan 1 kondisi (netral) ditambah 27 citra wajah dari hasil augmentasi data GAN dengan 9 identitas wajah dan 1 kondisi (ekspresi sedih). Spesifikasi data latihan pada skenario B2 dapat dilihat pada Tabel 5.14. Gambar plot hasil fitting model pengenalan wajah skenario B2 dapat dilihat pada Gambar 5.17.

Tabel 5.14 Spesifikasi data latihan skenario B2

Identitas Wajah	Data Latihan	Keterangan
Aldinata	2 citra asli, 3 citra GAN	2 kondisi
Ardo	2 citra asli, 3 citra GAN	2 kondisi
Fahmi	2 citra asli, 3 citra GAN	2 kondisi
Fasma	2 citra asli, 3 citra GAN	2 kondisi
Handika	2 citra asli, 3 citra GAN	2 kondisi
Rasyid	2 citra asli, 3 citra GAN	2 kondisi
Steve	2 citra asli, 3 citra GAN	2 kondisi
Sugiarto	2 citra asli, 3 citra GAN	2 kondisi
Yuda	2 citra asli, 3 citra GAN	2 kondisi



Gambar 5.17 Plot hasil fitting model pengenalan wajah skenario B2

Evaluasi performa model *classifier* menggunakan nilai *Precision*, *Recall*, *F1-Score*, dan *Accuracy*. Hasil evaluasi pada skenario B2 dapat dilihat pada Tabel 5.15.

Tabel 5.15 Hasil evaluasi pada skenario B2

	Precision	Recall	F1-Score	Support
Aldinata	1,00	0,83	0,91	6
Ardo	1,00	1,00	1,00	6
Fahmi	0,60	1,00	0,75	6
Fasma	1,00	0,83	0,91	6
Handika	1,00	0,83	0,91	6
Rasyid	0,75	1,00	0,86	6
Steve	0,62	0,83	0,71	6
Sugiarto	1,00	0,67	0,80	6
Yuda	1,00	0,50	0,67	6
accuracy			0,83	54
macro avg	0,89	0,83	0,84	54
weighted avg	0,89	0,83	0,84	54

Dari hasil evaluasi pada skenario B2 maka didapatkan nilai rata-rata *Precision* 0,89; *Recall* 0,83; *F1-Score* 0,84; dan *Accuracy* 0,83. Contoh hasil prediksi pada salah satu identitas wajah skenario B2 dapat dilihat pada Tabel 5.16. Skor yang dimaksud adalah nilai output prediksi tertinggi dari model *classifier* pengenalan wajah.

Tabel 5.16 Hasil prediksi pada salah satu identitas wajah Skenario B2

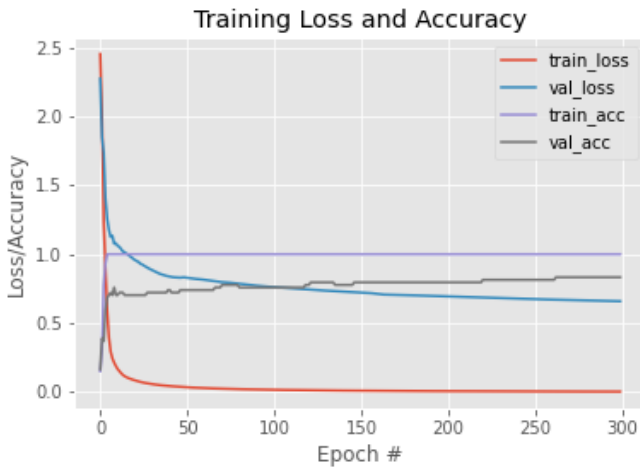
Nama File Citra Wajah	Hasil Prediksi	Skor
Aldinata_Kacamata.jpeg	Fahmi	0,31963733
Aldinata_Netral.jpeg	Aldinata	0,8668376
Aldinata_PoseKanan.jpeg	Aldinata	0,45373562
Aldinata_PoseKiri.jpeg	Aldinata	0,5844573
Aldinata_Sedih.jpeg	Aldinata	0,54445815
Aldinata_Senyum.jpeg	Aldinata	0,79648614

5.4.4.4 Uji Coba Skenario B3 (penambahan data latih pose kanan dan pose kiri)

Uji coba yang dilakukan pada skenario B3 menggunakan data latih yaitu 18 citra wajah asli dengan 9 identitas wajah dan 1 kondisi (netral) ditambah 54 citra wajah dari hasil augmentasi data GAN dengan 9 identitas wajah dan 2 kondisi (pose kanan dan pose kiri). Spesifikasi data latih pada skenario B3 dapat dilihat pada Tabel 5.17. Gambar plot hasil fitting model pengenalan wajah skenario B3 dapat dilihat pada Gambar 5.18.

Tabel 5.17 Spesifikasi data latih skenario B3

Identitas Wajah	Data Latih	Keterangan
Aldinata	2 citra asli, 6 citra GAN	3 kondisi
Ardo	2 citra asli, 6 citra GAN	3 kondisi
Fahmi	2 citra asli, 6 citra GAN	3 kondisi
Fasma	2 citra asli, 6 citra GAN	3 kondisi
Handika	2 citra asli, 6 citra GAN	3 kondisi
Rasyid	2 citra asli, 6 citra GAN	3 kondisi
Steve	2 citra asli, 6 citra GAN	3 kondisi
Sugiarto	2 citra asli, 6 citra GAN	3 kondisi
Yuda	2 citra asli, 6 citra GAN	3 kondisi



Gambar 5.18 Plot hasil fitting model pengenalan wajah skenario B3

Evaluasi performa model *classifier* menggunakan nilai *Precision*, *Recall*, *F1-Score*, dan *Accuracy*. Hasil evaluasi pada skenario B3 dapat dilihat pada Tabel 5.18.

Tabel 5.18 Hasil evaluasi pada skenario B3

	Precision	Recall	F1-Score	Support
Aldinata	1,00	0,83	0,91	6
Ardo	1,00	0,50	0,67	6
Fahmi	0,67	1,00	0,80	6
Fasma	0,83	0,83	0,83	6
Handika	1,00	0,83	0,91	6
Rasyid	0,67	1,00	0,80	6
Steve	0,71	0,83	0,77	6
Sugiarto	1,00	0,67	0,80	6
Yuda	1,00	1,00	1,00	6
accuracy			0,83	54
macro avg	0,88	0,83	0,83	54
weighted avg	0,88	0,83	0,83	54

Dari hasil evaluasi pada skenario B3 maka didapatkan nilai rata-rata *Precision* 0,88; *Recall* 0,83; *F1-Score* 0,83; dan *Accuracy* 0,83. Contoh hasil prediksi pada salah satu identitas wajah skenario B3 dapat dilihat pada Tabel 5.19. Skor yang dimaksud adalah nilai output prediksi tertinggi dari model *classifier* pengenalan wajah.

Tabel 5.19 Hasil prediksi pada salah satu identitas wajah Skenario B3

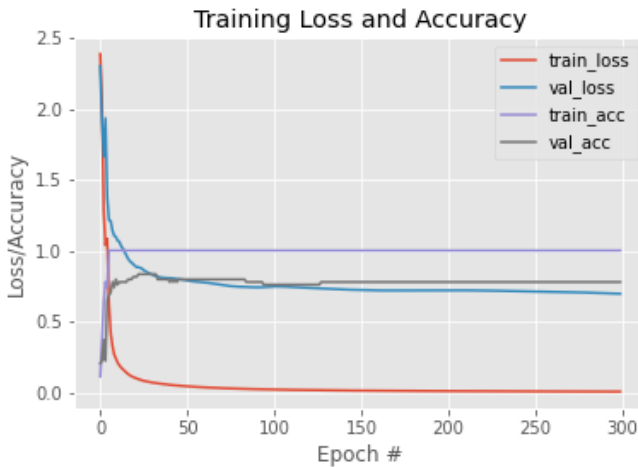
Nama File Citra Wajah	Hasil Prediksi	Skor
Aldinata_Kacamata.jpeg	Rasyid	0,26211995
Aldinata_Netral.jpeg	Aldinata	0,979764
Aldinata_PoseKanan.jpeg	Aldinata	0,9370881
Aldinata_PoseKiri.jpeg	Aldinata	0,34400073
Aldinata_Sedih.jpeg	Aldinata	0,6175791
Aldinata_Senyum.jpeg	Aldinata	0,7716424

5.4.4.5 Uji Coba Skenario B4 (penambahan data latih atribut kacamata)

Uji coba yang dilakukan pada skenario B4 menggunakan data latih yaitu 18 citra wajah asli dengan 9 identitas wajah dan 1 kondisi (netral) ditambah 9 citra wajah dari hasil augmentasi data GAN dengan 9 identitas wajah dan 1 kondisi (atribut kacamata). Spesifikasi data latih pada skenario B4 dapat dilihat pada Tabel 5.20. Gambar plot hasil fitting model pengenalan wajah skenario B4 dapat dilihat pada Gambar 5.19.

Tabel 5.20 Spesifikasi data latih skenario B4

Identitas Wajah	Data Latih	Keterangan
Aldinata	2 citra asli, 1 citra GAN	2 kondisi
Ardo	2 citra asli, 1 citra GAN	2 kondisi
Fahmi	2 citra asli, 1 citra GAN	2 kondisi
Fasma	2 citra asli, 1 citra GAN	2 kondisi
Handika	2 citra asli, 1 citra GAN	2 kondisi
Rasyid	2 citra asli, 1 citra GAN	2 kondisi
Steve	2 citra asli, 1 citra GAN	2 kondisi
Sugiarto	2 citra asli, 1 citra GAN	2 kondisi
Yuda	2 citra asli, 1 citra GAN	2 kondisi



Gambar 5.19 Plot hasil fitting model pengenalan wajah skenario B4

Evaluasi performa model *classifier* menggunakan nilai *Precision*, *Recall*, *F1-Score*, dan *Accuracy*. Hasil evaluasi pada skenario B4 dapat dilihat pada Tabel 5.21.

Tabel 5.21 Hasil evaluasi pada skenario B4

	Precision	Recall	F1-Score	Support
Aldinata	1,00	0,67	0,80	6
Ardo	1,00	0,33	0,50	6
Fahmi	0,60	0,50	0,55	6
Fasma	0,75	1,00	0,86	6
Handika	0,56	0,83	0,67	6
Rasyid	1,00	1,00	1,00	6
Steve	0,62	0,83	0,71	6
Sugiarto	0,83	0,83	0,83	6
Yuda	1,00	1,00	1,00	6
accuracy			0,78	54
macro avg	0,82	0,78	0,77	54
weighted avg	0,82	0,78	0,77	54

Dari hasil evaluasi pada skenario B4 maka didapatkan nilai rata-rata *Precision* 0,82; *Recall* 0,78; *F1-Score* 0,77; dan *Accuracy* 0,78. Contoh hasil prediksi pada salah satu identitas wajah skenario B4 dapat dilihat pada Tabel 5.22. Skor yang dimaksud adalah nilai output prediksi tertinggi dari model *classifier* pengenalan wajah.

Tabel 5.22 Hasil prediksi pada salah satu identitas wajah Skenario B4

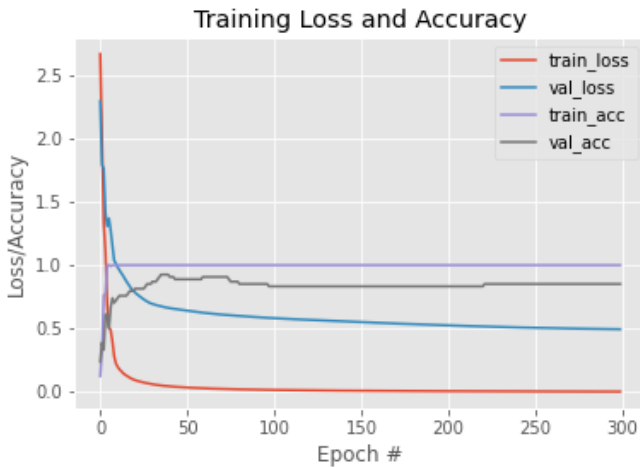
Nama File Citra Wajah	Hasil Prediksi	Skor
Aldinata_Kacamata.jpeg	Aldinata	0,8104978
Aldinata_Netral.jpeg	Aldinata	0,9387545
Aldinata_PoseKanan.jpeg	Aldinata	0,8004558
Aldinata_PoseKiri.jpeg	Sugiarto	0,36279157
Aldinata_Sedih.jpeg	Aldinata	0,41804722
Aldinata_Senyum.jpeg	Fahmi	0,3962862

5.4.4.6 Uji Coba Skenario B5 (penambahan data latihan ekspresi senyum dan sedih, pose kanan dan kiri)

Uji coba yang dilakukan pada skenario B5 menggunakan data latihan yaitu 18 citra wajah asli dengan 9 identitas wajah dan 1 kondisi (netral) ditambah 108 citra wajah dari hasil augmentasi data GAN dengan 9 identitas wajah dan 4 kondisi (ekspresi senyum dan sedih, pose kanan dan kiri). Spesifikasi data latihan pada skenario B5 dapat dilihat pada Tabel 5.23. Gambar plot hasil fitting model pengenalan wajah skenario B5 dapat dilihat pada Gambar 5.20.

Tabel 5.23 Spesifikasi data latihan skenario B5

Identitas Wajah	Data Latihan	Keterangan
Aldinata	2 citra asli, 12 citra GAN	5 kondisi
Ardo	2 citra asli, 12 citra GAN	5 kondisi
Fahmi	2 citra asli, 12 citra GAN	5 kondisi
Fasma	2 citra asli, 12 citra GAN	5 kondisi
Handika	2 citra asli, 12 citra GAN	5 kondisi
Rasyid	2 citra asli, 12 citra GAN	5 kondisi
Steve	2 citra asli, 12 citra GAN	5 kondisi
Sugiarto	2 citra asli, 12 citra GAN	5 kondisi
Yuda	2 citra asli, 12 citra GAN	5 kondisi



Gambar 5.20 Plot hasil fitting model pengenalan wajah skenario B5

Evaluasi performa model *classifier* menggunakan nilai *Precision*, *Recall*, *F1-Score*, dan *Accuracy*. Hasil evaluasi pada skenario B5 dapat dilihat pada Tabel 5.24.

Tabel 5.24 Hasil evaluasi pada skenario B5

	Precision	Recall	F1-Score	Support
Aldinata	1,00	0,83	0,91	6
Ardo	0,86	1,00	0,92	6
Fahmi	0,83	0,83	0,83	6
Fasma	1,00	0,83	0,91	6
Handika	1,00	0,50	0,67	6
Rasyid	0,86	1,00	0,92	6
Steve	0,60	1,00	0,75	6
Sugiarto	0,80	0,67	0,73	6
Yuda	1,00	1,00	1,00	6
accuracy			0,85	54
macro avg	0,88	0,85	0,85	54
weighted avg	0,88	0,85	0,85	54

Dari hasil evaluasi pada skenario B5 maka didapatkan nilai rata-rata *Precision* 0,88; *Recall* 0,85; *F1-Score* 0,85; dan *Accuracy* 0,85. Contoh hasil prediksi pada salah satu identitas wajah skenario B5 dapat dilihat pada Tabel 5.25. Skor yang dimaksud adalah nilai output prediksi tertinggi dari model *classifier* pengenalan wajah.

Tabel 5.25 Hasil prediksi pada salah satu identitas wajah Skenario B5

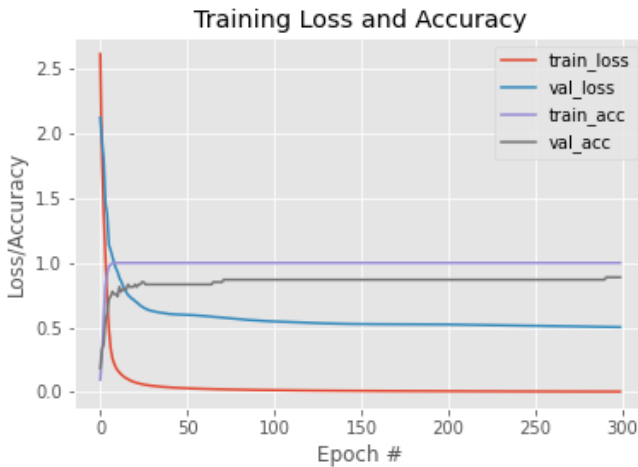
Nama File Citra Wajah	Hasil Prediksi	Skor
Aldinata_Kacamata.jpeg	Aldinata	0,7950637
Aldinata_Netral.jpeg	Aldinata	0,9192038
Aldinata_PoseKanan.jpeg	Aldinata	0,8953622
Aldinata_PoseKiri.jpeg	Steve	0,38646472
Aldinata_Sedih.jpeg	Aldinata	0,4849425
Aldinata_Senyum.jpeg	Aldinata	0,8626001

5.4.4.7 Uji Coba Skenario B6 (penambahan data latih seluruh jenis kondisi hasil augmentasi data)

Uji coba yang dilakukan pada skenario B6 menggunakan data latih yaitu 18 citra wajah asli dengan 9 identitas wajah dan 1 kondisi (netral) ditambah 117 citra wajah dari hasil augmentasi data GAN dengan 9 identitas wajah dan 5 kondisi (seluruh jenis kondisi hasil augmentasi data). Spesifikasi data latih pada skenario B6 dapat dilihat pada Tabel 5.26. Gambar plot hasil fitting model pengenalan wajah skenario B6 dapat dilihat pada Gambar 5.21.

Tabel 5.26 Spesifikasi data latih skenario B6

Identitas Wajah	Data Latih	Keterangan
Aldinata	2 citra asli, 12 citra GAN	6 kondisi
Ardo	2 citra asli, 12 citra GAN	6 kondisi
Fahmi	2 citra asli, 12 citra GAN	6 kondisi
Fasma	2 citra asli, 12 citra GAN	6 kondisi
Handika	2 citra asli, 12 citra GAN	6 kondisi
Rasyid	2 citra asli, 12 citra GAN	6 kondisi
Steve	2 citra asli, 12 citra GAN	6 kondisi
Sugiarto	2 citra asli, 12 citra GAN	6 kondisi
Yuda	2 citra asli, 12 citra GAN	6 kondisi



Gambar 5.21 Plot hasil fitting model pengenalan wajah skenario B6

Evaluasi performa model *classifier* menggunakan nilai *Precision*, *Recall*, *F1-Score*, dan *Accuracy*. Hasil evaluasi pada skenario B6 dapat dilihat pada Tabel 5.27.

Tabel 5.27 Hasil evaluasi pada skenario B6

	Precision	Recall	F1-Score	Support
Aldinata	1,00	1,00	1,00	6
Ardo	1,00	1,00	1,00	6
Fahmi	1,00	0,67	0,80	6
Fasma	0,75	1,00	0,86	6
Handika	0,71	0,83	0,77	6
Rasyid	1,00	1,00	1,00	6
Steve	0,75	1,00	0,86	6
Sugiarto	1,00	0,50	0,67	6
Yuda	1,00	1,00	1,00	6
accuracy			0,89	54
macro avg	0,91	0,89	0,88	54
weighted avg	0,91	0,89	0,88	54

Dari hasil evaluasi pada skenario B6 maka didapatkan nilai rata-rata *Precision* 0,91; *Recall* 0,89; *F1-Score* 0,88; dan *Accuracy* 0,89. Contoh hasil prediksi pada salah satu identitas wajah skenario B6 dapat dilihat pada Tabel 5.28. Skor yang dimaksud adalah nilai output prediksi tertinggi dari model *classifier* pengenalan wajah.

Tabel 5.28 Hasil prediksi pada salah satu identitas wajah Skenario B6

Nama File Citra Wajah	Hasil Prediksi	Skor
Aldinata_Kacamata.jpeg	Aldinata	0,69615006
Aldinata_Netral.jpeg	Aldinata	0,9774539
Aldinata_PoseKanan.jpeg	Aldinata	0,86993045
Aldinata_PoseKiri.jpeg	Aldinata	0,40973344
Aldinata_Sedih.jpeg	Aldinata	0,8680885
Aldinata_Senyum.jpeg	Aldinata	0,89334804

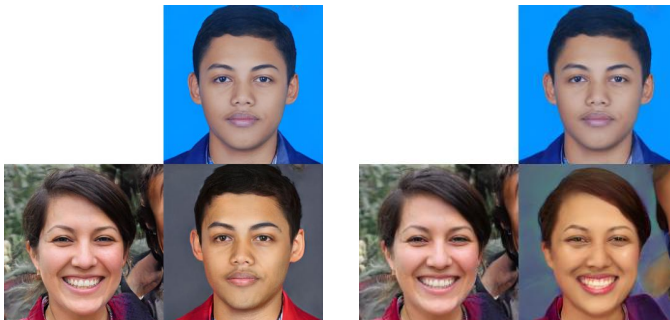
5.5 Hasil dan Evaluasi

Untuk uji coba pada tahap implementasi GAN, awalnya dilakukan dengan penggantian jenis *pretrained* StyleGAN, diperoleh nilai *loss* terkecil adalah pada penggunaan *pretrained* StyleGAN-FFHQ yaitu dengan nilai *loss* 0,25. ***Pretained* StyleGAN-FFHQ lebih baik digunakan pada model GAN yang telah dibangun dalam menghasilkan representasi *latent space* dari citra wajah. Hal ini diperkirakan karena jumlah data latih pada dataset FFHQ lebih banyak dibandingkan CelebA-HQ, serta variasi wajah mulai dari usia, etnis, latar belakang, dan atribut wajah pada dataset FFHQ juga lebih banyak (tidak hanya khusus citra selebritas seperti pada CelebA-HQ).**

Selanjutnya, dilakukan uji coba penggantian parameter model GAN. Pertama penggantian nilai ukuran *batch* pada model GAN, didapatkan nilai *loss* terkecil dengan mengubah **ukuran *batch* menjadi bernilai 1** yaitu dengan nilai *loss* 0,25. Kedua penggantian nilai *learning rate* pada model GAN, didapatkan nilai *loss* terkecil dengan mengubah ***learning rate* menjadi bernilai 1** yaitu dengan nilai *loss* 0,25. Ketiga penggantian nilai *random noise* pada model GAN, didapatkan nilai *loss* terkecil dengan mengubah

nilai *random noise* menjadi bernilai *False* yaitu dengan nilai *loss* 0,25. Keempat penggantian nilai jumlah iterasi pada model GAN, didapatkan nilai *loss* yang terkecil dengan mengubah **jumlah iterasi menjadi bernilai 4000** yaitu dengan nilai *loss* 0,15. Jika jumlah iterasi lebih dari 4000 model GAN akan mulai konvergen dan nilai *loss* model tidak dapat turun secara signifikan lagi.

Untuk uji coba pada tahap penyesuaian representasi *latent space* dilakukan dengan mencoba 4 metode dan evaluasi yang dilakukan adalah dengan kualitas visual dari hasil pembentukan ulang citra wajah karena tidak ada matriks evaluasi dalam penilaian kuantitatif dari hasil tersebut. Pertama dilakukan percobaan dengan metode *common average style mixing*, namun dari hasil yang didapatkan citra wajah mengalami perubahan identitas yang sangat terlihat. Kedua dilakukan percobaan dengan metode *vector operation in latent space*, namun dari hasil yang didapatkan citra wajah juga mengalami perubahan identitas yang sangat terlihat. Ketiga dilakukan percobaan dengan metode *NVIDIA's style mixing technique*, namun dari hasil yang didapatkan citra wajah tidak sesuai dengan gaya atau kondisi wajah yang diharapkan. Hal ini dikarenakan parameter *range style* pada metode tersebut tidak mendukung untuk mempertahankan identitas jika juga digunakan untuk perubahan gaya atau kondisi wajah (seperti yang terlihat pada Gambar 5.22).

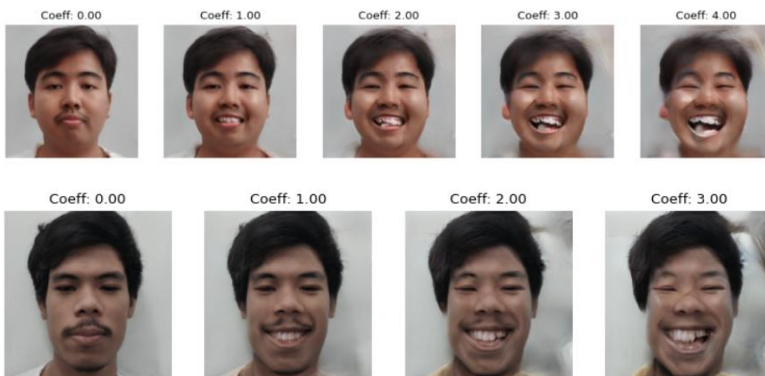


Gambar 5.22 Contoh gambar hasil citra wajah yang tidak sesuai seperti yang diharapkan pada metode *NVIDIA's style mixing technique*

Terakhir dilakukan percobaan dengan metode *latent direction*, hasil yang didapatkan citra wajah yang dihasilkan tidak mengalami perubahan identitas yang terlihat atau signifikan ketika penambahan gaya atau kondisi pada wajah. Jadi didapatkan bahwa **metode latent direction lebih baik digunakan untuk melakukan penyesuaian *latent space* dari hasil GAN.**

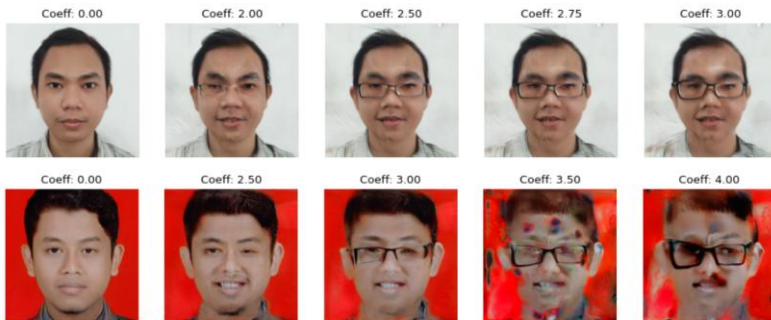
Untuk hasil uji coba pada proses augmentasi data terhadap 9 citra wajah dengan identitas yang berbeda-beda, metode *latent direction* sudah cukup baik untuk menghasilkan citra wajah sesuai dengan gaya atau kondisi yang diharapkan (ekspresi senyum, ekspresi sedih, pose kanan, pose kiri, dan atribut kacamata).

Namun dalam penggunaan metode *latent direction* ini pemilihan koefisien perkalian yang digunakan memiliki beberapa nilai batas agar tidak merubah identitas dan berbeda-beda untuk setiap identitas wajah dan gaya atau kondisi yang digunakan. Sehingga masih dilakukan penyesuaian koefisien perkalian yang digunakan secara visual yaitu dengan membandingkan hasil pembentukan ulang citra wajah. Contoh hasil penggunaan koefisien perkalian yang melebihi batas pada metode *latent direction* dapat dilihat pada Gambar 5.23.



Gambar 5.23 Contoh hasil citra wajah pada metode *latent direction* dengan koefisien perkalian yang melebihi batas

Selain itu, untuk penambahan gaya atau kondisi yang lebih kompleks seperti penambahan atribut kacamata, *latent direction* tetap lebih unggul dibandingkan metode yang lain. Namun tetap saja untuk mendapatkan hasil citra wajah yang sesuai dengan yang diharapkan harus memilih koefisien pada *latent direction* secara tepat dan hati-hati. Apalagi untuk kondisi yang kompleks batas koefien sangat sensitif dalam mempengaruhi perubahan identitas wajah dan bahkan bisa merusak citra wajah awal. Contoh hasil citra wajah dengan metode *latent direction* untuk gaya atau kondisi yang kompleks dapat dilihat pada Gambar 5.24.



Gambar 5.24 Contoh hasil citra wajah dengan metode *latent direction* untuk gaya atau kondisi yang kompleks

Untuk hasil uji coba pada proses pengenalan wajah bertujuan untuk membandingkan hasil performa pengenalan wajah setelah adanya penambahan data latih dengan hasil citra wajah dari GAN. Uji coba dibagi menjadi 7 kondisi skenario pada data latih terhadap model *classifier*. Hasil evaluasi dari keseluruhan skenario yang telah dicoba terangkum pada Tabel 5.29.

Tabel 5.29 Hasil evaluasi pada semua skenario proses pengenalan wajah

Skenario	Precision	Recall	F1-Score	Accuracy
A	0,79	0,74	0,73	0,74
B1	0,89	0,81	0,83	0,81
B2	0,89	0,83	0,84	0,83

Skenario	Precision	Recall	F1-Score	Accuracy
B3	0,88	0,83	0,83	0,83
B4	0,82	0,78	0,77	0,78
B5	0,88	0,85	0,85	0,85
B6	0,91	0,89	0,88	0,89

Karena data uji yang digunakan memiliki proporsi seimbang pada setiap kelas (setiap identitas wajah) yaitu 6 citra wajah pada setiap kelas, maka nilai evaluasi yang akan digunakan untuk membandingkan setiap skenario cukup dengan nilai *accuracy*.

Skenario A (tanpa ada penambahan data hasil augmentasi) memperoleh hasil *accuracy* yang paling rendah. Hal ini sesuai perkiraan awal bahwa **ketika data latih terbatas maka model *classifier* juga akan menghasilkan performa pengenalan wajah yang kurang baik**. Hasil evaluasi yang kecil diperkirakan juga dipengaruhi oleh variasi kondisi wajah serta kebaruan dari data uji yang digunakan pada setiap identitas wajah.

Skenario B1 (data latih awal ditambah hasil augmentasi data ekspresi senyum) memperoleh hasil *accuracy* yang lebih tinggi dibandingkan skenario A. Hal ini sesuai perkiraan awal bahwa ketika data latih ditambah dengan hasil augmentasi dari GAN, maka performa model *classifier* juga akan bertambah. Selain itu hasil ini juga menunjukkan bahwa **kondisi ekspresi senyum lumayan baik untuk meningkatkan performa model *classifier***.

Skenario B2 (data latih awal ditambah hasil augmentasi data ekspresi sedih) memperoleh hasil *accuracy* yang lebih tinggi dibandingkan skenario A dan B1. Hal ini sesuai perkiraan awal bahwa ketika data latih ditambah dengan hasil augmentasi dari GAN, maka performa model *classifier* juga akan bertambah. Selain itu, hasil ini juga menunjukkan **bahwa kondisi ekspresi sedih lebih baik ditambahkan pada data latih dibandingkan kondisi ekspresi senyum**.

Skenario B3 (data latih awal ditambah hasil augmentasi data pose kanan dan pose kiri) memperoleh hasil *accuracy* yang lebih

tinggi dibandingkan skenario A dan B1, serta bernilai sama dengan skenario B2. Hal ini sesuai perkiraan awal bahwa ketika data latih ditambah dengan hasil augmentasi dari GAN, maka performa model *classifier* juga akan bertambah. Selain itu, hasil ini juga menunjukkan bahwa **kondisi pose kanan dan pose kiri sama baiknya dengan kondisi ekspresi sedih untuk ditambahkan pada data latih.**

Skenario B4 (data latih awal ditambah hasil augmentasi data atribut kacamata) memperoleh hasil *accuracy* yang lebih tinggi dibandingkan skenario A, namun lebih rendah dibandingkan dengan skenario B1, B2, dan B3. Hal ini sesuai perkiraan awal bahwa ketika data latih ditambah dengan hasil augmentasi dari GAN, maka performa model *classifier* juga akan bertambah. Selain itu, hasil ini juga menunjukkan bahwa **kondisi atribut kacamata tetap bisa meningkatkan performa model *classifier* meskipun tidak sebaik dengan kondisi ekspresi senyum, ekspresi sedih, pose kanan, dan pose kiri.** Namun hal ini diperkirakan juga dipengaruhi oleh hasil sintesa citra wajah dari GAN untuk kondisi kompleks seperti atribut kacamata tidak bisa sebaik kondisi lainnya (sedikit merubah identitas wajah awal atau bahkan merusak bagian dari citra wajah awal).

Skenario B5 (data latih awal ditambah hasil augmentasi data ekspresi senyum, ekspresi sedih, pose kanan, dan pose kiri) memperoleh hasil *accuracy* yang lebih tinggi dibandingkan skenario A, B1, B2, B3, dan B4. Hal ini sesuai perkiraan awal bahwa ketika data latih ditambah dengan hasil augmentasi dari GAN, maka performa model *classifier* juga akan bertambah. Selain itu, hasil ini juga menunjukkan bahwa **semakin banyak kondisi wajah yang ditambahkan pada data latih bisa meningkatkan performa model *classifier*.**

Skenario B6 (data latih awal ditambah seluruh jenis kondisi hasil augmentasi data) memperoleh hasil *accuracy* yang lebih tinggi dibandingkan skenario A, B1, B2, B3, B4, dan B5. Hal ini sesuai perkiraan awal bahwa ketika data latih ditambah dengan

hasil augmentasi dari GAN, maka performa model *classifier* juga akan bertambah. Selain itu, hasil ini juga menunjukkan bahwa **penambahan seluruh jenis kondisi hasil augmentasi citra wajah dari GAN pada data latih pengenalan wajah bisa meningkatkan performa model *classifier*.**

Dari seluruh hasil uji coba yang telah dilakukan, pada Tabel 5.30 ditetapkan parameter optimal untuk arsitektur *Generative Adversarial Network* (GAN). Kemudian ditetapkan metode yang paling baik dalam melakukan penyesuaian *latent space* untuk menghasilkan citra wajah dengan menambah kondisi yang diharapkan adalah metode ***latent direction***, jenis kondisi citra wajah hasil GAN yang paling baik memperbaiki performa pengenalan wajah adalah **ekspresi sedih, pose kanan dan kiri**. Skenario terbaik untuk meningkatkan performa pengenalan wajah adalah **Skenario B6** yaitu menambahkan seluruh jenis kondisi hasil augmentasi citra wajah dari GAN.

Tabel 5.30 Parameter GAN optimal yang ditetapkan

Keterangan	Parameter Optimal
Jenis <i>pre-trained</i>	StyleGAN-FFHQ
Ukuran <i>batch</i>	1
<i>Learning rate</i>	1
<i>Random noise</i>	<i>False</i>
Jumlah iterasi	4000

BAB VI

KESIMPULAN DAN SARAN

Bab ini membahas tentang kesimpulan yang didasari oleh hasil uji coba yang telah dilakukan terhadap sistem sintesa citra wajah menggunakan metode *Generative Adversarial Network* (GAN) dan sistem pengenalan wajah menggunakan metode *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 mendatang baik oleh pengusul atau pihak lain.

6.1 Kesimpulan

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

1. Berdasarkan uji coba pada tahap implementasi GAN maka pembuatan model GAN dilakukan melalui metode *transfer learning* terhadap *pretrained* StyleGAN dari NVIDIA kemudian mengambil model *generator* dari GAN dan dikombinasikan dengan model *perceptual* untuk menghasilkan representasi *latent space* dari citra wajah, serta didapatkan parameter optimal pada implementasi GAN mampu menghasilkan nilai *loss* 0,15 yaitu dengan *pretrained* StyleGAN-FFHQ, ukuran *batch* = 1, *learning rate* = 1, *random noise* = *false*, dan jumlah iterasi = 4000. Kemudian untuk membentuk citra wajah kembali adalah dengan menggunakan model *generator* untuk mengubah representasi *latent space* yang sudah disesuaikan menjadi citra wajah yang diharapkan.
2. Berdasarkan uji coba pada penyesuaian representasi *latent space* maka didapatkan metode yang paling baik adalah *latent direction*, karena cukup baik dalam mempertahankan identitas wajah asli. Metode *latent direction* adalah menambahkan *latent space* target (gaya atau kondisi yang diumpamakan

sebagai arah) terhadap *latent space* awal dan dapat diatur dengan mengubah nilai koefisien perkalian terhadap *latent space* target. Kemudian berdasarkan hasil uji coba pada proses augmentasi data, model GAN dan metode *latent direction* mampu melakukan sintesa citra wajah dengan baik sesuai dengan gaya atau kondisi yang diharapkan yaitu menambah ekspresi senyum, ekspresi sedih, pose kanan, pose kiri, dan atribut kacamata pada citra wajah awal.

3. Berdasarkan hasil uji coba pada proses pengenalan wajah didapatkan bahwa penambahan data latih dengan hasil sintesa citra wajah dari GAN sebagai bentuk dari augmentasi data mampu meningkatkan performa dari model *classifier* pengenalan wajah, yaitu pada skenario terbaik (Skenario B6: menambahkan seluruh jenis kondisi hasil augmentasi citra wajah dari GAN) mampu meningkatkan nilai *accuracy* dari model *classifier* yang awalnya 0,74 bertambah menjadi 0,89.

6.2 Saran

Saran yang diberikan untuk pengembangan sistem sintesa citra wajah menggunakan *Generative Adversarial Network* (GAN) untuk augmentasi data pada aplikasi pengenalan wajah, yaitu:

1. Mencari atau membuat model *Generative Adversarial Network* (GAN) yang lebih handal dalam menghasilkan representasi *latent space* dari sebuah citra wajah asli sebagai target dan membentuk ulang citra wajah tersebut hingga terlihat mirip dengan aslinya atau dengan nilai *loss* yang sangat kecil. Untuk sementara pada Tugas Akhir ini, pembuatan model GAN masih dilakukan melalui metode *transfer learning* terhadap *pretrained* StyleGAN dari NVIDIA dan belum bisa membangun serta melatih model GAN dari awal dikarenakan keterbatasan perangkat keras yang tersedia dan waktu yang diperlukan.
2. Melakukan eksplorasi parameter pada GAN, selain ukuran *batch*, *learning rate*, *random noise*, dan jumlah iterasi, yang

dapat menambah performa GAN seperti *loss* atau *activation function*, dan *optimizer*. Hal ini terutama jika sudah mampu melakukan saran nomor satu untuk membuat serta melatih model GAN dari awal.

3. Melakukan eksplorasi dan pengembangan pada metode *latent direction* dalam melakukan penyesuaian *latent space* atau proses sintesa wajah yang lebih baik, yaitu lebih handal untuk mempertahankan identitas wajah meskipun dengan penambahan gaya atau kondisi yang kompleks seperti pemakaian atribut. Untuk sementara pada Tugas Akhir ini, penentuan koefisien perkalian yang digunakan pada metode *latent direction* masih dilakukan secara manual berdasarkan evaluasi secara visual dari hasil sintesa citra wajah.
4. Menambah jenis gaya atau kondisi untuk sintesa citra wajah misalnya ekspresi kaget, takut, marah, penambahan rias pada wajah atau bahkan penambahan atribut lain pada wajah seperti topi dan masker. Hal ini terutama jika sudah mampu melakukan saran nomor tiga untuk membuat metode *latent direction* agar lebih handal untuk mempertahankan identitas wajah ataupun menemukan metode lainnya yang lebih handal.

(Halaman ini sengaja dikosongkan)

DAFTAR PUSTAKA

- [1] S. Arya, N. Pratap dan K. Bhatia, "Future of Face Recognition: A Review," *Procedia Computer Science*, vol. 58, pp. 578-585, 2015.
- [2] Y. Li, G. Su dan Y. Shang, "Generating Optimal Face Image in Face Recognition System," *In-Tech: Recent Advances in Face Recognition*, pp. 71-78, 2008.
- [3] F. U. Nuha dan Afiahayati, "Training Dataset Reduction On Generative Adversarial Network," *Procedia Computer Science*, vol. 144, pp. 133-139, 2018.
- [4] Z. Pan, W. Yu, X. Yi, A. Khan, F. Yuan dan Y. Zheng, "Recent Progress on Generative Adversarial Networks (GANs): A Survey," *IEEE*, vol. 7, pp. 36322-36333, 2019.
- [5] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville dan Y. Bengio, "Generative Adversarial Networks," *NIPS*, 2014.
- [6] J. Brownlee, "A Gentle Introduction to StyleGAN the Style Generative Adversarial Network," *machinelearningmastery.com*, 9 Agustus 2019. [Online]. Available: <https://machinelearningmastery.com/introduction-to-style-generative-adversarial-network-stylegan/>. [Diakses 5 Juni 2020].
- [7] J. Hui, "GAN — StyleGAN & StyleGAN2," *medium.com*, 9 Maret 2020. [Online]. Available: https://medium.com/@jonathan_hui/gan-stylegan-stylegan2-479bdf256299. [Diakses 5 Juni 2020].
- [8] E. Tiu, "Understanding Latent Space in Machine Learning," *towardsdatascience.com*, 4 Februari 2020. [Online]. Available: <https://towardsdatascience.com/understanding-latent->

- space-in-machine-learning-de5a7c687d8d. [Diakses 7 Juni 2020].
- [9] J. Brownlee, "How to Explore the GAN Latent Space When Generating Faces," [machinelearningmastery.com](https://machinelearningmastery.com/how-to-interpolate-and-perform-vector-arithmetic-with-faces-using-a-generative-adversarial-network/), 3 Juli 2019. [Online]. Available: <https://machinelearningmastery.com/how-to-interpolate-and-perform-vector-arithmetic-with-faces-using-a-generative-adversarial-network/>. [Diakses 7 Juni 2020].
- [10] R. Horev, "Explained: A Style-Based Generator Architecture for GANs - Generating and Tuning Realistic Artificial Faces," [towardsdatascience.com](https://towardsdatascience.com/explained-a-style-based-generator-architecture-for-gans-generating-and-tuning-realistic-6cb2be0f431), 30 Desember 2018. [Online]. Available: <https://towardsdatascience.com/explained-a-style-based-generator-architecture-for-gans-generating-and-tuning-realistic-6cb2be0f431>. [Diakses 5 Juni 2020].
- [11] S. Sena, "Pengenalan Deep Learning Part 7: Convolutional Neural Network (CNN)," [medium.com](https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94), 13 November 2017. [Online]. Available: <https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94>. [Diakses 3 Januari 2020].
- [12] "Convolutional Neural Network: 3 Things You Need To Know," [mathworks.com](https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html), [Online]. Available: <https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html>. [Diakses 3 Januari 2020].
- [13] S. Sena, "Pengenalan Deep Learning Part 1: Neural Network," [medium.com](https://medium.com/@samuelsena/pengenalan-deep-learning-8fbb7d8028ac), 28 Oktober 2017. [Online]. Available: <https://medium.com/@samuelsena/pengenalan-deep-learning-8fbb7d8028ac>. [Diakses 3 Januari 2020].
- [14] S. Narkhede, "Understanding Confusion Matrix," [towardsdatascience.com](https://towardsdatascience.com/understanding-confusion-matrix), 9 Mei 2018. [Online].

- Available:
<https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62>. [Diakses 5 Januari 2020].
- [15] J. Brownlee, “Data, Learning, and Modeling,” machinelearningmastery.com, 6 Januari 2017. [Online]. Available: <https://machinelearningmastery.com/data-learning-and-modeling/>. [Diakses 5 Januari 2020].
- [16] P. Marcelino, “Transfer Learning from Pre-Trained Models,” towardsdatascience.com, 23 Oktober 2019. [Online]. Available: <https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751>. [Diakses 1 Juni 2020].
- [17] “About Python,” Python, [Online]. Available: <https://www.python.org/about/>. [Diakses 1 Januari 2020].
- [18] “Keras: The Python Deep Learning library,” Keras, [Online]. Available: <https://keras.io/>. [Diakses 1 Januari 2020].
- [19] “TensorFlow,” TensorFlow, [Online]. Available: <https://www.tensorflow.org/>. [Diakses 1 Januari 2020].
- [20] “OpenCV,” [Online]. Available: <https://opencv.org/>. [Diakses 1 Januari 2020].
- [21] “NumPy,” NumPy, [Online]. Available: <http://www.numpy.org/>. [Diakses 1 Januari 2020].
- [22] “Scikit-learn,” Scikit-learn, [Online]. Available: <http://scikit-learn.org/stable/index.html>. [Diakses 1 Januari 2020].
- [23] “Matplotlib,” Matplotlib, [Online]. Available: <https://matplotlib.org/index.html>. [Diakses 1 Januari 2020].
- [24] “Scikit-image,” Scikit-image, [Online]. Available: <https://scikit-image.org>. [Diakses 1 Januari 2020].

- [25] “Sci-Py.org,” Sci-Py.org, [Online]. Available: <https://www.scipy.org/about.html>. [Diakses 1 Januari 2020].
- [26] The Chinese University of Hong Kong, “CelebFaces Attributes (CelebA) Dataset,” Multimedia Laboratory, 11 Oktober 2018. [Online]. Available: <http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html>. [Diakses 1 Januari 2020].
- [27] NVIDIA Corporation, “Flickr-Faces-HQ Dataset (FFHQ),” NVlabs, 5 Februari 2019. [Online]. Available: <https://github.com/NVLabs/ffhq-dataset>. [Diakses 1 Januari 2020].
- [28] A. Rosebrock, “Facial Landmarks with DLib, OpenCV, and Python,” 3 April 2017. [Online]. Available: <https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/>. [Diakses 5 Januari 2020].
- [29] T. Karras, S. Laine dan T. Aila, “A Style-Based Generator Architecture for Generative Adversarial Networks,” *IEEE: CVPR*, 2019.
- [30] S. I. Serengil, “Deep Face Recognition with Keras,” 6 Agustus 2018. [Online]. Available: <https://sefiks.com/2018/08/06/deep-face-recognition-with-keras/>. [Diakses 5 Juni 2020].
- [31] L. N. Santha, “Face Recognition with VGG-Face in Keras,” medium.com, 16 Oktober 2019. [Online]. Available: <https://medium.com/analytics-vidhya/face-recognition-with-vgg-face-in-keras-96e6bc1951d5>. [Diakses 5 Juni 2020].
- [32] Y. Choi, Y. Uh, J. Yoo dan J.-W. Ha, “StarGAN v2: Diverse Image Synthesis for Multiple Domains,” *IEEE: CVPR*, 2020.

- [33] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen dan T. Aila, “Analyzing and Improving the Image Quality of StyleGAN,” *IEEE: CVPR*, 2020.
- [34] A. Karnewar dan O. Wang, “MSG-GAN: Multi-Scale Gradients for Generative Adversarial Networks,” *IEEE: CVPR*, 2019.
- [35] A. Brock, J. Donahue dan K. Simonyan, “Large Scale GAN Training for High Fidelity Natural Image Synthesis,” *ArXiv*, vol. abs/1809.11096v2, 2019.
- [36] L. Tran, X. Yin dan X. Liu, “Representation Learning by Rotating Your Faces,” *IEEE: TPAMI*, vol. 41, no. 12, pp. 3007-3021, 2019.
- [37] R. Huang, S. Zhang, T. Li dan R. He, “Beyond Face Rotation: Global and Local Perception GAN for Photorealistic and Identity Preserving Frontal View Synthesis,” *IEEE: International Conference on Computer Vision (ICCV)*, pp. 2458-2467, 2017.
- [38] R. Sun, C. Huang, J. Shi dan L. Ma, “Mask-aware Photorealistic Face Attribute Manipulation,” *ArXiv*, vol. abs/1804.08882, 2018.

(Halaman ini sengaja dikosongkan)

LAMPIRAN

L.1 Dataset Sumber Citra Wajah Asli Sebagai Input Untuk Dilakukan Sintesa Citra Wajah Dengan GAN



L.2 Hasil Praproses Citra Wajah Untuk Siap Digunakan Sebagai Input Pada GAN



L.3 Hasil Semua Uji Coba pada tahap implementasi Generative Adversarial Network (GAN)

Jenis <i>Pretrained</i>	Ukuran <i>Batch</i>	Learning Rate	Random Noise	Jumlah Iterasi	<i>Loss</i>
StyleGAN-CelebAHQ	1	1	False	1000	0,28
StyleGAN-CelebAHQ	1	1	False	4000	0,18
StyleGAN-CelebAHQ	1	1	True	1000	0,29
StyleGAN-CelebAHQ	1	1	True	4000	0,19
StyleGAN-FFHQ	1	1	False	500	0,42
StyleGAN-FFHQ	1	1	False	1000	0,25
StyleGAN-FFHQ	1	1	False	2000	0,21
StyleGAN-FFHQ	1	1	False	4000	0,15
StyleGAN-FFHQ	1	1	False	8000	0,15
StyleGAN-FFHQ	1	1	True	500	0,44
StyleGAN-FFHQ	1	1	True	1000	0,26
StyleGAN-FFHQ	1	1	True	2000	0,23
StyleGAN-FFHQ	1	1	True	4000	0,16
StyleGAN-FFHQ	1	1	True	8000	0,16
StyleGAN-FFHQ	1	0,5	False	1000	0,28
StyleGAN-FFHQ	1	0,1	False	1000	0,39

StyleGAN- FFHQ	1	0,01	False	1000	0,67
StyleGAN- FFHQ	2	1	False	1000	0,32
StyleGAN- FFHQ	2	1	False	2000	0,28
StyleGAN- FFHQ	2	1	True	1000	0,27
StyleGAN- FFHQ	2	1	True	2000	0,29
StyleGAN- FFHQ	2	0,5	False	1000	0,34
StyleGAN- FFHQ	2	0,5	False	2000	0,30
StyleGAN- FFHQ	2	0,5	True	1000	0,29
StyleGAN- FFHQ	2	0,5	True	2000	0,31

L.4 Hasil Sintesa Citra Wajah Setelah Diproses Melalui GAN, Penyesuaian Latent Space, dan Pembentukan Ulang Citra Wajah

❖ Identitas Wajah Aldinata



❖ Identitas Wajah Ardo



❖ Identitas Wajah Fahmi



❖ Identitas Wajah Fasma



❖ Identitas Wajah Handika



❖ Identitas Wajah Rasyid



❖ Identitas Wajah Steve



❖ Identitas Wajah Sugiarto

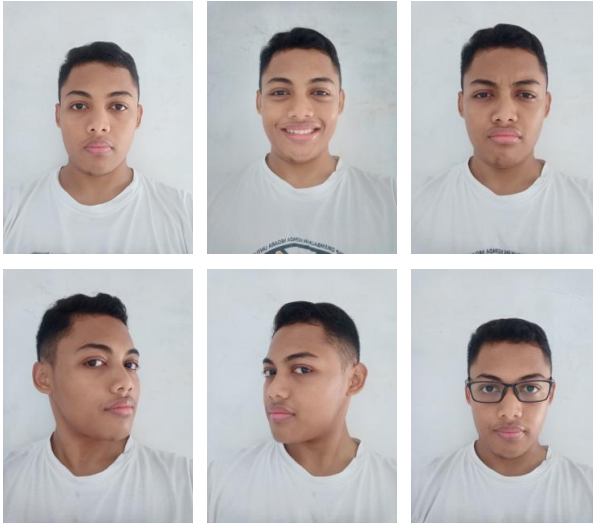


❖ Identitas Wajah Yuda



L.5 Dataset Sumber Citra Wajah Asli Sebagai Data Uji Pada Uji Coba Pengenalan Wajah

❖ Identitas Wajah Aldinata



❖ Identitas Wajah Ardo



❖ Identitas Wajah Fahmi



❖ Identitas Wajah Fasma



❖ Identitas Wajah Handika



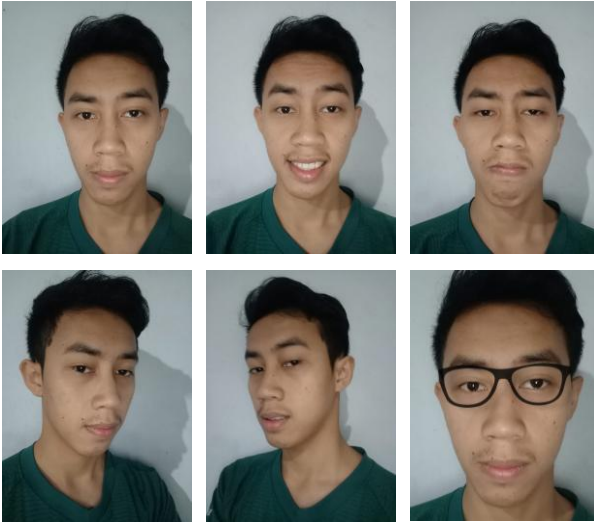
❖ Identitas Wajah Rasyid



❖ Identitas Wajah Steve



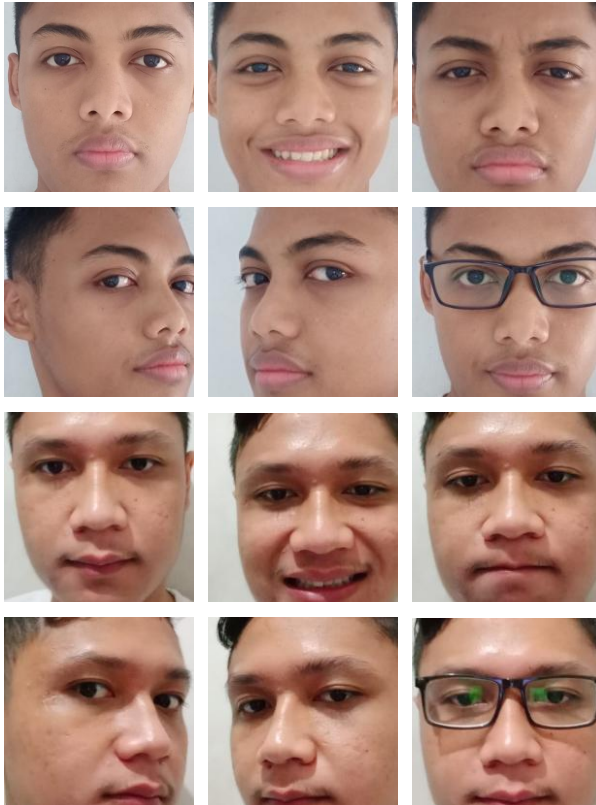
❖ Identitas Wajah Sugiarto



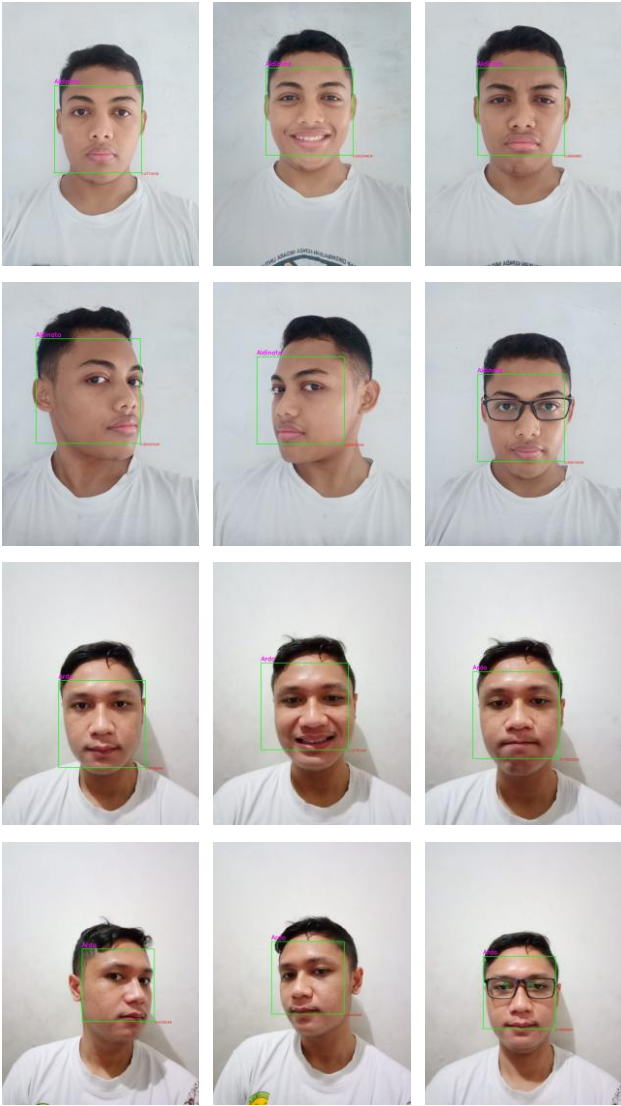
❖ Identitas Wajah Yuda



L.6 Contoh Hasil Face Detection and Extraction Pada Proses Pengenalan Wajah



L.7 Contoh Hasil Face Classification Pada Proses Pengenalan Wajah



L.8 Hasil Lengkap Face Classification Pada Semua Skenario Uji Coba Pengenalan Wajah

❖ Hasil Klasifikasi Pengenalan Wajah Skenario A (Skor adalah nilai output prediksi tertinggi dari model *classifier*)

Data uji	Hasil Prediksi	Skor
Aldinata_Kacamata.jpeg	Fasma	0,6058979
Aldinata_Netral.jpeg	Aldinata	0,9568714
Aldinata_PoseKanan.jpeg	Aldinata	0,8509558
Aldinata_PoseKiri.jpeg	Sugiarto	0,68165606
Aldinata_Sedih.jpeg	Aldinata	0,6553414
Aldinata_Senyum.jpeg	Aldinata	0,495758
Ardo_Kacamata.jpeg	Fahmi	0,8456218
Ardo_Netral.jpeg	Fahmi	0,3295116
Ardo_PoseKanan.jpeg	Ardo	0,28090858
Ardo_PoseKiri.jpeg	Ardo	0,4748223
Ardo_Sedih.jpeg	Ardo	0,44062477
Ardo_Senyum.jpeg	Ardo	0,53250974
Fahmi_Kacamata.jpeg	Steve	0,60745084
Fahmi_Netral.jpeg	Fahmi	0,9132889
Fahmi_PoseKanan.jpeg	Fasma	0,6551249
Fahmi_PoseKiri.jpeg	Fahmi	0,792423
Fahmi_Sedih.jpeg	Steve	0,593361
Fahmi_Senyum.jpeg	Fahmi	0,6969982
Fasma_Kacamata.jpeg	Fasma	0,71068877
Fasma_Netral.jpeg	Fasma	0,992663
Fasma_PoseKanan.jpeg	Fasma	0,9864254
Fasma_PoseKiri.jpeg	Fasma	0,9936079
Fasma_Sedih.jpeg	Fasma	0,99399817
Fasma_Senyum.jpeg	Fasma	0,995154
Handika_Kacamata.jpeg	Fahmi	0,27505243

Data uji	Hasil Prediksi	Skor
Handika_Netral.jpeg	Handika	0,83690894
Handika_PoseKanan.jpeg	Handika	0,42748153
Handika_PoseKiri.jpeg	Handika	0,66089386
Handika_Sedih.jpeg	Handika	0,46947935
Handika_Senyum.jpeg	Fasma	0,3932256
Rasyid_Kacamata.jpeg	Rasyid	0,7913891
Rasyid_Netral.jpeg	Rasyid	0,9915119
Rasyid_PoseKanan.jpeg	Rasyid	0,9932783
Rasyid_PoseKiri.jpeg	Rasyid	0,98939276
Rasyid_Sedih.jpeg	Rasyid	0,65529937
Rasyid_Senyum.jpeg	Rasyid	0,9922808
Steve_Kacamata.jpeg	Steve	0,2481676
Steve_Netral.jpeg	Steve	0,9683382
Steve_PoseKanan.jpeg	Steve	0,9662469
Steve_PoseKiri.jpeg	Steve	0,9711146
Steve_Sedih.jpeg	Steve	0,97543544
Steve_Senyum.jpeg	Steve	0,93260777
Sugiarto_Kacamata.jpeg	Fasma	0,26390183
Sugiarto_Netral.jpeg	Steve	0,83027625
Sugiarto_PoseKanan.jpeg	Steve	0,5438103
Sugiarto_PoseKiri.jpeg	Sugiarto	0,46104997
Sugiarto_Sedih.jpeg	Steve	0,9139561
Sugiarto_Senyum.jpeg	Steve	0,8809516
Yuda_Kacamata.jpeg	Yuda	0,7235338
Yuda_Netral.jpeg	Yuda	0,98321116
Yuda_PoseKanan.jpeg	Yuda	0,9447637
Yuda_PoseKiri.jpeg	Yuda	0,9543802
Yuda_Sedih.jpeg	Yuda	0,934127
Yuda_Senyum.jpeg	Yuda	0,7575333

❖ Hasil Klasifikasi Pengenalan Wajah Skenario B1 (Skor adalah nilai output prediksi tertinggi dari model *classifier*)

Data uji	Hasil Prediksi	Skor
Aldinata_Kacamata.jpeg	Fahmi	0,50613165
Aldinata_Netral.jpeg	Aldinata	0,76727897
Aldinata_PoseKanan.jpeg	Aldinata	0,69936275
Aldinata_PoseKiri.jpeg	Aldinata	0,25011402
Aldinata_Sedih.jpeg	Aldinata	0,29790172
Aldinata_Senyum.jpeg	Aldinata	0,74366385
Ardo_Kacamata.jpeg	Fahmi	0,9557363
Ardo_Netral.jpeg	Ardo	0,49974063
Ardo_PoseKanan.jpeg	Ardo	0,3261688
Ardo_PoseKiri.jpeg	Fahmi	0,66366655
Ardo_Sedih.jpeg	Ardo	0,8586815
Ardo_Senyum.jpeg	Ardo	0,6988038
Fahmi_Kacamata.jpeg	Fahmi	0,9650875
Fahmi_Netral.jpeg	Fahmi	0,95014393
Fahmi_PoseKanan.jpeg	Fahmi	0,6586732
Fahmi_PoseKiri.jpeg	Fahmi	0,9771666
Fahmi_Sedih.jpeg	Steve	0,48397103
Fahmi_Senyum.jpeg	Fahmi	0,8632437
Fasma_Kacamata.jpeg	Fahmi	0,78057706
Fasma_Netral.jpeg	Fasma	0,97048765
Fasma_PoseKanan.jpeg	Fasma	0,9891852
Fasma_PoseKiri.jpeg	Fasma	0,9834663
Fasma_Sedih.jpeg	Fasma	0,9921085
Fasma_Senyum.jpeg	Fasma	0,98812175
Handika_Kacamata.jpeg	Fahmi	0,38146135
Handika_Netral.jpeg	Handika	0,33800364
Handika_PoseKanan.jpeg	Handika	0,45968527

Data uji	Hasil Prediksi	Skor
Handika_PoseKiri.jpeg	Handika	0,46097878
Handika_Sedih.jpeg	Steve	0,36130407
Handika_Senyum.jpeg	Handika	0,48509803
Rasyid_Kacamata.jpeg	Rasyid	0,6909711
Rasyid_Netral.jpeg	Rasyid	0,9855531
Rasyid_PoseKanan.jpeg	Rasyid	0,9958305
Rasyid_PoseKiri.jpeg	Rasyid	0,8610997
Rasyid_Sedih.jpeg	Sugiarto	0,49427792
Rasyid_Senyum.jpeg	Rasyid	0,9938684
Steve_Kacamata.jpeg	Steve	0,35395604
Steve_Netral.jpeg	Steve	0,9810354
Steve_PoseKanan.jpeg	Steve	0,98404074
Steve_PoseKiri.jpeg	Steve	0,870375
Steve_Sedih.jpeg	Steve	0,98863876
Steve_Senyum.jpeg	Steve	0,9761545
Sugiarto_Kacamata.jpeg	Fahmi	0,5949376
Sugiarto_Netral.jpeg	Sugiarto	0,73705566
Sugiarto_PoseKanan.jpeg	Sugiarto	0,6943029
Sugiarto_PoseKiri.jpeg	Sugiarto	0,6210391
Sugiarto_Sedih.jpeg	Fahmi	0,3415594
Sugiarto_Senyum.jpeg	Sugiarto	0,67409366
Yuda_Kacamata.jpeg	Yuda	0,7780546
Yuda_Netral.jpeg	Yuda	0,94346493
Yuda_PoseKanan.jpeg	Yuda	0,8642926
Yuda_PoseKiri.jpeg	Yuda	0,9693011
Yuda_Sedih.jpeg	Yuda	0,72921175
Yuda_Senyum.jpeg	Yuda	0,3606639

❖ Hasil Klasifikasi Pengenalan Wajah Skenario B2 (Skor adalah nilai output prediksi tertinggi dari model *classifier*)

Data uji	Hasil Prediksi	Skor
Aldinata_Kacamata.jpeg	Fahmi	0,31963733
Aldinata_Netral.jpeg	Aldinata	0,8668376
Aldinata_PoseKanan.jpeg	Aldinata	0,45373562
Aldinata_PoseKiri.jpeg	Aldinata	0,5844573
Aldinata_Sedih.jpeg	Aldinata	0,54445815
Aldinata_Senyum.jpeg	Aldinata	0,79648614
Ardo_Kacamata.jpeg	Ardo	0,6712077
Ardo_Netral.jpeg	Ardo	0,858429
Ardo_PoseKanan.jpeg	Ardo	0,50759923
Ardo_PoseKiri.jpeg	Ardo	0,72809374
Ardo_Sedih.jpeg	Ardo	0,82480043
Ardo_Senyum.jpeg	Ardo	0,8319751
Fahmi_Kacamata.jpeg	Fahmi	0,7505872
Fahmi_Netral.jpeg	Fahmi	0,9802365
Fahmi_PoseKanan.jpeg	Fahmi	0,9005908
Fahmi_PoseKiri.jpeg	Fahmi	0,39547062
Fahmi_Sedih.jpeg	Fahmi	0,70640004
Fahmi_Senyum.jpeg	Fahmi	0,9754582
Fasma_Kacamata.jpeg	Rasyid	0,4293629
Fasma_Netral.jpeg	Fasma	0,9483589
Fasma_PoseKanan.jpeg	Fasma	0,98674715
Fasma_PoseKiri.jpeg	Fasma	0,9883713
Fasma_Sedih.jpeg	Fasma	0,98673254
Fasma_Senyum.jpeg	Fasma	0,99064106
Handika_Kacamata.jpeg	Fahmi	0,3459551
Handika_Netral.jpeg	Handika	0,8483452
Handika_PoseKanan.jpeg	Handika	0,7918634

Data uji	Hasil Prediksi	Skor
Handika_PoseKiri.jpeg	Handika	0,29515535
Handika_Sedih.jpeg	Handika	0,39820883
Handika_Senyum.jpeg	Handika	0,47058925
Rasyid_Kacamata.jpeg	Rasyid	0,97992754
Rasyid_Netral.jpeg	Rasyid	0,9901382
Rasyid_PoseKanan.jpeg	Rasyid	0,9923915
Rasyid_PoseKiri.jpeg	Rasyid	0,9587316
Rasyid_Sedih.jpeg	Rasyid	0,6897747
Rasyid_Senyum.jpeg	Rasyid	0,9903071
Steve_Kacamata.jpeg	Fahmi	0,7577454
Steve_Netral.jpeg	Steve	0,8066509
Steve_PoseKanan.jpeg	Steve	0,9833104
Steve_PoseKiri.jpeg	Steve	0,95795447
Steve_Sedih.jpeg	Steve	0,96742153
Steve_Senyum.jpeg	Steve	0,9805834
Sugiarto_Kacamata.jpeg	Fahmi	0,7552334
Sugiarto_Netral.jpeg	Sugiarto	0,83347064
Sugiarto_PoseKanan.jpeg	Sugiarto	0,88657326
Sugiarto_PoseKiri.jpeg	Steve	0,4116017
Sugiarto_Sedih.jpeg	Sugiarto	0,5344311
Sugiarto_Senyum.jpeg	Sugiarto	0,8250338
Yuda_Kacamata.jpeg	Rasyid	0,3334066
Yuda_Netral.jpeg	Yuda	0,96045035
Yuda_PoseKanan.jpeg	Yuda	0,5030336
Yuda_PoseKiri.jpeg	Yuda	0,9708321
Yuda_Sedih.jpeg	Steve	0,65847516
Yuda_Senyum.jpeg	Steve	0,92093986

❖ Hasil Klasifikasi Pengenalan Wajah Skenario B3 (Skor adalah nilai output prediksi tertinggi dari model *classifier*)

Data uji	Hasil Prediksi	Skor
Aldinata_Kacamata.jpeg	Rasyid	0,26211995
Aldinata_Netral.jpeg	Aldinata	0,979764
Aldinata_PoseKanan.jpeg	Aldinata	0,9370881
Aldinata_PoseKiri.jpeg	Aldinata	0,34400073
Aldinata_Sedih.jpeg	Aldinata	0,6175791
Aldinata_Senyum.jpeg	Aldinata	0,7716424
Ardo_Kacamata.jpeg	Fahmi	0,5592594
Ardo_Netral.jpeg	Steve	0,49701598
Ardo_PoseKanan.jpeg	Steve	0,5679757
Ardo_PoseKiri.jpeg	Ardo	0,31023672
Ardo_Sedih.jpeg	Ardo	0,61368275
Ardo_Senyum.jpeg	Ardo	0,37501106
Fahmi_Kacamata.jpeg	Fahmi	0,9011644
Fahmi_Netral.jpeg	Fahmi	0,9872502
Fahmi_PoseKanan.jpeg	Fahmi	0,40596998
Fahmi_PoseKiri.jpeg	Fahmi	0,5683916
Fahmi_Sedih.jpeg	Fahmi	0,35097942
Fahmi_Senyum.jpeg	Fahmi	0,89453346
Fasma_Kacamata.jpeg	Rasyid	0,5988237
Fasma_Netral.jpeg	Fasma	0,9490285
Fasma_PoseKanan.jpeg	Fasma	0,98044384
Fasma_PoseKiri.jpeg	Fasma	0,9736708
Fasma_Sedih.jpeg	Fasma	0,9857788
Fasma_Senyum.jpeg	Fasma	0,98342806
Handika_Kacamata.jpeg	Rasyid	0,45449993
Handika_Netral.jpeg	Handika	0,5691954
Handika_PoseKanan.jpeg	Handika	0,81883323

Data uji	Hasil Prediksi	Skor
Handika_PoseKiri.jpeg	Handika	0,64385855
Handika_Sedih.jpeg	Handika	0,6003908
Handika_Senyum.jpeg	Handika	0,47334796
Rasyid_Kacamata.jpeg	Rasyid	0,9796363
Rasyid_Netral.jpeg	Rasyid	0,99305046
Rasyid_PoseKanan.jpeg	Rasyid	0,99402666
Rasyid_PoseKiri.jpeg	Rasyid	0,94631934
Rasyid_Sedih.jpeg	Rasyid	0,88934225
Rasyid_Senyum.jpeg	Rasyid	0,9937781
Steve_Kacamata.jpeg	Fahmi	0,5940334
Steve_Netral.jpeg	Steve	0,9516179
Steve_PoseKanan.jpeg	Steve	0,9770718
Steve_PoseKiri.jpeg	Steve	0,96466863
Steve_Sedih.jpeg	Steve	0,98090017
Steve_Senyum.jpeg	Steve	0,9363021
Sugiarto_Kacamata.jpeg	Fasma	0,45277986
Sugiarto_Netral.jpeg	Sugiarto	0,39635786
Sugiarto_PoseKanan.jpeg	Sugiarto	0,91640437
Sugiarto_PoseKiri.jpeg	Sugiarto	0,3554566
Sugiarto_Sedih.jpeg	Fahmi	0,48919943
Sugiarto_Senyum.jpeg	Sugiarto	0,37418213
Yuda_Kacamata.jpeg	Yuda	0,47977787
Yuda_Netral.jpeg	Yuda	0,9908112
Yuda_PoseKanan.jpeg	Yuda	0,97280586
Yuda_PoseKiri.jpeg	Yuda	0,9489837
Yuda_Sedih.jpeg	Yuda	0,94143766
Yuda_Senyum.jpeg	Yuda	0,90213144

❖ Hasil Klasifikasi Pengenalan Wajah Skenario B4 (Skor adalah nilai output prediksi tertinggi dari model *classifier*)

Data uji	Hasil Prediksi	Skor
Aldinata_Kacamata.jpeg	Aldinata	0,8104978
Aldinata_Netral.jpeg	Aldinata	0,9387545
Aldinata_PoseKanan.jpeg	Aldinata	0,8004558
Aldinata_PoseKiri.jpeg	Sugiarto	0,36279157
Aldinata_Sedih.jpeg	Aldinata	0,41804722
Aldinata_Senyum.jpeg	Fahmi	0,3962862
Ardo_Kacamata.jpeg	Ardo	0,6388736
Ardo_Netral.jpeg	Steve	0,32948223
Ardo_PoseKanan.jpeg	Handika	0,3971082
Ardo_PoseKiri.jpeg	Ardo	0,5040799
Ardo_Sedih.jpeg	Handika	0,39527926
Ardo_Senyum.jpeg	Fahmi	0,27544475
Fahmi_Kacamata.jpeg	Fasma	0,5054695
Fahmi_Netral.jpeg	Fahmi	0,83436084
Fahmi_PoseKanan.jpeg	Fasma	0,68598497
Fahmi_PoseKiri.jpeg	Fahmi	0,589387
Fahmi_Sedih.jpeg	Steve	0,5383941
Fahmi_Senyum.jpeg	Fahmi	0,6653342
Fasma_Kacamata.jpeg	Fasma	0,744549
Fasma_Netral.jpeg	Fasma	0,98685676
Fasma_PoseKanan.jpeg	Fasma	0,9848909
Fasma_PoseKiri.jpeg	Fasma	0,98368394
Fasma_Sedih.jpeg	Fasma	0,97537464
Fasma_Senyum.jpeg	Fasma	0,99258256
Handika_Kacamata.jpeg	Steve	0,6648923
Handika_Netral.jpeg	Handika	0,6690174
Handika_PoseKanan.jpeg	Handika	0,65212345

Data uji	Hasil Prediksi	Skor
Handika_PoseKiri.jpeg	Handika	0,26996186
Handika_Sedih.jpeg	Handika	0,37796128
Handika_Senyum.jpeg	Handika	0,49811375
Rasyid_Kacamata.jpeg	Rasyid	0,9677693
Rasyid_Netral.jpeg	Rasyid	0,99321634
Rasyid_PoseKanan.jpeg	Rasyid	0,986961
Rasyid_PoseKiri.jpeg	Rasyid	0,98030406
Rasyid_Sedih.jpeg	Rasyid	0,9352152
Rasyid_Senyum.jpeg	Rasyid	0,9907603
Steve_Kacamata.jpeg	Handika	0,49834463
Steve_Netral.jpeg	Steve	0,9210205
Steve_PoseKanan.jpeg	Steve	0,7178988
Steve_PoseKiri.jpeg	Steve	0,94106674
Steve_Sedih.jpeg	Steve	0,93845385
Steve_Senyum.jpeg	Steve	0,79397255
Sugiarto_Kacamata.jpeg	Handika	0,6747912
Sugiarto_Netral.jpeg	Sugiarto	0,44576436
Sugiarto_PoseKanan.jpeg	Sugiarto	0,77687544
Sugiarto_PoseKiri.jpeg	Sugiarto	0,6812993
Sugiarto_Sedih.jpeg	Sugiarto	0,426543
Sugiarto_Senyum.jpeg	Sugiarto	0,4827858
Yuda_Kacamata.jpeg	Yuda	0,91846055
Yuda_Netral.jpeg	Yuda	0,988181
Yuda_PoseKanan.jpeg	Yuda	0,9414395
Yuda_PoseKiri.jpeg	Yuda	0,9713082
Yuda_Sedih.jpeg	Yuda	0,81123877
Yuda_Senyum.jpeg	Yuda	0,48992977

❖ Hasil Klasifikasi Pengenalan Wajah Skenario B5 (Skor adalah nilai output prediksi tertinggi dari model *classifier*)

Data uji	Hasil Prediksi	Skor
Aldinata_Kacamata.jpeg	Aldinata	0,7950637
Aldinata_Netral.jpeg	Aldinata	0,9192038
Aldinata_PoseKanan.jpeg	Aldinata	0,8953622
Aldinata_PoseKiri.jpeg	Steve	0,38646472
Aldinata_Sedih.jpeg	Aldinata	0,4849425
Aldinata_Senyum.jpeg	Aldinata	0,8626001
Ardo_Kacamata.jpeg	Ardo	0,6611331
Ardo_Netral.jpeg	Ardo	0,86462224
Ardo_PoseKanan.jpeg	Ardo	0,52930754
Ardo_PoseKiri.jpeg	Ardo	0,66378605
Ardo_Sedih.jpeg	Ardo	0,93809694
Ardo_Senyum.jpeg	Ardo	0,8725074
Fahmi_Kacamata.jpeg	Fahmi	0,8763636
Fahmi_Netral.jpeg	Fahmi	0,99060047
Fahmi_PoseKanan.jpeg	Fahmi	0,66572946
Fahmi_PoseKiri.jpeg	Steve	0,46743655
Fahmi_Sedih.jpeg	Fahmi	0,45765194
Fahmi_Senyum.jpeg	Fahmi	0,8601754
Fasma_Kacamata.jpeg	Rasyid	0,37362665
Fasma_Netral.jpeg	Fasma	0,97775686
Fasma_PoseKanan.jpeg	Fasma	0,9935801
Fasma_PoseKiri.jpeg	Fasma	0,9911787
Fasma_Sedih.jpeg	Fasma	0,9895058
Fasma_Senyum.jpeg	Fasma	0,98959374
Handika_Kacamata.jpeg	Steve	0,41899014
Handika_Netral.jpeg	Handika	0,7908059
Handika_PoseKanan.jpeg	Handika	0,7041437

Data uji	Hasil Prediksi	Skor
Handika_PoseKiri.jpeg	Sugiarto	0,39145014
Handika_Sedih.jpeg	Ardo	0,38353828
Handika_Senyum.jpeg	Handika	0,37024945
Rasyid_Kacamata.jpeg	Rasyid	0,84920293
Rasyid_Netral.jpeg	Rasyid	0,9923907
Rasyid_PoseKanan.jpeg	Rasyid	0,994962
Rasyid_PoseKiri.jpeg	Rasyid	0,8695119
Rasyid_Sedih.jpeg	Rasyid	0,7281988
Rasyid_Senyum.jpeg	Rasyid	0,9946221
Steve_Kacamata.jpeg	Steve	0,6118413
Steve_Netral.jpeg	Steve	0,91013175
Steve_PoseKanan.jpeg	Steve	0,9863326
Steve_PoseKiri.jpeg	Steve	0,9217416
Steve_Sedih.jpeg	Steve	0,9795829
Steve_Senyum.jpeg	Steve	0,9814217
Sugiarto_Kacamata.jpeg	Fahmi	0,59960836
Sugiarto_Netral.jpeg	Sugiarto	0,57373077
Sugiarto_PoseKanan.jpeg	Sugiarto	0,7791218
Sugiarto_PoseKiri.jpeg	Sugiarto	0,48312318
Sugiarto_Sedih.jpeg	Steve	0,44552535
Sugiarto_Senyum.jpeg	Sugiarto	0,36035946
Yuda_Kacamata.jpeg	Yuda	0,5930142
Yuda_Netral.jpeg	Yuda	0,9845462
Yuda_PoseKanan.jpeg	Yuda	0,92899
Yuda_PoseKiri.jpeg	Yuda	0,9663373
Yuda_Sedih.jpeg	Yuda	0,84515214
Yuda_Senyum.jpeg	Yuda	0,5241513

❖ Hasil Klasifikasi Pengenalan Wajah Skenario B6 (Skor adalah nilai output prediksi tertinggi dari model *classifier*)

Data uji	Hasil Prediksi	Skor
Aldinata_Kacamata.jpeg	Aldinata	0,69615006
Aldinata_Netral.jpeg	Aldinata	0,9774539
Aldinata_PoseKanan.jpeg	Aldinata	0,86993045
Aldinata_PoseKiri.jpeg	Aldinata	0,40973344
Aldinata_Sedih.jpeg	Aldinata	0,8680885
Aldinata_Senyum.jpeg	Aldinata	0,89334804
Ardo_Kacamata.jpeg	Ardo	0,8594924
Ardo_Netral.jpeg	Ardo	0,4138689
Ardo_PoseKanan.jpeg	Ardo	0,4208289
Ardo_PoseKiri.jpeg	Ardo	0,5469048
Ardo_Sedih.jpeg	Ardo	0,79203063
Ardo_Senyum.jpeg	Ardo	0,3775748
Fahmi_Kacamata.jpeg	Fahmi	0,8712558
Fahmi_Netral.jpeg	Fahmi	0,9146147
Fahmi_PoseKanan.jpeg	Fasma	0,24305034
Fahmi_PoseKiri.jpeg	Fahmi	0,86919516
Fahmi_Sedih.jpeg	Fasma	0,3406769
Fahmi_Senyum.jpeg	Fahmi	0,39970747
Fasma_Kacamata.jpeg	Fasma	0,5291822
Fasma_Netral.jpeg	Fasma	0,9626526
Fasma_PoseKanan.jpeg	Fasma	0,9869479
Fasma_PoseKiri.jpeg	Fasma	0,9836753
Fasma_Sedih.jpeg	Fasma	0,9764994
Fasma_Senyum.jpeg	Fasma	0,9872751
Handika_Kacamata.jpeg	Steve	0,43614438
Handika_Netral.jpeg	Handika	0,74358714
Handika_PoseKanan.jpeg	Handika	0,70813936

Data uji	Hasil Prediksi	Skor
Handika_PoseKiri.jpeg	Handika	0,6617592
Handika_Sedih.jpeg	Handika	0,48531967
Handika_Senyum.jpeg	Handika	0,76157016
Rasyid_Kacamata.jpeg	Rasyid	0,9245258
Rasyid_Netral.jpeg	Rasyid	0,994735
Rasyid_PoseKanan.jpeg	Rasyid	0,9969103
Rasyid_PoseKiri.jpeg	Rasyid	0,95763636
Rasyid_Sedih.jpeg	Rasyid	0,6514609
Rasyid_Senyum.jpeg	Rasyid	0,99579924
Steve_Kacamata.jpeg	Steve	0,2892434
Steve_Netral.jpeg	Steve	0,9854881
Steve_PoseKanan.jpeg	Steve	0,99149865
Steve_PoseKiri.jpeg	Steve	0,96502244
Steve_Sedih.jpeg	Steve	0,9855364
Steve_Senyum.jpeg	Steve	0,9891075
Sugiarto_Kacamata.jpeg	Handika	0,43457964
Sugiarto_Netral.jpeg	Sugiarto	0,36387694
Sugiarto_PoseKanan.jpeg	Sugiarto	0,6861547
Sugiarto_PoseKiri.jpeg	Sugiarto	0,3547034
Sugiarto_Sedih.jpeg	Steve	0,52907306
Sugiarto_Senyum.jpeg	Handika	0,555695
Yuda_Kacamata.jpeg	Yuda	0,8784694
Yuda_Netral.jpeg	Yuda	0,99242616
Yuda_PoseKanan.jpeg	Yuda	0,9077744
Yuda_PoseKiri.jpeg	Yuda	0,9872344
Yuda_Sedih.jpeg	Yuda	0,9626526
Yuda_Senyum.jpeg	Yuda	0,75522995

BIODATA PENULIS



Aldinata Rizky Revanda, lahir di Ponorogo pada tanggal 26 Juni 1998. Penulis menempuh pendidikan mulai dari TK Al-Amanah (2002-2004), kemudian SD Muhammadiyah Ponorogo (2004-2010), SMP Negeri 1 Ponorogo (2010-2013), SMA Negeri 1 Ponorogo (2013-2016), dan sekarang sedang menjalani pendidikan S1 Teknik Informatika di ITS. Penulis aktif dalam beberapa organisasi, antara lain adalah Staf Departemen Media dan Informasi Himpunan Mahasiswa Teknik

Computer-Informatika (HMTIC) ITS 2017-2018, Staf Departemen Ukhuwah Keluarga Muslim Informatika (KMI) HMTIC ITS 2017-2018, Staf Ahli Departemen Kaderisasi KMI-HMTIC ITS 2019, Staf Departemen Dalam Negeri Badan Eksekutif Mahasiswa Fakultas Teknologi Informasi dan Komunikasi (BEM FTIK) ITS 2017-2018, Ketua BEM FTIK ITS 2019. Penulis juga aktif dalam beberapa kepanitiaan, diantaranya adalah Koordinator Seminar Keilmuan Parade KMI 2018, Wakil Koordinator II National Logic Competition (NLC) Schematics 2017, Koordinator NLC Schematics 2018, dan Tim Kawal GemasTIK XII ITS 2019. Selain itu penulis juga aktif di kegiatan lain, yaitu Pemandu Latihan Keterampilan Manajemen Mahasiswa (LKMM) FTIK, Forum Daerah Ponorogo ITS, Keluarga Mahasiswa Pelajar Ponorogo-Surabaya, Tim Bayucaraka Kompetisi Robot Terbang Indonesia (KRTI) UKM Robotik ITS, dan Admin Laboratorium Komputasi Cerdas dan Visi (KCV) Teknik Informatika ITS. Komunikasi dengan penulis dapat melalui telepon: +6285790337273 dan *email*: **aldinatarizky@gmail.com**.