



TUGAS AKHIR - EC184801

**SISTEM Pendeteksi Kondisi KESEHATAN BERBASIS
CITRA WAJAH MENGGUNAKAN *DEEP
CONVOLUTIONAL NEURAL NETWORK***

Naufal Reyhan Fadhil
NRP 07211540000001

Dosen Pembimbing
Dr. I Ketut Eddy Purnama, ST., MT.
Dr. Reza Fuad Rachmadi, ST., MT.

DEPARTEMEN TEKNIK KOMPUTER
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2019



TUGAS AKHIR - EC184801

**SISTEM PENDETEKSI KONDISI KESEHATAN BERBASIS
CITRA WAJAH MENGGUNAKAN *DEEP
CONVOLUTIONAL NEURAL NETWORK***

Naufal Reyhan Fadhil
NRP 07211540000001

Dosen Pembimbing
Dr. I Ketut Eddy Purnama, ST., MT.
Dr. Reza Fuad Rachmadi, ST., MT.

DEPARTEMEN TEKNIK KOMPUTER
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2019



FINAL PROJECT - EC184801

**FACE RECOGNITION FOR HEALTH CONDITION
DETECTION SYSTEM USING DEEP CONVOLUTIONAL
NEURAL NETWORK**

Naufal Reyhan Fadhil
NRP 0721154000001

Advisor
Dr. I Ketut Eddy Purnama, ST., MT.
Dr. Reza Fuad Rachmadi, ST., MT.

Department of Computer Engineering
Faculty of Electrical Technology
Sepuluh Nopember Institute of Technology
Surabaya 2019

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "**Sistem Pendekripsi Kondisi Kesehatan Seseorang Berbasis Citra Wajah Menggunakan Deep Convolutional Neural Network**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 15 Juli 2019



Naufal Reyhan Fadhil

NRP. 07211540000001

LEMBAR PENGESAHAN

Sistem Pendekksi Kondisi Kesehatan Berbasis Citra Wajah menggunakan Deep Convolutional Neural Network

Tugas Akhir ini disusun untuk memenuhi salah satu syarat memperoleh gelar Sarjana Teknik di Institut Teknologi Sepuluh Nopember Surabaya

Oleh : Naufal Reyhan Fadhil (NRP : 07211540000001)

Tanggal Ujian : 20 Juni 2019

Periode Wisuda : September 2019

Disetujui oleh :

Dr. I Ketut Eddy Purnama, ST., MT.
NIP. 196907301995121001

(Pembimbing I)

(Pembimbing II)

Dr. Reza Fuad Rachmadi, ST., MT.
NIP. 198504032012121001

(Penguji I)

(Penguji II)

Susi Juniaستuti, ST., M.Eng.
NIP. 196506181999032001

Prof. Dr. Ir. Yoyon K. Suprapto, M.Sc.
NIP. 195409251978031001



Mengetahui

Kepala Departemen Teknik Komputer

Dr. I Ketut Eddy Purnama, ST., MT.
NIP. 196907301995121001

ABSTRAK

- Nama Mahasiswa : Naufal Reyhan Fadhil
Judul Tugas Akhir : Sistem Pendekripsi Kondisi Kesehatan Seseorang Berbasis Citra Wajah Menggunakan *Deep Convolutional Neural Network*
Pembimbing : 1. Dr. I Ketut Eddy Purnama, ST., MT.
 2. Dr. Reza Fuad Rachmadi, ST., MT.

Kesehatan adalah keadaan sehat, baik secara fisik, mental, spiritual maupun sosial yang memungkinkan setiap orang untuk hidup produktif secara sosial dan ekonomis. Kesehatan manusia merupakan hal yang sangat penting untuk selalu diperhatikan, terutama ketika seseorang tersebut mulai mengalami gejala penyakit. Aktivitas manusia yang sangat kompleks dikarenakan tuntutan pekerjaan dapat menyebabkan kondisi kesehatan seseorang menjadi menurun, apalagi jika tidak menjaga pola hidup sehat. Salah satu poin penting dalam menjaga pola hidup sehat adalah dengan mengetahui kondisi kesehatan diri sendiri. Untuk itu pada tugas akhir ini dikembangkan sebuah sistem yang berfungsi untuk dapat mengenali kondisi kesehatan pada seseorang dengan melakukan pendekripsi pada wajah. Metode yang digunakan pada sistem ini adalah dengan menggunakan *Deep Convolutional Neural Network Classifier*. Sistem ini akan melakukan proses *scanning* pada wajah, kemudian hasilnya akan diklasifikasi menjadi dua kategori, yaitu sehat atau sakit. Berdasarkan hasil percobaan dengan menggunakan 11 model *CNN* yang berbeda, didapatkan hasil yaitu model *ShuffleNet V2* memiliki tingkat akurasi yang tinggi yaitu 94 % dengan waktu evaluasi yang cepat, sebesar 2,068 detik, sehingga model ini dipilih dan diimplementasikan pada aplikasi berbasis *Android*. Penggunaan dari sistem ini diharapkan dapat memantau kondisi kesehatan seseorang secara berkala untuk menjaga pola hidup sehat.

Kata Kunci : Kesehatan, Deteksi Wajah, *Convolutional Neural Network*

Halaman ini sengaja dikosongkan

ABSTRACT

*Name : Naufal Reyhan Fadhil
Title : Face Recognition for Health Condition Detection System using Deep Convolutional Neural Networks
Advisors : 1. Dr. I Ketut Eddy Purnama, ST., MT.
2. Dr. Reza Fuad Rachmadi, ST., MT.*

Health is a healthy condition, both physically, mentally, spiritually and socially, which enables everyone to live productively socially and economically. Human health is a very important thing to always pay attention to, especially when a person begins to experience symptoms of the disease. Human activities are very complex because the demands of work can cause a person's health conditions to decrease, especially if we do not maintain a healthy lifestyle. One important point in maintaining a healthy lifestyle is to know the condition of one's own health. For this reason, a final system was developed in this final project that serves to be able to recognize a person's health conditions by performing facial detection. The method used in this system is to use the Deep Convolutional Neural Network Classifier. This system will do the scanning process on the face, then the results will be classified into two categories, namely healthy or sick. Based on the results of the experiment using 11 different CNN models, the results are that the ShuffleNetV2 model has a high level of accuracy of 94 % with a fast evaluation time of 2.068 seconds, so this model is selected and implemented on Android-based applications. The use of this system is expected to be able to monitor one's health condition regularly to maintain a healthy lifestyle.

Keywords : Health, Face Recognition, Convolutional Neural Network

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji dan syukur kehadirat Allah SWT atas segala limpahan berkah, rahmat, serta ridho-Nya, penulis dapat menyelesaikan penelitian ini dengan judul **Sistem Pendekripsi Kondisi Kesehatan Seseorang Berbasis Citra Wajah Menggunakan Deep Convolutional Neural Networks**.

Penelitian ini disusun dalam rangka pemenuhan bidang riset di Departemen Teknik Komputer ITS, serta digunakan sebagai persyaratan menyelesaikan pendidikan Sarjana. Penelitian ini dapat terselesaikan tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada :

1. Keluarga, Ibu, Bapak, dan Saudara tercinta yang telah memberikan dorongan baik spiritual dan material dalam penyelesaian buku penelitian ini.
2. Bapak Dr. I Ketut Eddy Purnama, ST., MT. selaku Kepala Departemen Teknik Komputer ITS dan juga sebagai dosen pembimbing I.
3. Bapak Dr. Reza Fuad Rachmadi, ST., MT. selaku dosen pembimbing II yang selalu memberikan arahan dan dukungan selama mengerjakan penelitian tugas akhir ini.
4. Bapak-ibu dosen pengajar Departemen Teknik Komputer ITS, atas pengajaran, bimbingan serta perhatian yang diberikan kepada penulis selama ini.
5. Seluruh Staf Departemen Teknik Komputer ITS yang telah membantu penulis dalam hal administrasi.
6. Saudari Tiara Shafira yang senantiasa memberikan dukungan dan motivasi dalam pelaksanaan penelitian tugas akhir ini.
7. Seluruh teman-teman dari angkatan e55, Asisten Laboraturium B201 dan B401 Departemen Teknik Komputer ITS, dan keluarga minat bakat ITS.

Surabaya, 15 Juli 2019

Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

ABSTRAK	i
ABSTRACT	iii
KATA PENGANTAR	v
DAFTAR ISI	vii
DAFTAR GAMBAR	xi
DAFTAR TABEL	xv
NOMENKLATUR	xvii
1 PENDAHULUAN	1
1.1 Latar belakang	1
1.2 Permasalahan	2
1.3 Tujuan	2
1.4 Batasan masalah	3
1.5 Sistematika Penulisan	3
2 TINJAUAN PUSTAKA	5
2.1 Artificial Intelligence(AI)	5
2.2 Machine Learning	6
2.3 Deep Learning	8
2.4 Artificial Neural Network	10
2.5 Feedforward Neural Network	12
2.6 Multilayer Perceptron	13
2.6.1 Activation Function	13
2.6.2 Layers	14
2.6.3 Learning	14
2.7 Convolutional Neural Network	15
2.7.1 Convolutional Layer	16
2.7.2 Pooling Layer	19
2.7.3 ReLU (Rectified Linear Unit) Activation Function	20

2.7.4	Softmax Classifier	21
2.8	Image Classification	22
2.9	TensorFlow	22
2.9.1	TensorBoard	23
2.9.2	TensorFlow Lite	24
2.10	Inception-v3	25
2.11	Inception-v4	26
2.12	Inception-ResNet-v2	26
2.13	ResNet-v2-152	27
2.14	MobileNetV1	28
2.15	MobileNetV2	29
2.16	MobileFaceNets	30
2.17	SqueezeNet	30
2.18	NasNet Mobile	31
2.19	DenseNet	32
2.20	ShuffleNetV2	33
2.21	Wajah	34
2.21.1	Simetris Wajah	35
2.21.2	Adipositas Wajah	36
2.21.3	Kondisi Kulit (Tekstur dan Warna)	37
2.21.4	Ekspresi Wajah	38
3	DESAIN DAN IMPLEMENTASI SISTEM	41
3.1	Desain Sistem	41
3.2	Collecting Image Datasets	41
3.3	Data Splitting	43
3.4	Pre-processing	45
3.5	Building a CNN Model	50
3.6	Training Process	53
3.7	Validation Process	55
3.8	Testing Process	57
3.9	Deploying Model to Android	59
4	PENGUJIAN DAN ANALISA	61
4.1	Pengujian Jenis Model	61
4.1.1	Inception-v3	61
4.1.2	Inception-v4	62
4.1.3	Inception-ResNet-v2	62

4.1.4	ResNet-v2-152	62
4.1.5	MobileNetV1	63
4.1.6	MobileNetV2	63
4.1.7	MobileFaceNets	64
4.1.8	SqueezeNet	64
4.1.9	NasNet Mobile	65
4.1.10	DenseNet	65
4.1.11	ShuffleNetV2	66
4.1.12	Perbandingan Hasil Prediksi	68
4.2	Pengujian Performa Aplikasi	80
4.2.1	Pengujian Tanpa Menggunakan <i>Face Detection</i>	80
4.2.2	Pengujian Menggunakan <i>Face Detection</i>	83
4.2.3	Perbandingan Performa Aplikasi	86
4.3	Pengujian Pada Orang Sakit	86
4.3.1	Feature Maps Dataset Orang Sakit	87
4.3.2	Pengujian Pada Orang Sakit	89
5	PENUTUP	91
5.1	Kesimpulan	91
5.2	Saran	91
DAFTAR PUSTAKA		93
Biografi Penulis		101

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

2.1	Ilustrasi <i>Artificial Intelligence</i> [1].	6
2.2	Kategori dari <i>Machine Learning</i> [2].	7
2.3	Perbandingan dari <i>Artificial Intelligence</i> , <i>Machine Learning</i> , dan <i>Deep Learning</i> [3].	9
2.4	Perbandingan <i>biological neuron</i> dan <i>artificial neuron</i> [4].	10
2.5	<i>Artificial Neural Network model</i> [5].	12
2.6	<i>Feedforward Neural Network</i> dengan 1 <i>hidden layer</i> dan 3 <i>neurons</i> [6].	12
2.7	<i>Convolutional Layer</i> [7].	16
2.8	<i>Max Pooling</i> dengan 2×2 filter dan stride = 2[7]. .	20
2.9	<i>Fungsi Aktivasi ReLU</i>	20
2.10	Ilustrasi pada <i>Softmax Layer</i> pada <i>neural network</i> [8].	21
2.11	Contoh dari <i>Image Classification</i> beserta <i>Localization</i> dan <i>Object Detection</i> [9]	22
2.12	<i>Workflow</i> pada <i>TensorFlow</i> [10].	23
2.13	<i>Dashboard</i> pada <i>TensorBoard</i> [10].	24
2.14	<i>Workflow</i> pada <i>TensorFlow Lite</i> [10].	24
2.15	Arsitektur dari <i>Inception-v3</i> [11].	25
2.16	Skema arsitektur dari <i>Inception-v4</i> [12].	26
2.17	Arsitektur dari <i>Inception-ResNet-v2</i> [12].	27
2.18	Arsitektur dari <i>ResNet-v2</i> [13].	27
2.19	Perbandingan dari <i>ResNet-v1</i> dengan <i>ResNet-v2</i> . .	28
2.20	<i>MobileNet</i> dan beberapa contoh aplikasinya[14]. . .	28
2.21	Arsitektur dari <i>MobileNetV2</i> [15].	29
2.22	Arsitektur dari <i>MobileFaceNets</i> [16].	30
2.23	Arsitektur dari <i>SqueezeNet</i> [17].	31
2.24	Arsitektur dari <i>NasNet</i> [18].	32
2.25	Arsitektur dari <i>DenseNet</i> [19].	32
2.26	Perbandingan <i>run time</i> <i>ShuffleNet</i> dengan <i>MobileNet</i>	33
2.27	Arsitektur dari <i>ShuffleNet</i> [20].	34
2.28	Arsitektur dari <i>ShuffleNetV2</i> [21].	34
3.1	Bagan Umum Metodologi Sistem	41
3.2	Diagram alir pada proses <i>collecting image datasets</i> .	43
3.3	Visualisasi dari <i>data splitting</i>	43

3.4	Diagram alir proses <i>data splitting</i>	45
3.5	Diagram alir proses <i>pre-processing</i>	46
3.6	Proses <i>cropping</i> pada gambar	47
3.7	Proses <i>image scaling</i> pada gambar	47
3.8	<i>Mean</i> (kiri) dan <i>Standard Deviation</i> (kanan)	48
3.9	Ilustrasi pada proses <i>data augmentation</i>	49
3.10	Diagram alir proses <i>building a CNN Model</i>	50
3.11	Arsitektur model untuk <i>standard neural network</i> , dengan warna hijau menunjukkan proses <i>training</i> untuk semua bobot dan bias.	51
3.12	Arsitektur model untuk <i>transfer-learning neural network model</i> , dengan warna merah menunjukkan bobot dan bias yang tetap, sedangkan warna hijau menunjukkan proses <i>training</i> pada bobot dan bias di <i>final layer</i>	52
3.13	Diagram alir <i>training process</i>	53
3.14	Diagram alir <i>validation process</i>	55
3.15	Visualisasi dari <i>Train/Test split</i> dan <i>Cross Validation</i>	56
3.16	Diagram alir <i>testing process</i>	57
3.17	Tahapan melakukan prediksi dari <i>test set</i>	58
3.18	Diagram alir proses <i>deploying model to Android</i>	59
3.19	Arsitektur <i>TensorFlow Lite</i>	60
4.1	Grafik Perbandingan Akurasi dengan <i>File Size</i>	67
4.2	Hasil uji dari model <i>Inception-v3</i>	69
4.3	Hasil uji dari model <i>Inception-v4</i>	70
4.4	Hasil uji dari model <i>Inception-ResNet-v2</i>	71
4.5	Hasil uji dari model <i>ResNet-v2-152</i>	72
4.6	Hasil uji dari model <i>MobileNetV1</i>	73
4.7	Hasil uji dari model <i>MobileNetV2</i>	73
4.8	Hasil uji dari model <i>MobileFaceNets</i>	74
4.9	Hasil uji dari model <i>SqueezeNet</i>	75
4.10	Hasil uji dari model <i>NasNet Mobile</i>	76
4.11	Hasil uji dari model <i>DenseNet</i>	77
4.12	Hasil uji dari model <i>ShuffleNetV2</i>	78
4.13	Hasil pengujian pada aplikasi <i>Android</i>	80
4.14	Hasil pengujian performa <i>CPU Usage</i> tanpa menggunakan <i>face detection</i>	81

4.15	Hasil pengujian performa <i>Memory Usage</i> tanpa menggunakan <i>face detection</i>	82
4.16	Hasil pengujian performa tanpa menggunakan <i>face detection</i>	82
4.17	Tampilan aplikasi ketika wajah tidak terdeteksi	83
4.18	Tampilan aplikasi ketika wajah terdeteksi	83
4.19	Hasil pengujian performa <i>CPU Usage</i> dengan menggunakan <i>face detection</i>	84
4.20	Hasil pengujian performa <i>Memory Usage</i> dengan menggunakan <i>face detection</i>	85
4.21	Hasil pengujian performa dengan menggunakan <i>face detection</i>	85
4.22	<i>Feature maps</i> citra wajah orang sakit pada Stage 1 dengan menggunakan arsitektur <i>ShuffleNetV2</i>	87
4.23	<i>Feature maps</i> citra wajah orang sakit pada Stage 2 dengan menggunakan arsitektur <i>ShuffleNetV2</i>	88
4.24	<i>Feature maps</i> citra wajah orang sakit pada Stage 3 dengan menggunakan arsitektur <i>ShuffleNetV2</i>	88
4.25	<i>Feature maps</i> citra wajah orang sakit pada Stage 4 dengan menggunakan arsitektur <i>ShuffleNetV2</i>	89
4.26	Hasil pengujian terhadap tiga orang sakit	89
4.27	Hasil pengujian terhadap dua orang sakit	90

Halaman ini sengaja dikosongkan

DAFTAR TABEL

4.1	Tabel spesifikasi dari model <i>Inception-v3</i>	61
4.2	Tabel spesifikasi dari model <i>Inception-v4</i>	62
4.3	Tabel spesifikasi dari model <i>Inception-ResNet-v2</i> . .	62
4.4	Tabel spesifikasi dari model <i>ResNet-v2-152</i>	63
4.5	Tabel spesifikasi dari model <i>MobileNetV1 1.4 224</i> . .	63
4.6	Tabel spesifikasi dari model <i>MobileNetV2 1.4 224</i> . .	64
4.7	Tabel spesifikasi dari model <i>MobileFaceNets</i>	64
4.8	Tabel spesifikasi dari model <i>SqueezeNet</i>	65
4.9	Tabel spesifikasi dari model <i>NasNet Mobile</i>	65
4.10	Tabel spesifikasi dari model <i>DenseNet</i>	66
4.11	Tabel spesifikasi dari model <i>ShuffleNetV2</i>	66
4.12	Tabel perbandingan dari berbagai model yang digunakan	67
4.13	Tabel perbandingan evaluation time, <i>Top-1 Accuracy</i> , dan testing results dari berbagai model yang digunakan	79
4.14	Tabel spesifikasi Laptop yang digunakan	80
4.15	Tabel perbandingan performa aplikasi dengan menggunakan <i>face detection</i> dan tanpa <i>face detection</i>	86

Halaman ini sengaja dikosongkan

NOMENKLATUR

X	:	<i>input layer</i>
Y	:	<i>output layer</i>
K	:	Fungsi aktivasi
w	:	<i>weight</i>
b	:	<i>bias</i>
$y(v_i)$:	fungsi <i>sigmoid</i>
$e_j(n)$:	tingkat <i>error</i> dalam <i>output node</i> pada titik data n th
$\mathcal{E}(n)$:	fungi untuk meminimalisir <i>error</i> pada <i>output</i>
$\Delta w_{ji}(n)$:	perubahan pada setiap <i>weight</i>
y_i	:	<i>output</i> dari neuron sebelumnya
η	:	<i>learning rate</i>
ϕ'	:	turunan fungsi aktivasi

Halaman ini sengaja dikosongkan

BAB 1

PENDAHULUAN

Penelitian ini di latar belakangi oleh berbagai kondisi yang menjadi acuan. Selain itu juga terdapat beberapa permasalahan yang akan dijawab sebagai luaran dari penelitian.

1.1 Latar belakang

Teknologi *Artificial Intelligence* (*AI*) atau kecerdasan buatan memiliki potensi yang tinggi untuk membantu dalam bidang kesehatan [22]. Berkat revolusi digital, pengembangan *Artificial Intelligence* di bidang kesehatan pun terlahir, para profesional medis tidak perlu lagi menghafal banyak hal termasuk istilah-istilah dalam bidang kesehatan. Teknologi digital telah membebaskan dokter, perawat dan peneliti untuk lebih memfokuskan energi dan mental pada tugas-tugas kognitif tingkat tinggi dan perawatan pasien. Kecerdasan buatan akan membawa dunia medis ke tingkat selanjutnya.

Kesehatan adalah keadaan sehat, baik secara fisik, mental, spiritual maupun sosial yang memungkinkan setiap orang untuk hidup produktif secara sosial dan ekonomis. Menjaga kesehatan tubuh sangatlah penting, karena akan menentukan sehat atau tidaknya tubuh. Sehat merupakan kondisi normal seseorang yang merupakan hak hidupnya. Sehat berhubungan dengan hukum alam yang mengatur tubuh, jiwa, lingkungan yang berupa udara segar, kebiasaan, dan gaya hidup yang baik. Dalam hidup yang sehat maka tidak terlepas dari adanya faktor-faktor pendukung yang dapat memenuhi kebutuhan tubuh, diantaranya dengan mengolah tubuh dengan melakukan aktivitas seperti olahraga. Namun, aktivitas manusia yang sangat kompleks dikarenakan tuntutan pekerjaan dapat menyebabkan kondisi kesehatan seseorang menjadi menurun, apalagi jika tidak dijaga dengan pola hidup yang sehat. Salah satu poin penting dalam menjaga pola hidup sehat adalah dengan mengetahui kondisi kesehatan diri sendiri sejak dulu.

Kondisi kesehatan seseorang dapat dilihat dari *Vital Signs*, *Voice Analysis*, atau *Voice Recognition*. *Vital Signs* atau tanda-tanda vital adalah ukuran statistik berbagai fisiologis yang digunakan un-

tuk membantu menentukan status atau kondisi kesehatan seseorang, terutama pada paises yang secara medis tidak stabil atau memiliki faktor-faktor resiko komplikasi kardiopulmonal dan untuk menilai respons terhadap intervensi. Tanda vital juga berguna untuk menentukan dosis yang adekuat bagi tindakan fisioterapi, khususnya *exercise*. *Vital Signs* terdiri atas tekanan darah, denyut nadi, suhu tubuh, dan laju pernapasan. *Voice Analysis* dapat digunakan untuk mengenali kondisi kesehatan seseorang terutama yang berhubungan dengan kondisi paru-paru. Penerapan *Face Recognition* juga dapat digunakan untuk memantau gejala atau kondisi kesehatan seseorang.

Dalam tugas akhir ini akan dirancang sebuah sistem yang mampu mengenali kondisi kesehatan seseorang dengan melakukan pendektsian pada kondisi wajah. Setelah didapatkan citra wajah, kemudian akan dilakukan proses klasifikasi kondisi seseorang tersebut, apakah sedang dalam kondisi sehat, atau mengalami gejala penyakit. Dalam sistem ini, metode yang digunakan untuk mendeksi gejala visual yang dapat diamati pada wajah menggunakan *Supervised Learning* yang menggunakan berbagai data dari *Labeled Dataset*. Kemudian, algoritma *Deep Convolutional Neural Network* digunakan untuk melakukan proses klasifikasi citra wajah.

Dengan adanya sebuah sistem yang dirancang untuk dapat mampu mengenali kondisi kesehatan seseorang dengan melakukan pendektsian pada kondisi wajah ini, diharapkan penggunaan dari sistem ini dapat memantau kondisi kesehatan seseorang secara berkala untuk menjaga pola hidup sehat. Sehingga, orang tersebut diharapkan dapat lebih mencegah resiko terkena penyakit.

1.2 Permasalahan

Kurangnya perhatian seseorang akan pentingnya menjaga kesehatan sejak dini menjadi salah satu penyebab timbulnya berbagai macam gejala penyakit. Oleh karena itu, diperlukan sebuah sistem yang mampu mengenali dan memantau kondisi kesehatan seseorang secara berkala dengan melakukan pendektsian pada kondisi wajah.

1.3 Tujuan

Berdasarkan rumusan permasalahan di atas, tujuan dari penelitian ini adalah untuk mengembangkan sistem yang dapat menge-

nali dan memantau kondisi kesehatan seseorang secara berkala dengan melakukan pendektsian pada kondisi wajah, sehingga orang tersebut dapat lebih mencegah resiko terkena penyakit dengan menerapkan pola hidup sehat. Untuk melakukan pemantauan secara berkala, pada sistem ini akan digunakan sebuah *Smartphone*.

1.4 Batasan masalah

Untuk memfokuskan permasalahan yang akan diangkat maka dilakukan pembatasan masalah. Batasan-batasan masalah tersebut di antaranya adalah :

1. Menggunakan dataset publik terkait kondisi wajah.
2. Sistem ini hanya melakukan pendektsian kondisi kesehatan seseorang hanya melalui kondisi pada wajah.
3. Menggunakan *Smartphone* berbasis *Android*.

1.5 Sistematika Penulisan

Laporan penelitian tugas akhir ini tersusun dalam sistematika dan terstruktur sehingga mudah dipahami dan dipelajari oleh pembaca maupun seseorang yang ingin melanjutkan penelitian ini. Alur sistematika penulisan laporan penelitian ini yaitu :

1. BAB I Pendahuluan

Bab ini berisi uraian tentang latar belakang permasalahan, penegasan dan alasan pemilihan judul, sistematika laporan, tujuan dan metodologi penelitian.

2. BAB II Dasar Teori

Pada bab ini berisi tentang uraian secara sistematis teori-teori yang berhubungan dengan permasalahan yang dibahas pada penelitian ini. Teori-teori ini digunakan sebagai dasar dalam penelitian, yaitu informasi terkait kondisi kesehatan seseorang, algoritma *Deep Convolutional Neural Network*, algoritma untuk melakukan *Face Recognition*, *TensorFlow API*, dan teori-teori penunjang lainnya.

3. BAB III Perancangan Sistem dan Impementasi

Bab ini berisi tentang penjelasan-penjelasan terkait eksperimen yang akan dilakukan dan langkah-langkah pengambilan data citra wajah dan proses kontrol hingga ditampilkan pada *Smartphone Android*. Guna mendukung hal tersebut, digu-

nakanlah blok diagram atau *work flow* agar sistem yang akan dibuat dapat terlihat dan mudah dibaca untuk implemtasi pada pelaksanaan tugas akhir.

4. BAB IV Pengujian dan Analisa

Bab ini menjelaskan tentang pengujian eksperimen yang dilakukan terhadap citra wajah dan proses klasifikasinya. Serta terkait tingkat akurasi keberhasilan pengujian yang dilengkapi dengan analisanya.

5. BAB V Penutup

Bab ini merupakan penutup yang berisi kesimpulan yang diambil dari penelitian dan pengujian yang telah dilakukan. Saran dan kritik yang membangun untuk pengembangan lebih lanjut juga dituliskan pada bab ini.

BAB 2

TINJAUAN PUSTAKA

Demi mendukung penelitian ini, dibutuhkan beberapa teori penunjang sebagai bahan acuan dan refensi. Dengan demikian penelitian ini menjadi lebih terarah.

2.1 Artificial Intelligence(AI)

Artificial Intelligence(AI) adalah bidang multidisiplin yang tujuannya adalah untuk melakukan otomatisasi kegiatan yang saat ini membutuhkan kecerdasan manusia[23]. Keberhasilan terbaru dalam bidang *AI* dapat dilihat dari berbagai contoh, misalnya diagnosa medis yang terkomputerisasi dan sistem yang secara otomatis menyesuaikan perangkat keras dengan kebutuhan pengguna tertentu.

Bidang permasalahan yang dibahas pada *AI* adalah persepsi, manipulasi, penalaran, komunikasi, dan pembelajaran. Persepsi berkaitan dengan membangun model dunia fisik dari input sensorik seperti *visual* dan *audio*. Manipulasi berkaitan dengan artikulasi pelengkap seperti lengan mekanik dan alat penggerak, untuk menghasilkan keadaan yang diinginkan di kehidupan nyata. Penalaran berkaitan dengan fungsi kognitif tingkat tinggi seperti perencanaan, melakukan proses diagnosis, dan mendesain. Komunikasi menangani masalah dalam memahami dan menyampaikan informasi melalui penggunaan bahasa. Pembelajaran menangani permasalahan secara otomatis dengan berdasarkan pengalaman sistem untuk meningkatkan kinerja sistem.

Banyak konsep teknis penting telah muncul dari bidang *AI* yang menyatukan bidang-bidang masalah yang beragam dan membentuk dasar dari disiplin ilmu pengetahuan. Secara umum, fungsi dari sistem *AI* didasarkan dari basis pengetahuan tentang fakta dan aturan yang menjadi ciri domain kemahiran sistem.

Artificial Intelligence dapat diklasifikasikan ke dalam tiga jenis sistem yaitu kecerdasan buatan analitis, kecerdasan buatan yang terinspirasi oleh manusia, dan kecerdasan buatan yang menyerupai manusia. Kecerdasan buatan yang bersifat analitis hanya memiliki karakteristik yang konsisten dengan kecerdasan kognitif dan menggunakan pembelajaran berdasarkan pengalaman masa lalu untuk

menginformasikan dan menghasilkan sebuah keputusan. Kecerdasan buatan yang terinspirasi oleh manusia memiliki unsur-unsur dari kecerdasan kognitif dan emosional dan mampu memahami emosi dari manusia di samping elemen kognitif, untuk dipertimbangkan dalam mengambil sebuah keputusan. Sedangkan kecerdasan buatan yang menyerupai manusia menunjukkan karakteristik semua jenis kompetensi, yaitu kecerdasan kognitif, emosional, dan sosial, serta mampu untuk mengendalikan diri dan berinteraksi dengan orang lain. Ilustrasi dari *Artificial Intelligence* ditunjukkan pada gambar 2.1.



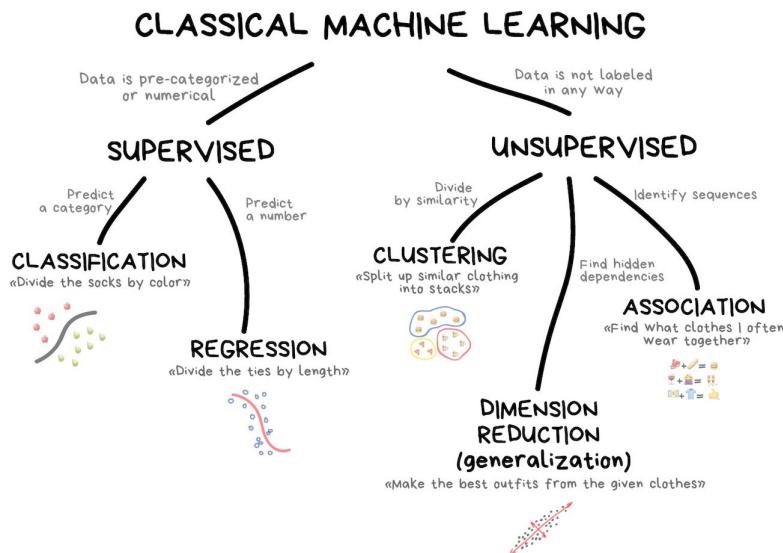
Gambar 2.1: Ilustrasi *Artificial Intelligence*[1].

2.2 Machine Learning

Machine Learning adalah cabang dari bidang *Artificial Intelligence* dan merupakan studi ilmiah tentang algoritma dan model statistik yang digunakan sistem komputer untuk secara efektif melakukan tugas tertentu tanpa menggunakan instruksi eksplisit, dengan mengandalkan pola dan inferensi sebagai penggantinya[24]. Algoritma dari *Machine Learning* membangun model matematika berdasarkan *sample data*, yang dikenal sebagai *training data*, untuk membuat prediksi atau keputusan tanpa diprogram secara eksplisit untuk melakukan tugas tersebut.

Algoritma *Machine Learning* digunakan dalam berbagai apli-

kasi, seperti *E-Mail filtering* dan *Computer Vision*, di mana tidak memungkinkan untuk mengembangkan sebuah algoritma instruksi khusus untuk melakukan tugas tersebut. *Machine Learning* sangat berkaitan erat dengan statistik komputasi, yang berfokus pada membuat prediksi menggunakan komputer. Studi tentang optimasi matematika memberikan metode, teori, dan domain aplikasi ke bidang *Machine Learning*. *Data Mining* adalah bidang studi dalam *Machine Learning* dan berfokus pada analisis data eksplorasi melalui *Unsupervised Learning*. Dalam penerapannya pada permasalahan di dunia bisnis, *Machine Learning* juga disebut sebagai analisis prediktif.



Gambar 2.2: Kategori dari *Machine Learning*[2].

Penggunaan dari *Machine Learning* diklasifikasikan ke dalam beberapa kategori besar seperti pada Gambar 2.2. Dalam *Supervised Learning*, algoritma tersebut membangun model matematika dari sekumpulan data yang berisi input dan *output* yang diinginkan. Misalnya, jika terdapat permasalahan untuk menentukan apakah suatu gambar berisi objek tertentu, *training data* untuk algoritma

supervised learning akan mencakup gambar dengan dan tanpa objek (*input*), dan setiap gambar akan memiliki label (*output*) yang menentukan apakah itu berisi objek. Dalam kasus khusus, *input* mungkin hanya sebagian tersedia, atau terbatas pada umpan balik khusus. Algoritma *Semi Supervised Learning* mengembangkan model matematika dari *training data* yang tidak lengkap, di mana sebagian dari *sample input* tidak memiliki label.

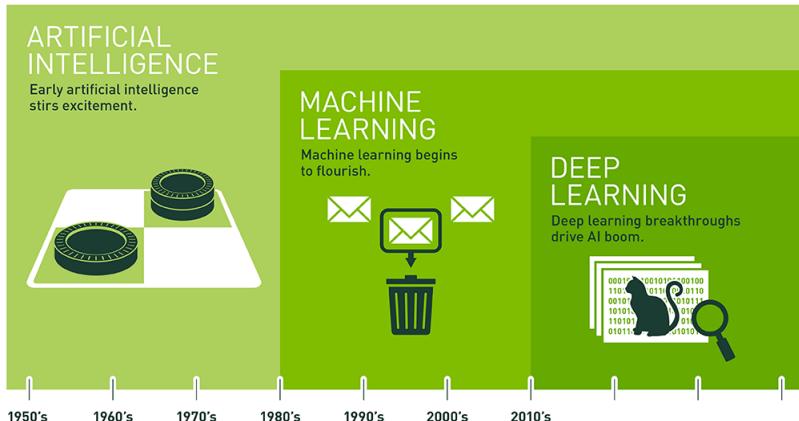
Algoritma klasifikasi dan algoritma regresi adalah jenis dari *Supervised Learning*. Algoritma klasifikasi digunakan ketika *output* dibatasi untuk satu set nilai terbatas. Untuk algoritma klasifikasi dengan kasus melakukan *E-Mail filtering*, *input* akan berupa *E-Mail* masuk, dan hasilnya adalah nama *folder* tempat *file* dari *E-Mail* tersebut. Untuk algoritma klasifikasi dengan kasus untuk mengidentifikasi *spam E-Mail*, hasilnya adalah prediksi *spam* atau *not spam* yang diwakili oleh nilai-nilai *boolean* benar dan salah. Algoritma regresi dinamai untuk *output* kontinu tersebut, yang berarti mereka memiliki nilai apa pun dalam suatu rentang. Contoh dari nilai kontinu adalah suhu, panjang, atau harga suatu objek.

Dalam algoritma *Unsupervised Learning*, algoritma ini membangun model matematika dari satu set data yang hanya berisi *input* dan tidak ada label *output* yang diinginkan. Algoritma *Unsupervised Learning* digunakan untuk menemukan struktur dalam data, seperti pengelompokan data. Algoritma *Unsupervised Learning* dapat menemukan pola dalam data, dan dapat mengelompokkan input ke dalam kategori, seperti dalam pembelajaran fitur. Pengurangan dimensi adalah proses mengurangi jumlah fitur atau *input* dalam satu set data.

2.3 Deep Learning

Deep Learning adalah salah satu dari sub domain *Machine Learning*, di mana klasifikasi pola dan representasi pembelajaran dicapai dengan mencakup berbagai *layer* dari tahapan penanganan data dalam model hierarki. Semakin luasnya implementasi dari *Deep Learning* pada akhir-akhir ini disebabkan oleh beberapa faktor, di antaranya adalah kemampuan pemrosesan dari *chip processing* yang semakin meningkat, harga perangkat keras komputer yang semakin murah, serta *Machine Learning* yang semakin banyak dikembangkan sehingga menjadi bermacam-macam dan semakin mudah untuk

dipahami. Masalah optimalisasi dari *Deep Learning* telah berhasil diselesaikan ketika algoritma dari *Unsupervised Learning* yang sangat efektif ditemukan pada tahun 2006. Perbandingan dari *Artificial Intelligence*, *Machine Learning*, dan *Deep Learning* dapat dilihat pada gambar 2.3.



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

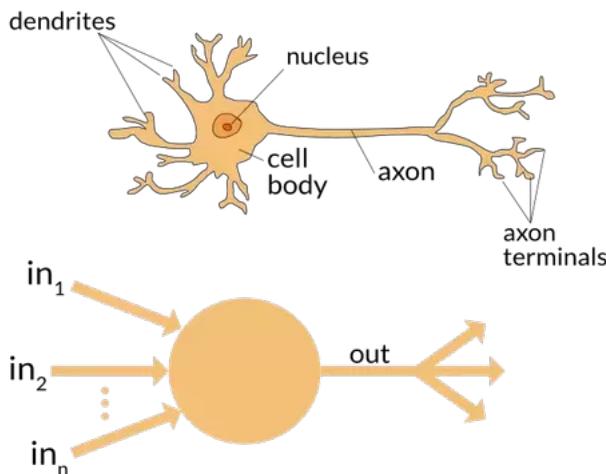
Gambar 2.3: Perbandingan dari *Artificial Intelligence*, *Machine Learning*, dan *Deep Learning*[3].

Sebagaimana besar model *Deep Learning* saat ini didasarkan pada *Artificial Neural Network*, khususnya Convolutional Neural Network, walaupun juga dapat menggunakan formula proporsional atau *latent variables organized layer-wise* pada *deep generative model* seperti *node* pada *deep belief networks* dan *deep Boltzmann machines*.

Dalam *deep learning*, setiap level melakukan pembelajaran untuk mengubah data *input* menjadi representasi yang sedikit lebih abstrak dan komposit. Dalam aplikasi *image recognition*, *input* berupa matriks piksel. Lapisan Representasional pertama dapat mengabstraksi piksel dan melakukan *encode* pada tepi. Lapisan kedua dapat menyusun dan melakukan *encode* pengaturan tepi. Lapisan ketiga dapat melakukan *encode* pada hidung dan mata. Kemudian lapisan keempat dapat mengenali bahwa gambat tersebut merupakan wajah. Hal yang terpenting adalah proses pada *deep*

learning dapat mempelajari fitur mana yang ditempatkan secara optimal di tingkat mana saja.

2.4 Artificial Neural Network



Gambar 2.4: Perbandingan *biological neuron* dan *artificial neuron*[4].

Neural Network adalah jaringan atau sirkuit neuron yang terdiri dari neuron atau node buatan[25]. Gambar 2.4 menunjukkan perbandingan *biological neuron* dan *artificial neuron*. Dengan demikian, *Neural Network* adalah *Biological Neural Network*, yang terdiri dari neuron biologis nyata, atau *Artificial Neural Network* untuk menyelesaikan masalah dalam bidang *Artificial Intelligence*[26]. Koneksi dari neuron biologis dimodelkan sebagai bobot. Bobot positif mencerminkan koneksi rangsang, sedangkan nilai negatif berarti koneksi penghambatan. Semua *input* dimodifikasi dengan bobot dan dijumlahkan. Aktivitas ini disebut sebagai kombinasi linear. Pada akhirnya, fungsi aktivasi akan mengontrol amplitudo dari *output*. Misalnya, rentang *output* yang diterima biasanya antara 0 dan 1, atau bisa jadi -1 dan 1.

Tidak seperti perhitungan model *von Neumann*, *Artificial Neural Network* tidak memisahkan memori dan pemrosesan dan bero-

perasi melalui aliran sinyal melalui koneksi internet, sehingga mirip dengan jaringan biologis. *Artificial Neural Network* dapat digunakan untuk pemodelan prediktif, kontrol adaptif, dan aplikasi di mana mereka dapat dilatih melalui *dataset*. *Self-learning* yang dihasilkan dari pengalaman dapat terjadi dalam jaringan, yang dapat mengambil kesimpulan dari serangkaian informasi yang kompleks dan tidak saling terkait.

Neural Network menerima *input* dan melewati sejumlah *hidden layers*. Setiap *hidden layer* memiliki sejumlah neuron, di mana setiap neuron terhubung sepenuhnya ke semua neuron di *layer* sebelumnya. Setiap *layer* dalam sebuah *layer* berfungsi secara independen. *Layer* terakhir dalam *Neural Network* disebut *output layer*, yang mewakili beberapa kelas yang dimiliki oleh *input layer*.

Model pada *Artificial Neural Network* pada dasarnya merupakan fungsi model matematika seperti pada persamaan 2.4.1.

$$f : X \rightarrow Y \quad (2.4.1)$$

Istilah *network* pada *Artificial Neural Network* merujuk pada interkoneksi dari beberapa *neuron* yang diletakkan pada lapisan yang berbeda. Secara umum, lapisan pada *Artificial Neural Network* dibagi menjadi tiga bagian :

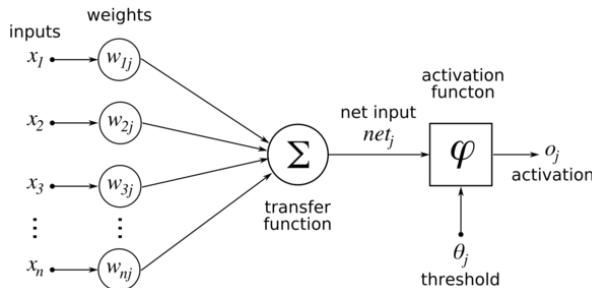
1. *Input Layer* yang terdiri dari *neuron* yang menerima data masukan dari variabel X . Semua *neuron* pada *layer* ini dapat terhubung ke *neuron* pada *hidden layer* atau langsung ke *output layer* jika jaringan tidak menggunakan *hidden layer*.
2. *Hidden Layer* yang terdiri dari *neuron* yang menerima data dari *input layer*.
3. *Output Layer* yang terdiri dari *neuron* yang menerima data dari *hidden layer* atau langsung dari *input layer* yang nilai *output*-nya melambangkan hasil kalkulasi dari X menjadi nilai Y .

Secara matematis, *neuron* merupakan sebuah fungsi yang menerima *input* dari *layer* sebelumnya $g_i(x)$ (lapisan ke- i). Fungsi ini pada umumnya mengolah sebuah vektor untuk kemudian diubah ke

nilai skalar melalui komposisi *nonlinear weighted sum*, yang dinyatakan dengan persamaan 2.4.2

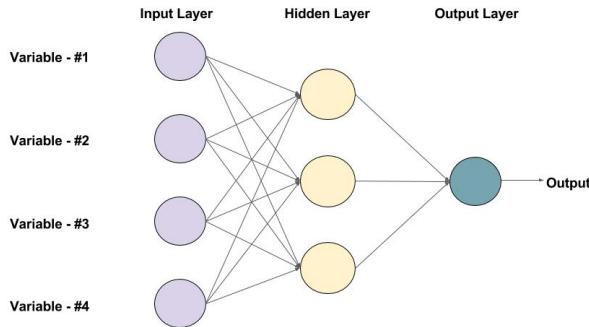
$$f(x) = K\left(\sum_i w_i g_i(x)\right) \quad (2.4.2)$$

di mana K merupakan fungsi khusus yang sering disebut dengan fungsi aktifasi dan w merupakan beban atau *weight*.



Gambar 2.5: *Artificial Neural Network model*[5].

2.5 Feedforward Neural Network



An example of a Feed-forward Neural Network with one hidden layer (with 3 neurons)

Gambar 2.6: *Feedforward Neural Network* dengan 1 *hidden layer* dan 3 *neurons*[6].

Gambar 2.6 merupakan ilustrasi dari *Feedforward Neural Network* dengan 1 *hidden layer* dan 3 *neurons*. *Feedforward Neural Network* adalah *Artificial Neural Network* di mana koneksi antara node tidak membentuk siklus. Dengan demikian, hal ini sangat berbeda dengan *Recurrent Neural Network*. *Feedforward Neural Network* adalah tipe pertama dan paling sederhana dari *Artificial Neural Network* yang dibuat. Dalam jaringan ini, informasi bergerak hanya dalam satu arah yaitu maju dari *input node* melalui *hidden layer* (jika ada) dan kemudian menuju *output node*. Tidak ada siklus atau *loop* dalam jaringan ini. *Feedforward Network* dapat dibangun dengan berbagai jenis unit, seperti *binary McCulloch-Pitts neurons*, yang paling sederhana dan merupakan *perceptron*.

2.6 Multilayer Perceptron

Multilayer Perceptron adalah bagian dari *Feedforward Artificial Neural Network*. *Multilayer Perceptron* terdiri dari setidaknya tiga lapisan node yaitu *input node*, *hidden layer*, dan *output layer*. Setiap *node* merupakan *neuron* yang menggunakan fungsi aktivasi nonlinear, kecuali untuk *input node*. *Multilayer Perceptron* menggunakan teknik *supervised learning* yang disebut dengan *backpropagation* untuk pembelajaran. Berbagai *layer* dan aktivasi non-linearnya membedakan *Multilayer Perceptron* dengan *Linear Perceptron*. Dengan demikian, *Multilayer Perceptron* dapat membedakan data yang tidak dapat dipisahkan secara linear.

2.6.1 Activation Function

Jika sebuah *Multilayer Perceptron* memiliki fungsi aktivasi linier di semua *neuron*, yaitu fungsi linier yang memetakan *weighted inputs* ke *output* dari masing-masing *neuron*, maka fungsi Aljabar linear menunjukkan bahwa sejumlah *layer* dapat dikurangi menjadi *two-layer input-output model*. Dalam *Multilayer Perceptron*, beberapa *neuron* menggunakan fungsi aktivasi nonlinear yang dikembangkan untuk memodelkan frekuensi dari potensial aksi, atau memetakan *biological neuron*. Dua fungsi aktivasi yang secara historis umum keduanya adalah *sigmoid*, dijelaskan oleh fungsi sebagai pada persamaan 2.6.1.1.

$$y(v_i) = \tanh(v_i) \text{ and } y(v_i) = (1 + e^{-v_i})^{-1} \quad (2.6.1.1)$$

Dalam perkembangan terbaru dari *Deep Learning*, *Rectifier Linear Unit (ReLU)* lebih sering digunakan sebagai salah satu cara yang mungkin untuk mengatasi masalah numerik yang berkaitan dengan *sigmoid*. Yang pertama adalah garis singgung hiperbolik yang berkisar dari -1 hingga 1, sedangkan yang lainnya adalah fungsi logistik, yang bentuknya serupa tetapi berkisar dari 0 hingga 1. Di sini, y_i adalah *output* dari *node (neuron)* ke i th dan v_i adalah jumlah terbobot dari koneksi *input*. Beberapa fungsi aktivasi alternatif lainnya, yaitu fungsi penyuarah dan fungsi softplus juga telah diusulkan. Fungsi aktivasi yang terspesialisasi mencakup fungsi basis radial yang digunakan dalam *radial basis networks*, merupakan kelas lain dari *supervised neural network model*.

2.6.2 Layers

Multilayer Perceptron terdiri dari tiga *layers* atau lebih (satu *input* dan satu *output node* dengan satu atau lebih *hidden layers*) dari node yang mengaktifkan fungsi nonlinier. Karena *Multilayer Perceptron* sepenuhnya terhubung, setiap *node* dalam satu lapisan terhubung dengan bobot tertentu yaitu w_{ij} pada setiap *node* di *layer* selanjutnya.

2.6.3 Learning

Pembelajaran terjadi dalam *perceptron* dengan mengubah bobot koneksi setelah setiap bagian data diproses, berdasarkan jumlah kesalahan dalam *output* dibandingkan dengan hasil yang diharapkan. Hal ini merupakan contoh dari *supervised learning*, dan dilakukan melalui *backpropagation*.

Fungsi untuk merepresentasikan kesalahan pada *output node* j pada *data point* ke n^{th} ditunjukkan dengan persamaan 2.6.3.1

$$e_j(n) = d_j(n) - y_j(n) \quad (2.6.3.1)$$

di mana d adalah nilai target dan y adalah nilai yang dihasilkan oleh *perceptron*. *Node weights* disesuaikan berdasarkan koreksi yang menimbulkan kesalahan di seluruh *output*, yang diberikan oleh fungsi yang ditunjukkan pada persamaan 2.6.3.2.

$$\mathcal{E}(n) = \frac{1}{2} \sum_j e_j^2(n). \quad (2.6.3.2)$$

Dengan menggunakan *gradient descent*, perubahan dari setiap bobot dapat dihitung seperti pada persamaan 2.6.3.3

$$\Delta w_{ji}(n) = -\eta \frac{\partial \mathcal{E}(n)}{\partial v_j(n)} y_i(n) \quad (2.6.3.3)$$

di mana y_i adalah *output* dari *neuron* sebelumnya dan η adalah *learning rate*, yang dipilih untuk memastikan bahwa bobot lebih cepat menyatu dengan respons, tanpa osilasi.

Turunan yang akan dihitung bergantung pada bidang lokal yang diinduksi v_j , yang bervariasi. *Output node* dari turunan ini dapat disederhanakan dengan menggunakan rumus seperti pada persamaan 2.6.3.4

$$-\frac{\partial \mathcal{E}(n)}{\partial v_j(n)} = e_j(n) \phi'(v_j(n)) \quad (2.6.3.4)$$

di mana ϕ' adalah turunan dari fungsi aktivasi yang dijelaskan di atas, yang dengan sendirinya tidak berbeda. Analisis menjadi lebih sulit dengan perubahan bobot pada *hidden node*, tetapi dapat ditunjukkan bahwa turunan yang relevan dapat dihitung seperti pada persamaan 2.6.3.5

$$-\frac{\partial \mathcal{E}(n)}{\partial v_j(n)} = \phi'(v_j(n)) \sum_k -\frac{\partial \mathcal{E}(n)}{\partial v_k(n)} w_{kj}(n). \quad (2.6.3.5)$$

Hal ini bergantung pada perubahan bobot dari node k th, yang mewakili *output layer*. Jadi, untuk mengubah bobot pada *hidden layer*, bobot pada *output layer* berubah sesuai dengan fungsi turunan dari fungsi aktivasi, dan karenanya algoritma ini mewakili *backpropagation* dari fungsi aktivasi.

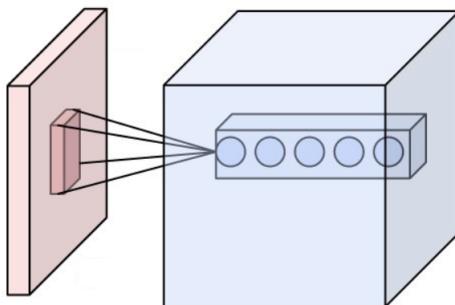
2.7 Convolutional Neural Network

Dalam *Deep Learning*, *Convolutional Neural Network (CNN)* adalah kelas dari *Deep Neural Network*, yang paling umum diterapkan untuk melakukan analisis citra visual. *Convolutional Neural Network* merupakan versi yang lebih sempurna dari *Multilayer Perceptron*. *Multilayer Perceptron* biasanya merujuk ke jaringan yang sepenuhnya terhubung, yaitu setiap *neuron* dalam satu

layer terhubung ke semua *neuron* di *layer* berikutnya. "The fully-connectedness" dari *network* ini membuat mereka rentan terhadap *overfitting data*. Cara untuk melakukan regularisasi termasuk menambahkan beberapa bentuk pengukuran besaran bobot ke dalam *loss function*. Namun, *Convolutional Neural Network* mengambil pendekatan yang berbeda terhadap proses regularisasi, yaitu dengan mengambil keuntungan dari pola hierarkis dalam data dan mengumpulkan pola yang lebih kompleks dengan menggunakan pola yang lebih kecil dan lebih sederhana.

Convolutional Network terinspirasi dari proses biologis dalam hal pola koneksi antara *neuron* menyerupai organisasi korteks visual hewan[27][28]. Neuron kortikal individual merespons rangsangan hanya di daerah terbatas pada bidang visual yang dikenal sebagai reseptif. Bidang reseptif dari *neuron* yang berbeda sebagian tumpang tindah sehingga menutupi seluruh bidang visual. *Convolutional Neural Network* menggunakan *pre-processing* yang relatif sedikit jika dibandingkan dengan algoritma *image classifier* lainnya. Aplikasi dari algoritma ini dapat dilihat dalam *image recognition and classification*, *medical image analysis*, dan *natural language processing*.

2.7.1 Convolutional Layer



Gambar 2.7: *Convolutional Layer*[7].

Convolutional layer adalah blok bangunan inti dari *Convolutional Neural Network*. Ilustrasi dari *Convolutional layer* dapat

dilihat pada gambar 2.7 Parameter dari *layer* terdiri dari satu set filter yang dapat dipelajari (atau *kernel*), yang memiliki bidang reseptif kecil, tetapi meperluas melalui kedalaman penuh dari *input volume*. Selama proses *forward pass*, setiap filter dilakukan proses konvolusi terhadap lebar dan tinggi dari *input volume*, menghitung *dot product* antara entri filter dan *input* dan menghasilkan *activation maps* berbentuk 2-dimensi dari filter tersebut. Sebagai hasilnya, jaringan ini mempelajari filter yang aktif ketika mendeteksi beberapa jenis fitur tertentu pada beberapa posisi spasial dalam *input*.

Menggabungkan *activation maps* untuk semua filter sepanjang *depth dimension* membentuk *full output volume* pada *convolution layer*. Setiap entri dalam *output volume* juga dapat diartikan sebagai *output* dari *neuron* yang terlihat pada daerah kecil di *input* dan berbagai parameter dengan *neuron* dalam *activation maps* yang sama.

Local Connectivity

Ketika berhadapan dengan *input* berdimensi tinggi seperti gambar, sangat tidak praktis untuk menghubungkan *neuron* ke semua *neuron* dalam volume sebelumnya karena arsitektur jaringan seperti itu tidak memperhitungkan struktur spasial data. *Convolutional networks* mengeksplorasi korelasi lokasi spasial dengan memberlakukan pola konektivitas lokal yang jarang antara *neuron* dari lapisan yang berdekatan di mana setiap *neuron* terhubung ke hanya sebagian kecil dari *input volume*.

Tingkat konektivitas ini adalah *hyperparameter* yang disebut bidang reseptif neuron. Koneksi bersifat lokal dalam dimensi ruang, bersamaan dengan lebar dan tinggi, tetapi selalu membentang sepanjang seluruh kedalaman dari *input volume*. Arsitektur seperti itu memastikan bahwa filter yang dipelajari menghasilkan respons terkuat terhadap pola input lokal spasial.

Spatial Arrangement

Tiga *hyperparameter* yang digunakan untuk mengontrol ukuran *output volume* dari *convolutional layer* adalah : *the depth*, *stride*, dan *zero-padding*.

The depth dari *output volume* mengontrol jumlah *neuron* dalam *layer* yang terhubung ke wilayah yang sama dari *input volume*. *Neuron* ini belajar untuk mengaktifkan berbagai fitur dalam

input. Misalnya, jika *convolutional layer* pertama mengambil gambar mentah sebagai *input*, maka *neuron* yang berbeda di sepanjang dimensi dari kedalaman dapat diaktifkan dengan adanya berbagai variasi orientasi dari tepi, atau gumpalan warna.

Stride mengontrol bagaimana kolom kedalaman di sekitar dimensi spasial (lebar dan tinggi) dialokasikan. Ketika langkahnya adalah satu maka filter akan dipindahkan satu piksel pada satu waktu. Hal ini menyebabkan bidang reseptif yang tumpang tindih antara kolom, dan juga *output volume* yang besar. Ketika langkahnya adalah dua maka filter akan melompat sebanyak 2 piksel sekaligus ketika mereka bergerak. Demikian pula, untuk bilangan bulat apapun $S > 0$, *stride* S menyebabkan filter diterjemahkan oleh unit S pada waktu per *output*. Dalam implementasinya, jumlah *stride* $S \geq 3$ sangat jarang. Bidang reseptif yang saling tumpang tindih lebih sedikit dan *output volume* yang dihasilkan memiliki dimensi spasial yang lebih kecil ketika panjang dari *stride* meningkat.

Terkadang lebih mudah untuk memasukkan *input* dengan nol pada batas *input volume*. Ukuran *padding* ini adalah *hyperparameter* ketiga. *Padding* menyediakan kontrol *output volume* pada ukuran spasial. Secara khusus, kadang-kadang diinginkan untuk secara tepat mempertahankan ukuran spasial dari *input volume*.

Ukuran spasial dari *output volume* dapat dihitung sebagai fungsi dari ukuran *input volume* W , ukuran bidang *kernel* dari *convolutional layer neurons* K , penerapan dari *stride* S , dan jumlah *zero padding* P digunakan untuk *border*. Rumus untuk menghitung berapa banyak *neuron* yang cocok dalam volume ditunjukkan pada persamaan 2.7.1.1

$$\frac{W - K + 2P}{S} + 1. \quad (2.7.1.1)$$

Jika angka ini bukan bilangan bulat, maka *stride* akan salah dan *neuron* tidak dapat dipasang agar sesuai dengan *input volume* dengan cara simetris. Secara umum, mengatur *zero padding* seperti yang ditunjukkan pada persamaan 2.7.1.2

$$P = (K - 1)/2 \quad (2.7.1.2)$$

ketika *stride* $S = 1$ memastikan bahwa *input volume* dan

output volume akan memiliki ukuran yang sama secara spasial.

Parameter Sharing

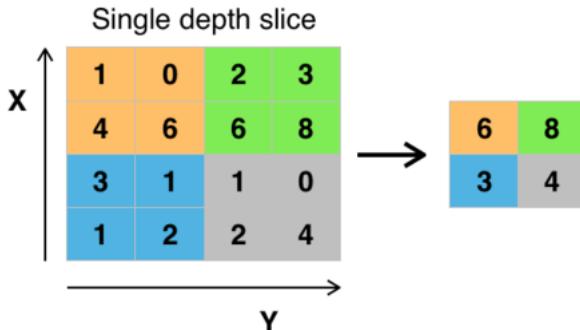
Skema *parameter sharing* digunakan dalam *convolutional layer* untuk mengontrol jumlah dari parameter. Parameter ini ber-gantung pada asumsi yaitu jika fitur *patch* berguna untuk menghitung pada beberapa posisi spasial, maka itu juga harus berguna untuk menghitung pada posisi lain. Dengan kata lain, akan digunakan *2-dimensional slice of depth* sebagai irisan kedalaman untuk membatasi *neuron* di setiap irisan kedalaman untuk menggunakan bobot dan bias yang sama.

Karena semua *neuron* dalam irisan kedalaman tunggal berbagi parameter yang sama, *forward pass* di setiap irisan kedalaman *convolutional layer* dapat dihitung sebagai *convolution* dari bobot neuron dengan *input volume*. Hasil dari *convolution* ini adalah *activation maps*, dan set dari *activation maps* untuk setiap filter yang berbeda digabungkan bersama dengan dimensi kedalaman untuk menghasilkan *output volume*. *Parameter sharing* berkontribusi pada *translation invariance* pada *CNN Architecture*.

2.7.2 Pooling Layer

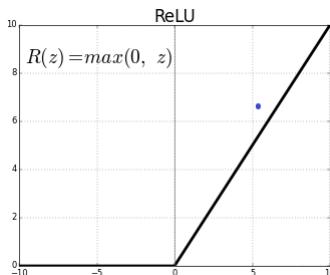
Konsep penting lain dari *Convolutional Neural Network* adalah *pooling*, yang merupakan bentuk non-linear dari *down-sampling*. Ada beberapa fungsi non-linear untuk menerapkan *pooling* di mana *max-pooling* adalah yang paling umum. *Pooling* membagi gambar *input* menjadi satu set persegi panjang dan untuk setiap sub-wilayah tersebut, nilai dari *output* akan maksimum.

Pooling Layer beroperasi secara independen pada setiap irisan kedalaman *input* dan mengubah ukurannya secara spasial. Bentuk yang paling umum adalah *pooling layer* dengan filter ukuran 2×2 diterapkan dengan *stride* 2 sampel bawah pada setiap irisan kedalaman *input* dengan 2 pada setiap lebar dan tinggi, sehingga membuang 75% dari aktivasi. Dalam hal ini, setiap operasi maksimal lebih dari 4 angka. Dimensi kedalaman akan tetap dan tidak berubah. Ilustrasi dari *Max Pooling* dengan 2×2 filter dan *stride* = 2 dapat dilihat pada gambar 2.8.



Gambar 2.8: *Max Pooling dengan 2×2 filter dan stride = 2[7].*

2.7.3 ReLU (Rectified Linear Unit) Activation Function



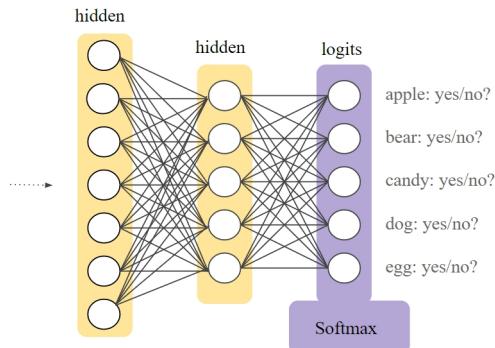
Gambar 2.9: *Fungsi Aktivasi ReLU*

ReLU merupakan *non-linear activation function* yang paling sederhana. *ReLU* adalah salah satu fungsi aktivasi yang paling banyak digunakan saat ini. Karena hal itulah, fungsi *ReLU* ini digunakan di hampir semua arsitektur *Convolutional Neural Network* atau *Deep Learning*. Grafik dari fungsi aktivasi *ReLU* ditunjukkan pada gambar 2.9. Rumus dari fungsi aktivasi *ReLU* ditunjukkan pada persamaan 2.7.3.1

$$R(z) = \max(0, z) \quad (2.7.3.1)$$

Fungsi ini akan menghasilkan nilai '0' ketika *input* kurang dari atau sama dengan '0' dan akan menghasilkan nilai positif jika *input* lebih dari '0'.

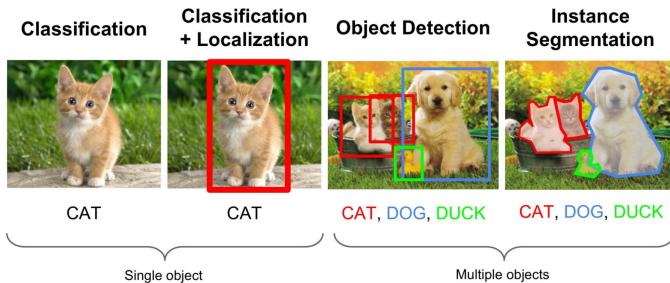
2.7.4 Softmax Classifier



Gambar 2.10: Ilustrasi pada *Softmax Layer* pada *neural network*[8].

Regresi logistik menghasilkan probabilitas antara 0 dan 1. Sebagai contoh, *output* dari regresi logistik sebesar 0.8 dari pengelempokan *E-Mail* menunjukkan kemungkinan sebesar 80 % dari *E-Mail* menjadi *spam* dan 20 % kemungkinan bukan *spam*. Jumlah probabilitas dari suatu kejadian yang berlangsung adalah 1. *Softmax* diimplementasikan melalui *neural network layer* sebelum *output layer*. *Softmax layer* harus memiliki jumlah *node* yang sama dengan *output layer*. Ilustrasi *Softmax Layer* pada *neural network* dapat dilihat pada gambar 2.10.

2.8 Image Classification



Gambar 2.11: Contoh dari *Image Classification* beserta *Localization* dan *Object Detection*[9]

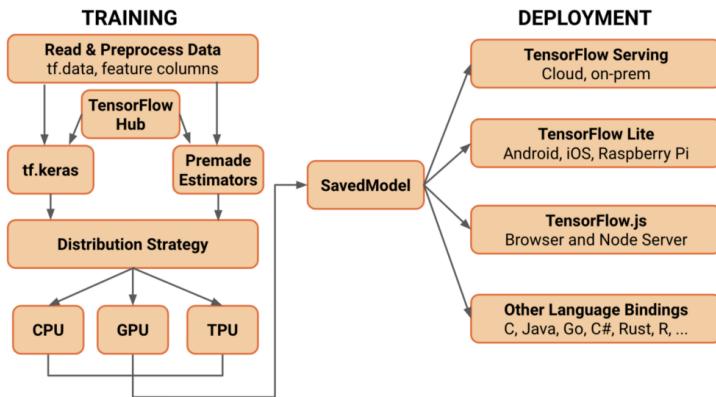
Dalam ilmu *Machine Learning* dan statistika, klasifikasi merupakan sebuah proses untuk mengidentifikasi beberapa kategori maupun kelompok, berdasarkan serangkaian proses dari *Data Training* yang kategori keanggotaannya sudah diketahui. Gambar 2.11 menunjukkan contoh dari *image classification*, *localization*, dan *object detection*. Salah satu contoh dari klasifikasi adalah *Image Classification*. Pada *Image Classification*, dataset yang digunakan adalah gambar yang kategori keanggotaannya sudah didefinisikan atau sudah diketahui sebelumnya. Sebuah algoritma yang digunakan untuk mengimplementasikan proses klasifikasi, disebut *Classifier*. Pada tugas akhir ini, *Classifier* yang digunakan adalah *Convolutional Neural Network (CNN)*.

2.9 TensorFlow

TensorFlow dibuat oleh tim dari *Google Brain*, *TensorFlow* adalah *open source library* untuk komputasi numerik dan *Machine Learning* skala besar. *TensorFlow* menggabungkan satu set model dari algoritma *Machine Learning* dan *Deep Learning*. *TensorFlow* menggunakan *Python* untuk menyediakan *front-end API* untuk membangun aplikasi dengan kerangka kerja, dengan mengeksekusi aplikasi tersebut dalam bahasa *C++*.

TensorFlow dapat melatih dan menjalankan *Deep Neural Network* untuk melakukan *Image Recognition and Classification*, *Natural Language Processing*, *Text Classification*, dan masih ba-

nyak yang lain. Aplikasi *TensorFlow* dapat dijalankan pada sebagian besar *platform*, seperti *local machine*, *iOS*, *Android*, *CPUs*, atau *GPUs*. Model yang dihasilkan dari *TensorFlow* dapat digunakan di hampir semua perangkat yang akan digunakan untuk melakukan prediksi. *Workflow* pada *TensorFlow* ditunjukkan pada gambar 2.12



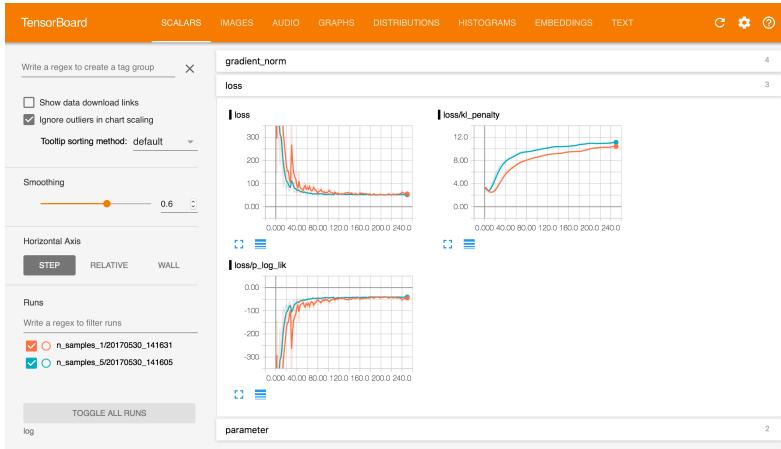
Gambar 2.12: *Workflow* pada *TensorFlow*[10].

2.9.1 TensorBoard

TensorBoard adalah seperangkat aplikasi web untuk memeriksa dan memahami grafik dari model yang telah dibuat dengan menggunakan *TensorFlow*. Tampilan antarmuka pada *TensorBoard* ditunjukkan pada gambar 2.13. *TensorBoard* menyediakan visualisasi dan peralatan yang diperlukan untuk eksperimen pada *machine learning* seperti :

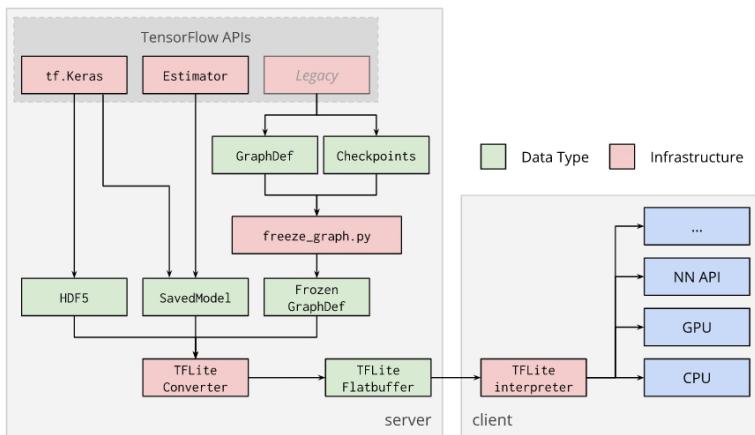
1. Melacak dan memvisualisasikan metrik seperti *loss* dan akurasi.
2. Memvisualisasikan grafik model (*ops* dan *layer*).
3. Melihat histogram dari bobot, bias, atau *tensor* lainnya secara *real time*.
4. Menampilkan gambar, teks, dan data audio.

5. Membuat profil program *TensorFlow*.



Gambar 2.13: Dashboard pada *TensorBoard*[10].

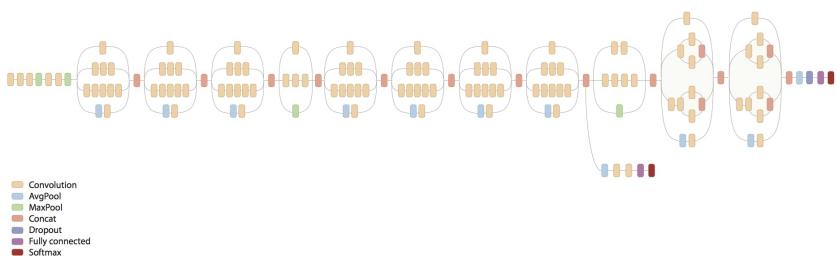
2.9.2 TensorFlow Lite



Gambar 2.14: Workflow pada *TensorFlow Lite*[10].

TensorFlow Lite adalah *deep learning framework* yang di-distribusikan secara *open source* yang dikhususkan untuk *mobile devices*. *TensorFlow Lite* memungkinkan inferensi latensi rendah pada *machine learning models* dengan ukuran biner yang kecil dan mendukung *hardware acceleration* dengan performa yang tinggi. *Workflow* pada *TensorFlow Lite* ditunjukkan pada gambar 2.14

2.10 Inception-v3

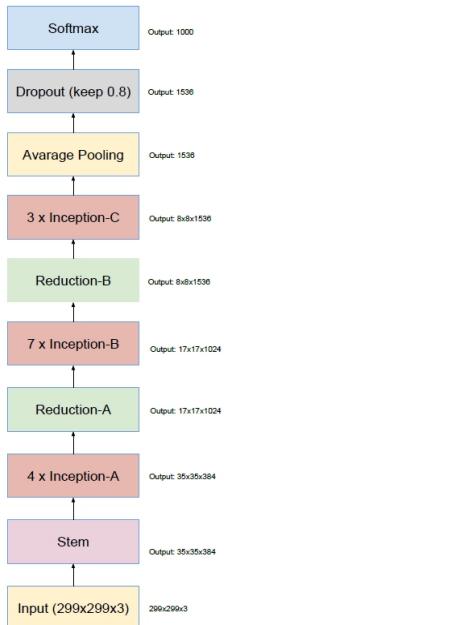


Gambar 2.15: Arsitektur dari *Inception-v3*[11].

Inception-v3 adalah salah satu arsitektur dalam *CNN* untuk melakukan *Image Classification* yang banyak digunakan yang dapat mencapai akurasi yang signifikan[11]. Arsitektur dari *Inception-v3* ditunjukkan pada gambar 2.15. Model ini memiliki campuran dari *symmetric* maupun *asymmetric building blocks*, meliputi :

1. *convolutions*
2. *average pooling*
3. *max pooling*
4. *concat*s
5. *dropouts*
6. *fully connected layers*

2.11 Inception-v4



Gambar 2.16: Skema arsitektur dari *Inception-v4*[12].

Inception-v4 [12] merupakan evolusi dari *GoogleNet* atau *Inception-v1*, yang memiliki arsitektur yang lebih disederhanakan dan memiliki lebih banyak *inception modules* dari versi sebelumnya, yaitu *Inception-v3*. *Inception-v4* merupakan varian dari *Inception* tanpa koneksi residual. Arsitektur dari *Inception-v4* ditunjukkan pada gambar 2.16.

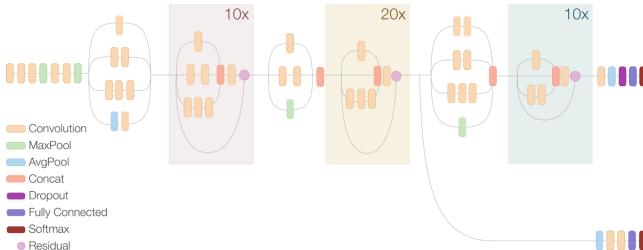
2.12 Inception-ResNet-v2

Inception-ResNet-v2 merupakan arsitektur dari *CNN* yang menunjukkan performa yang tinggi dengan biaya komputasi yang cukup rendah[12]. *Inception-ResNet-v2* menggabungkan dua arsitektur untuk meningkatkan performa dan akurasinya. *Residual connections* memungkinkan sebuah model untuk menjadi lebih efektif, dan memungkinkan para peneliti untuk berhasil melatih *Deep Neu-*

Inception Resnet V2 Network



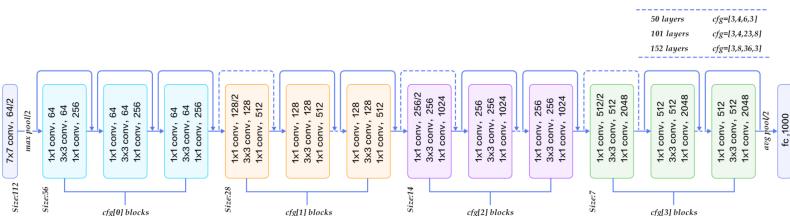
Compressed View



Gambar 2.17: Arsitektur dari *Inception-ResNet-v2*[12].

ral Network lebih dalam untuk mencapai performa yang tinggi. Hal ini juga memungkinkan penyederhanaan yang cukup signifikan dari blok *Inception*. Arsitektur dari *Inception-ResNet-v2* ditunjukkan pada gambar 2.17.

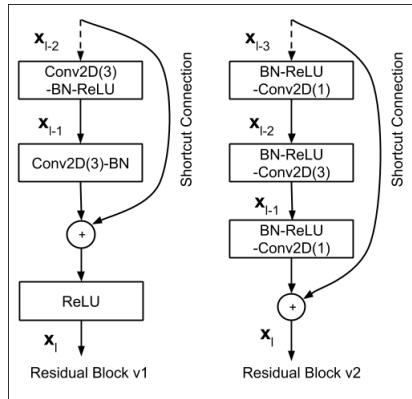
2.13 ResNet-v2-152



Gambar 2.18: Arsitektur dari *ResNet-v2*[13].

Residual Neural Network(ResNet) ditemukan pada tahun 2015 dengan mengenalkan arsitektur dengan konsep *skip connections* dan fitur seperti *heavy batch normalization*[13]. *ResNet-v2* merupakan model yang mengalami perbaikan dari versi sebelumnya yaitu *ResNet-v1*. Perbaikan yang signifikan terutama ditemukan dalam pengaturan *layer* pada *residual block*. Arsitektur dari *ResNet-v2-152* ditunjukkan pada gambar 2.18. Gambar 2.19 menunjukkan

perbandingan dari *ResNet-v1* dengan *ResNet-v2*.



Gambar 2.19: Perbandingan dari *ResNet-v1* dengan *ResNet-v2*

2.14 MobileNetV1

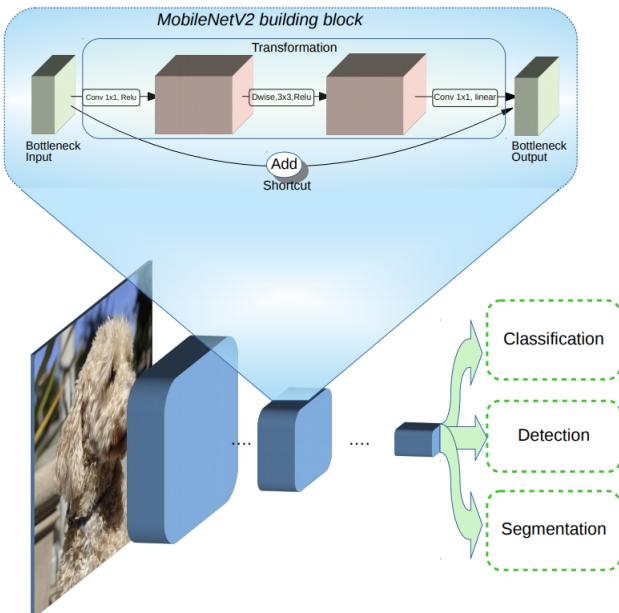


Gambar 2.20: *MobileNet* dan beberapa contoh aplikasinya[14].

MobileNet merupakan model yang sangat efisien dan dikhususkan untuk penggunaan pada *mobile device* maupun *embedded vision applications*[14]. Gambar 2.20 menunjukkan *MobileNet* dan beberapa contoh aplikasinya. *MobileNet* didasarkan pada arsitektur ramping yang menggunakan konvolusi yang terpisah untuk membangun *Deep Neural Network* yang ringan. *MobileNet* memperkenalkan paramater sederhana yang efektif untuk meningkatkan latensi

maupun tingkat akurasi. Dengan adanya parameter ini, pengguna dapat membuat dan mengatur model berdasarkan permasalahan yang ingin diselesaikan. Model ini dapat menyelesaikan permasalahan seperti klasifikasi, pendekripsi, dan segmentasi seperti model yang cukup populer yaitu *Inception*.

2.15 MobileNetV2

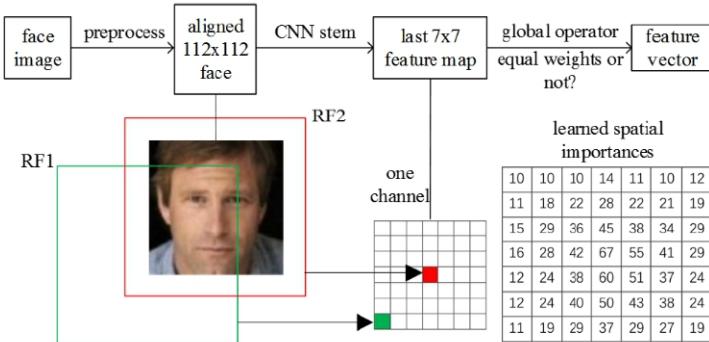


Gambar 2.21: Arsitektur dari *MobileNetV2*[15].

MobileNetV2 merupakan perbaikan dan perubahan dari versi sebelumnya yaitu *MobileNetV1* dalam hal performa dan dapat menjalankan tugas dengan efisien[15]. Arsitektur dari *MobileNetV2* seperti pada gambar 2.21 didasarkan pada struktur residu yang terbalik di mana *input* dan *output* dari *residual block* adalah *bottleneck layer* yang berbeda dengan model *traditional residual* yang menggunakan representasi yang diperluas dalam *input* tersebut. *MobileNetV2* menggunakan konsep dari *MobileNetV1* yaitu menggunakan konvolusi yang dapat dipasahkan secara mendalam sebagai blok ar-

sitektur yang efisien.

2.16 MobileFaceNets



Gambar 2.22: Arsitektur dari *MobileFaceNets*[16].

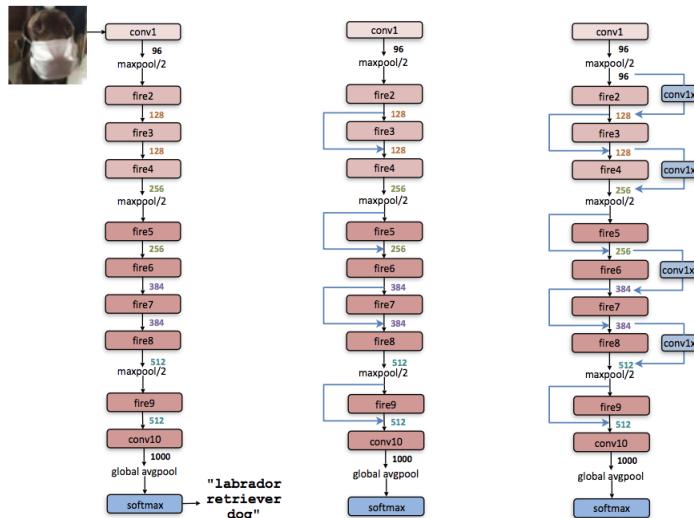
MobileFaceNets merupakan model dari *CNN* yang sangat efisien, karena hanya menggunakan kurang dari 1 juta parameter dan lebih digunakan secara spesifik dalam hal *real-time face verification* pada *mobile and embedded devices*[16]. Arsitektur dari *MobileFaceNets* ditunjukkan pada gambar 2.22. Kelemahan dari model *CNN* sebelumnya disempurnakan di dalam model *MobileFaceNets* ini. *MobileFaceNets* pada kondisi tertentu dapat melebihi tingkat akurasi dan kecepatan dari model *MobileNetV2*. Pada arsitektur ini, semua *convolution networks* pada ruang pencarian terdiri dari *convolutional layers* dengan struktur yang identik tetapi dengan bobot yang berbeda.

2.17 SqueezeNet

SqueezeNet merupakan model yang dapat mencapai tingkat akurasi yang sama seperti model *AlexNet* pada *ImageNet Dataset* dengan parameter yang lebih sedikit[17]. Pada tingkat akurasi tertentu, biasanya dimungkinkan untuk mengidentifikasi beberapa arsitektur dari *Deep Neural Network(DNN)* yang mencapai tingkat akurasi tersebut. Dengan tingkat akurasi yang setara, arsitektur *DNN* yang lebih kecil menawarkan setidaknya tiga keuntungan, yaitu :

1. *DNN* yang lebih kecil membutuhkan lebih sedikit komunikasi lintas *server* dalam proses *distributed training*.
2. *DNN* yang lebih kecil membutuhkan *bandwidth* yang lebih sedikit untuk mengekspor model baru dari *cloud* ke *target devices*.
3. *DNN* yang lebih kecil lebih cocok untuk digunakan pada *FPGA* dan perangkat keras lain dengan kapasitas memori yang terbatas.

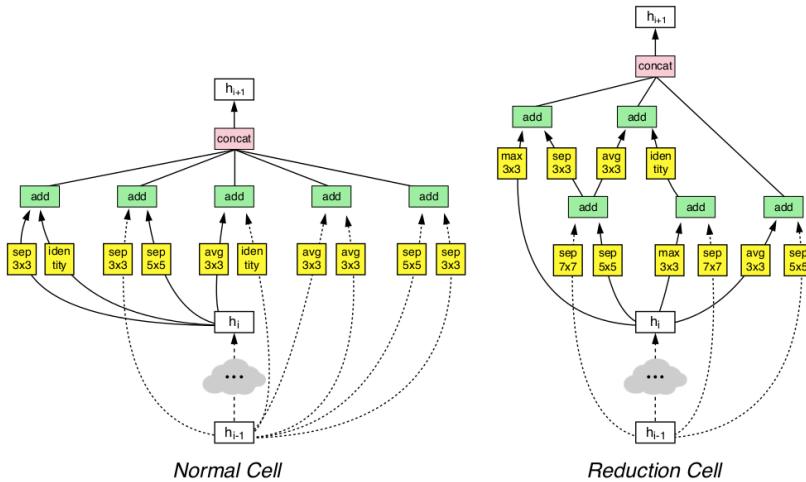
Selain hal itu, dengan menggunakan teknik kompresi, model *SqueezeNet* yang dihasilkan hanya menggunakan ukuran yang sangat kecil, yaitu sekitar 0.5 MB. Arsitektur dari *SqueezeNet* ditunjukkan pada gambar 2.23.



Gambar 2.23: Arsitektur dari *SqueezeNet*[17].

2.18 NasNet Mobile

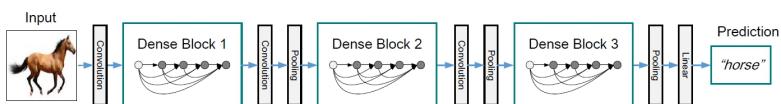
Neural Architecture Search(NAS) merupakan algoritma untuk melakukan pencarian terhadap *Neural Network Architecture* yang terbaik[18]. Dalam *ImageNet image classification*, algoritma *Nas-Net* dapat mendapatkan tingkat akurasi yang lebih tinggi daripada beberapa model lain seperti *Inception-v3* dan *ResNet* dengan meng-



Gambar 2.24: Arsitektur dari NasNet[18].

gunakan parameter yang lebih sedikit dan biaya komputasi yang lebih rendah. *NasNet Mobile* merupakan versi *mobile* dari *NasNet* dengan parameter yang lebih sedikit dan dikhususkan untuk *mobile devices*. Arsitektur dari *NasNet* ditunjukkan pada gambar 2.24.

2.19 DenseNet



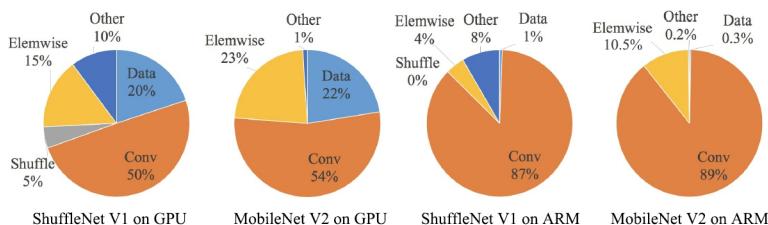
Gambar 2.25: Arsitektur dari DenseNet[19].

Dense Convolutional Neural Network (DenseNet) merupakan pengembangan tingkat lanjut untuk meningkatkan kedalaman dari *Deep Convolutional Neural Network*[19]. Arsitektur dari *DenseNet* seperti pada gambar 2.25 menggunakan parameter yang lebih sedikit daripada model tradisional *CNN* lainnya, karena tidak perlu melakukan *learning* pada redundant feature maps. Masalah yang muncul dari *Deep Convolutional Neural Network* adalah masalah

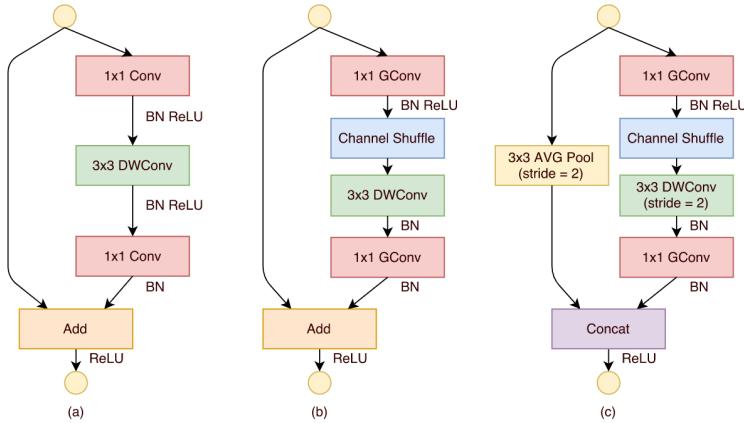
pada saat proses *training*, karena aliran informasi dan gradien yang kurang efisien pada setiap *layer*. *DenseNet* berhasil menyelesaikan permasalahan ini karena setiap *layer* memiliki akses langsung ke gradien dari *loss function* dan *original input image*. *DenseNet* berhasil mendapatkan peningkatan akurasi yang signifikan dengan membutuhkan lebih sedikit proses komputasi untuk mendapatkan performa yang tinggi.

2.20 ShuffleNetV2

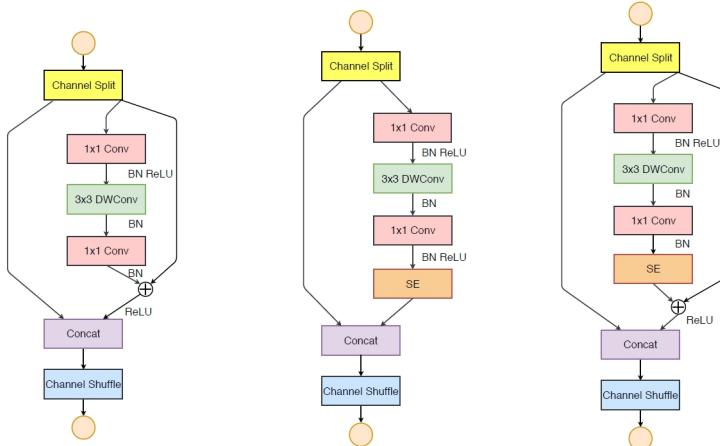
ShuffleNet merupakan salah satu arsitektur *CNN* yang didesain secara spesifik untuk *mobile devices* dengan kemampuan komputasi yang sangat terbatas (10-150 MFLOPs)[20]. Perbandingan arsitektur *ShuffleNet* dengan *MobileNet* dapat dilihat pada gambar 2.26. Arsitektur *ShuffleNet* yang ditunjukkan pada gambar 2.27 menggunakan dua operasi baru, yaitu *pointwise group convolution* dan *channel shuffle*, untuk mengurangi biaya komputasi dengan tetap menjaga tingkat akurasi. *ShuffleNet V2*[21] seperti pada gambar 2.28 merupakan penyempurnaan dari *ShuffleNet* dengan tujuan untuk lebih meningkatkan akurasi.



Gambar 2.26: Perbandingan *run time* *ShuffleNet* dengan *MobileNet*



Gambar 2.27: Arsitektur dari *ShuffleNet*[20].



Gambar 2.28: Arsitektur dari *ShuffleNet V2*[21].

2.21 Wajah

Wajah atau muka adalah bagian depan dari kepala, pada manusia meliputi wilayah dari dahi hingga dagu, termasuk rambut, dahi, alis, pelipis, mata, hidung, pipi, mulut, bibir, gigi, kulit,

dan dagu. Wajah terutama digunakan untuk ekspresi wajah, penampilan, serta identitas. Penampilan wajah sangat penting untuk pengenalan dan komunikasi manusia. Otot-otot wajah pada manusia memungkinkan ekspresi emosi. Wajah itu sendiri adalah daerah yang sangat sensitif dari tubuh manusia dan ekspresinya dapat berubah ketika otak dirangsang oleh banyak indera manusia, seperti sentuhan, suhu, bau, rasa, pendengaran, gerakan, rasa lapar, atau rangsangan visual.

2.21.1 Simetris Wajah

Simetris wajah dianggap sebagai tanda bagi kesehatan selama masa kanak-kanak dan pengembangan karena sumber-sumber stres lingkungan dan genetik seperti patogen atau tingkat mutasi, akan menguji kemampuan individu untuk menjaga stabilitas perkembangan dan ketahanan pertumbuhan asimetris. Dua penelitian sebelumnya yang menguji efek simetris wajah pada penilaian kondisi kesehatan menunjukkan hubungan yang positif.

Penelitian mengenai hubungan atau keterkaitan antara simetris wajah dengan indeks kesehatan juga semakin banyak diusulkan. Thornhill & Gangestad [29] menemukan hubungan positif antara wajah asimetri dan jumlah serangan penyakit pilek dan flu pada 3 tahun sebelumnya [29]. Asimetri wajah ditemukan sedikit terkait dengan jumlah hari sakit pilek dan flu, dan juga dengan penggunaan antibiotik, tetapi tidak berhubungan terhadap *gastroenteritis* atau sering disebut dengan flu perut atau flu lambung. Kemudian, Zebrowitz & Rhodes [30] menemukan bahwa simetris wajah terkait dengan penilaian dokter kesehatan, tetapi hal ini hanya berlaku di antara mereka yang memperoleh skor lebih rendah dari rata-rata yang sehubungan dengan simetris wajah [30].

Bertententangan dengan hipotesis bahwa simetris wajah merupakan salah satu indeks kesehatan, hasil dari studi terbesar tentang asimetris wajah dan kesehatan menyatakan bahwa sampai saat ini tidak ditemukan hubungan antara variabel-variabel ini [31]. Peneliti menggunakan data dari *British cohort study* pada 4732 individu dan menemukan bahwa simetris wajah pada usia 15 tidak ada keterkaitan dengan kesehatan anak, termasuk proporsi tindakan pada masa kanak-kanak yang dihabiskan secara tidak sehat, jumlah rata-rata gejala penyakit per tahun, dan jumlah total infeksi.

2.21.2 Adipositas Wajah

Obesitas diketahui terkait dengan sejumlah risiko kesehatan, dan sering diindeks berdasarkan berat, indeks massa tubuh seperti *BMI*, yaitu indeks berat badan dengan tinggi badan atau persentase lemak pada tubuh. Penelitian terbaru menunjukkan bahwa berat yang dirasakan dinilai dari wajah sendiri, juga disebut sebagai adipositas wajah, juga dapat memberikan tanda penting bagi kondisi kesehatan.

Studi perceptual telah menunjukkan adipositas wajah di wajah orang dewasa berhubungan negatif dengan penilaian kesehatan [32, 33], tetapi positif untuk penilaian kesehatan pada wajah bayi [34]. Terdapat bukti yang menyatakan bahwa adipositas yang didapatkan dari citra wajah memiliki keterkaitan dengan kondisi kesehatan yang sebenarnya. Misalnya, individu yang dinilai lebih berat melaporkan bahwa intensitas pilek lebih sering terjadi, lebih tahan lama, dan memiliki tekanan darah lebih tinggi [32]. Kemudian, entitas berat badan juga ditemukan memiliki keterkaitan dengan kondisi umum kesehatan termasuk hal-hal yang berkaitan dengan kondisi fisik dan psikologis kesehatan. Kondisi kesehatan juga tidak hanya dilihat atau diprediksidi berat badan, melainkan juga resiko obesitas yang dialami ketika dewasa, gejala penyakit, kondisi kronis, dan semua penyebab kematian.

Pada pria, persepsi adipositas wajah juga telah dikaitkan dengan respon antibodi terhadap vaksinasi Hepatitis C [35]. Hubungan ini sesuai dengan temuan bahwa reaksi vaksin hepatitis akan lebih kuat pada individu dengan berat badan yang lebih rendah dari *BMI*. Selain itu, adipositas wajah ditemukan sebagai irisan hubungan antara kekuatan respon antibodi terhadap Hepatitis C dan respons yang dirasakan pada individu, menunjukkan bahwa adipositas pada wajah adalah tanda yang valid dan digunakan sebagai indeks kesehatan [35].

Distribusi lemak dapat menjadi lebih informatif daripada massa lemak *per se* ketika digunakan untuk menentukan kondisi kesehatan. Jika adipositas pada wajah memberikan informasi penting untuk distribusi lemak pada tubuh, hal ini dapat menjelaskan hubungan yang lebih kuat antara persepsi kegemukan dari gambar wajah relatif ke ukuran indeks tubuh yang lebih umum seperti *BMI* atau persentase lemak.

Tiga aspek bentuk wajah yang dapat diukur juga telah diidentifikasi sebagai faktor yang mempengaruhi penilaian berat badan dan dapat memberikan target untuk menguji keterkaitan dengan hasil kesehatan, yaitu rasio lebar terhadap tinggi, rasio keliling terhadap luas, dan rasio lebar pipi terhadap rahang [36]. Tiga langkah ini belum tentu menangkap semua aspek bentuk wajah yang berhubungan dengan berat badan yang dirasakan dari wajah. Pendekatan secara empiris juga dilakukan untuk mendapatkan lebih banyak ukuran global dari bentuk wajah yang menjadi ciri adipositas wajah dengan menggunakan analisis komponen utama *PCA* dari *landmark* menangkap struktur dua dimensi (2D) gambar wajah [37] atau seluruh permukaan tiga dimensi (3D) wajah [38]. Hal ini menunjukkan bahwa masih harus ditunjukkan bagaimana pengukuran tersebut berhubungan dengan kondisi kesehatan yang dirasakan dan kondisi kesehatan yang diukur.

2.21.3 Kondisi Kulit (Tekstur dan Warna)

Warna dan tekstur kulit adalah tanda yang mudah dibentuk (dilihat) dan dapat berubah sebagai respons terhadap penyakit dalam waktu singkat. Untuk alasan ini, telah dikemukakan bahwa tanda tersebut harus dapat menyediakan informasi yang lebih relevan untuk kesehatan saat ini daripada bentuk informasi seperti maskulinitas wajah atau simetris wajah [39]. Studi yang menunjukkan korelasi antara kesehatan yang dirasakan pada seluruh wajah dan *isolated patches* pada kulit dari wajah yang sama membuktikan relevansi informasi kulit dalam perceptual penilaian kesehatan [40].

Informasi mengenai kulit dapat dibagi menjadi tanda yang lebih spesifik termasuk topografi permukaan (benjolan dan kerutan), warna dan distribusi warna, masing-masing telah terbukti mempengaruhi penilaian kesehatan, meskipun distribusi warna mungkin lebih berpengaruh dari topografi permukaan [41].

Ketika warna wajah dimanipulasi sepanjang tiga sumbu konsisten dengan persepsi warna manusia, yaitu kekuningan, kemerahan, atau tingkat kecerahan, peningkatan ketiganya mengarah ke lebih positif terhadap penilaian kondisi kesehatan [42]. Distribusi warna di wajah juga penting dalam penilaian kesehatan, dengan homogen distribusi warna meningkatkan penilaian kesehatan dibandingkan dengan distribusi warna yang tidak rata [43, 44, 45]. Kon-

tras dalam pencahayaan dan warna antara kulit wajah dan fitur-fitur seperti mata dan bibir juga berkontribusi pada penilaian kesehatan [42].

Dalam hal validitas tanda atau isyarat, ditunjukkan bahwa penilaian dari kondisi kesehatan akan lebih akurat ketika wajah yang disajikan mengandung informasi kulit yang relevan saja [46]. Hal ini konsisten dengan argumen di atas di mana tanda warna kulit (termasuk distribusi tekstur, warna, dan distribusi warna) dapat memberikan lebih banyak informasi kesehatan daripada bentuk wajah.

2.21.4 Ekspresi Wajah

Hingga saat ini, belum ada studi eksperimental yang menyelidiki peran ekspresi atau emosi pada kesehatan yang dirasakan. Satu studi menyelidiki senyum dalam foto dan keterkaitannya dengan hasil kesehatan [47]. Para peneliti menemukan dari foto-foto pemain *baseball* di liga utama di Amerika Serikat bahwa orang-orang yang memperlihatkan *Duchenne smile* secara penuh (dengan aktivitas otot *orbicularis* yang menyebabkan kerutan di sekitar mata selain aktivitas zygomatic yang meningkatkan sudut mulut), sepuh kemungkinan akan meninggal dalam beberapa tahun ke depan, dibandingkan terhadap individu yang tidak tersenyum. Penulis beralasan bahwa individu yang tersenyum dalam sebuah foto cenderung lebih sering tersenyum secara umum dan bahwa suasana hati berhubungan dengan fisiologi dan kesehatan umum. Keterkaitan antara suasana hati dan kesehatan telah diterima secara luas [48], sehingga prospek bahwa tanda wajah terhadap suasana hati akan memengaruhi penilaian kesehatan mengikuti secara logis.

Kurangnya studi empiris di bidang ini kemungkinan berasal dari penggunaan foto gaya paspor standar dalam studi persepsi wajah. Ada beberapa pekerjaan yang telah menunjukkan bahwa bahkan di antara foto standar, di mana peserta diminta untuk memegang ekspresi netral, penilai dapat menilai gambar individu yang kurang tidur sebagai lebih lelah dan lebih sedih daripada gambar dari peserta yang sama setelah tidur normal [49]. Gambar yang kurang tidur juga dikatakan memiliki bentuk mulut yang terlihat sedih dan kelopak mata yang menggantung. Meskipun penelitian ini tidak menyelidiki dari persepsi kesehatan, peneliti ini menyoroti

kemungkinan bahwa ada variasi ekspresi di antara foto-foto standar, dan bahwa penilai dapat menggunakan informasi ini untuk menginformasikan penilaian. Suasana hati yang tampak dari pose wajah netral sangat bervariasi dan merupakan kekuatan pendorong di balik banyak attribut sosial, misalnya kepercayaan terhadap gambar wajah. Oleh karena itu, penelitian lebih lanjut untuk mengeksplorasi penggunaan tanda atau isyarat pada wajah yang berkaitan dengan kondisi kesehatan dan validitasnya dalam penilaian kesehatan akan sangat diperlukan.

Halaman ini sengaja dikosongkan

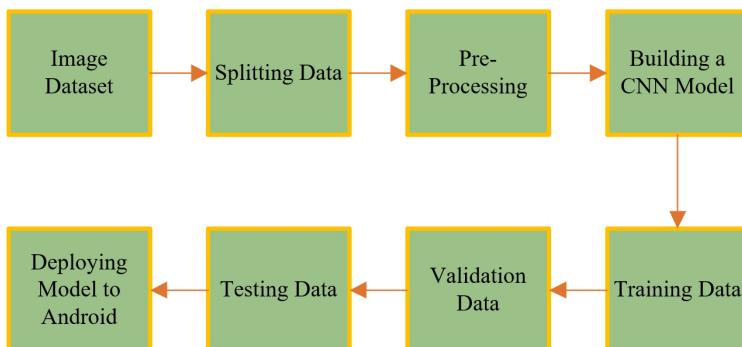
BAB 3

DESAIN DAN IMPLEMENTASI SISTEM

Penelitian ini dilaksanakan sesuai dengan sistem berikut dengan implementasinya. Desain sistem merupakan konsep dari pembuatan dan perancangan infrastruktur dan kemudian diwujudkan dalam bentuk blok-blok alur yang harus dikerjakan. Pada bagian implementasi merupakan pelaksanaan teknis untuk setiap blok pada desain sistem.

3.1 Desain Sistem

Tugas akhir ini merupakan salah satu bentuk penerapan dari disiplin ilmu *Deep Learning* yang bertujuan untuk mengenali kondisi kesehatan seseorang dengan melakukan pendekripsi pada kondisi wajah menggunakan citra wajah. Sistem ini dibangun pada komputer personal, kemudian setelah mendapatkan model yang diharapkan, dilakukanlah *Deployment* ke *Android*. Gambar 3.1 menunjukkan bagan umum metodologi sistem.



Gambar 3.1: Bagan Umum Metodologi Sistem

3.2 Collecting Image Datasets

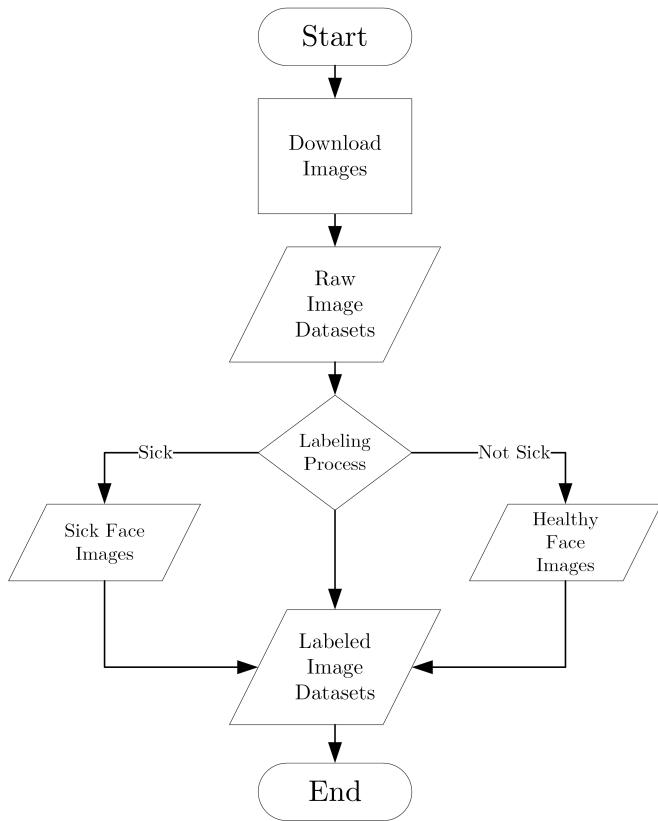
Cara untuk menjadi lebih baik dalam melakukan implementasi algoritma *Deep Learning* pada berbagai macam permasalahan

adalah dengan pembelajaran dan latihan yang mendalam. Pembelajaran atau latihan dapat berupa *image processing* hingga *speech recognition*. Setiap permasalahan memiliki cara dan pendekatan tersendiri yang dapat berbeda dengan yang lain.

Dataset sendiri merupakan kumpulan dari data-data. *Dataset* secara sederhana adalah kumpulan dari data sesuai dengan isi *database* tunggal, atau matriks data statistik tunggal, di mana setiap kolom tabel mewakili variabel tertentu, dan setiap baris sesuai dengan anggota tertentu dari set data yang dipertanyakan. Kumpulan data mencantumkan nilai untuk masing-masing variabel, seperti tinggi dan berat objek, untuk setiap anggota kumpulan data. Setiap nilai dikenal sebagai datum. Kumpulan data dapat terdiri dari data untuk satu atau lebih anggota, sesuai dengan jumlah baris.

Contoh dari *dataset* dapat berupa gambar[50], suara, maupun video. Pada tugas akhir ini, *dataset* yang digunakan berupa gambar yang terbagi menjadi dua kelas, yaitu gambar wajah orang sakit dan gambar wajah orang tidak sakit. Berikut adalah beberapa contoh dari dataset gambar wajah orang sakit dan tidak sakit.

Pada tugas akhir ini, *dataset* yang digunakan dibuat dengan cara *manual*, dengan mengumpulkan berbagai macam gambar baik itu wajah orang sakit maupun wajah orang tidak sakit. Gambar yang dipilih dan dikumpulkan harus berasal dari sumber yang terpercaya, seperti *article*, *paper*, atau *journal*. Setelah mencari referensi dari berbagai sumber, didapatkan hasil bahwa jumlah gambar wajah orang sakit lebih sedikit daripada jumlah gambar wajah orang tidak sakit. Hal ini disebabkan karena wajah merupakan privasi dari seseorang, sehingga penggunaan dari *dataset* ini sangat terbatas. Berbeda hal nya dengan gambar wajah orang tidak sakit yang banyak ditemukan dari berbagai referensi. Salah satu referensi *Dataset* untuk gambar wajah orang tidak sakit adalah *Labeled Faces in the Wild Home Dataset*. Setelah keseluruhan dataset terkumpul, langkah selanjutnya adalah melakukan pengecekan ulang terhadap gambar yang akan digunakan untuk proses selanjutnya. Setelah memastikan bahwasannya gambar tersebut sudah sesuai dengan labelnya, maka dataset gambar tersebut bisa diproses di tahap selanjutnya. Gambar 3.2 menunjukkan diagram alir pada tahapan *collecting image datasets*.



Gambar 3.2: Diagram alir pada proses *collecting image datasets*

3.3 Data Splitting



Gambar 3.3: Visualisasi dari *data splitting*

Dalam *machine learning*, sebelum menjalankan algoritma apapun ke dalam *dataset*, maka *dataset* perlu dibagi menjadi tiga set seperti yang dapat dilihat pada gambar 3.3, yaitu :

1. *Training Sets*

Training Sets merupakan *subset* dari *dataset* yang digunakan untuk melatih model. Pada umumnya, *training sets* menggunakan 80 % dari jumlah keseluruhan *dataset*. Semakin kecil persentase dari *training sets*, maka estimasi parameter akan memiliki varian yang lebih besar.

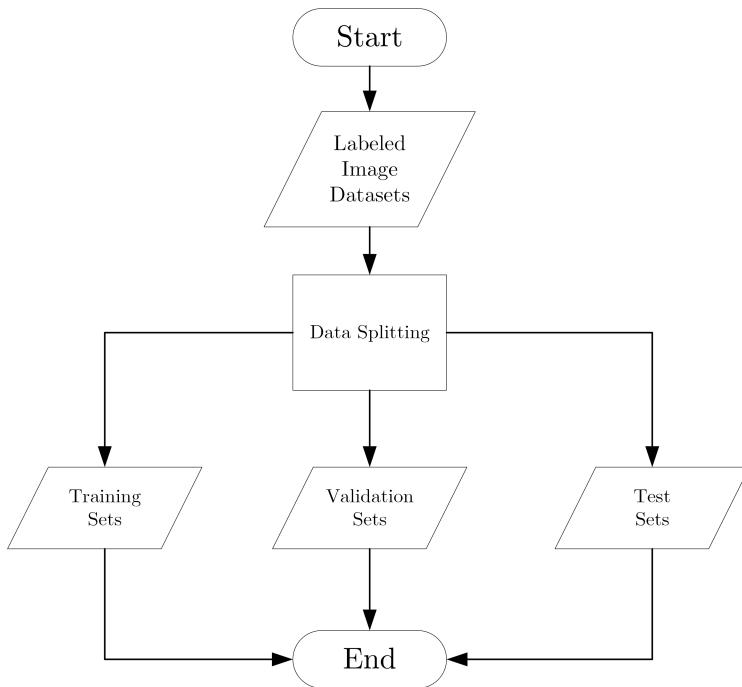
2. *Validation Sets*

Validation Sets merupakan *subset* yang terpisah dari *training sets* yang digunakan dalam melakukan validasi. Fungsi dari *validation sets* adalah untuk mengevaluasi hasil dari *training sets*. Kemudian, akan digunakan *test sets* untuk memeriksa kembali evaluasi tersebut setelah model telah lulus atau melalui tahapan validasi. Pada umumnya, *validation sets* menggunakan 10 % dari jumlah keseluruhan *dataset*.

3. *Test Sets*

Test Sets merupakan *subset* dari *dataset* yang digunakan untuk menguji sebuah model setelah model tersebut melalui tahapan validasi oleh *validation sets*. Pada umumnya, *test set* menggunakan 10 % dari jumlah keseluruhan *dataset*. Semakin kecil persentase dari *test sets*, maka performa statistik dari model akan memiliki varian yang lebih besar.

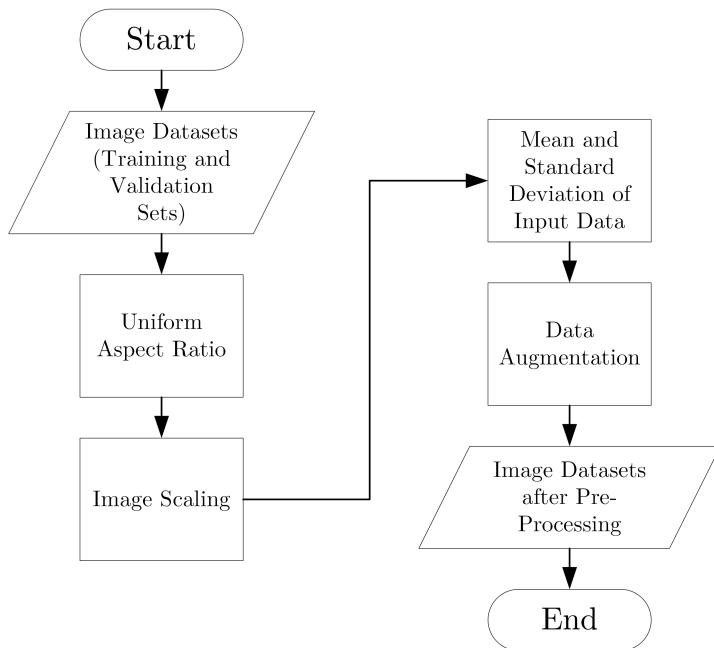
Gambar 3.4 menunjukkan diagram alir dari proses *data splitting*.



Gambar 3.4: Diagram alir proses *data splitting*

3.4 Pre-processing

Pada tahap *pre-processing*, *dataset* yang terdiri dari gambar wajah orang sakit maupun tidak sakit akan mengalami penyesuaian atau pengaturan sebelum masuk ke tahap selanjutnya yaitu *training data*[51]. *Labeled Faces in the Wild* adalah database yang pada awalnya dirancang untuk mempelajari masalah face recognition. Dataset ini berisi lebih dari 13.000 gambar wajah yang dikumpulkan dari website dan setiap wajah telah diberi label dengan nama orang yang digambarkan. Dataset gambar ini berisi gambar wajah orang sehat dengan pengaturan cahaya dan rotasi yang berbeda. Ada sejumlah langkah *pre-processing* yang akan dilakukan sebelum menggunakan pada tahapan selanjutnya. Gambar 3.5 berikut merupakan diagram alir tahapan *pre-processing*.

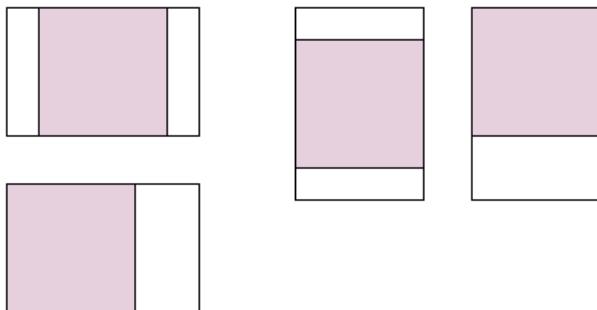


Gambar 3.5: Diagram alir proses *pre-processing*

Berikut adalah beberapa langkah *pre-processing* yang digunakan :

1. *Uniform Aspect Ratio*

Salah satu langkah pertama adalah memastikan bahwa gambar dari *dataset* tersebut memiliki ukuran dan rasio aspek yang sama. Sebagian besar model dari *neural network* mengasumsikan gambar *input* berbentuk persegi, yang berarti bahwa setiap gambar perlu diperiksa apakah gambar tersebut persegi atau tidak. Jika gambar tersebut bukan merupakan persegi, maka akan dilakukan proses *cropping* pada gambar tersebut dengan tepat. *Cropping* dapat dilakukan untuk memilih bagian persegi dari gambar. Pada saat proses *cropping* seperti yang ditunjukkan pada gambar 3.6, gambar yang diambil adalah ada bagian tengah gambar.



Gambar 3.6: Proses *cropping* pada gambar

2. *Image Scaling*

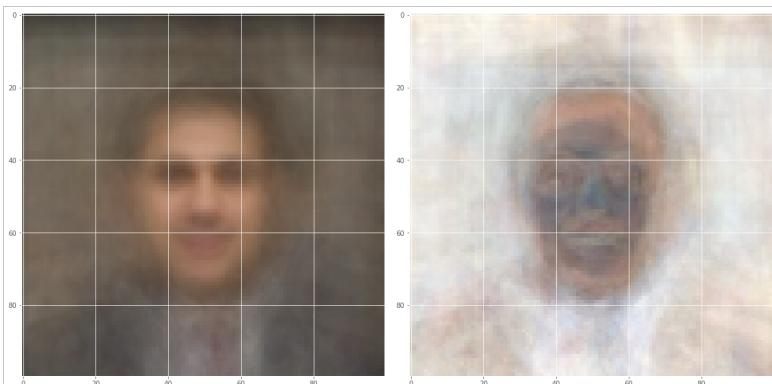


Gambar 3.7: Proses *image scaling* pada gambar

Setelah memastikan bahwa semua gambar memiliki rasio aspek berbentuk persegi atau memiliki beberapa aspek rasio yang telah ditentukan, langkah selanjutnya adalah untuk melakukan *image scaling* seperti yang ditunjukkan pada gambar 3.7 dengan tepat. Pada tugas akhir ini, ukuran dari lebar dan tinggi dari setiap gambar adalah 480 px. Selanjutnya, untuk setiap gambar dengan aspek rasio persegi yang ukuran lebar dan tingginya tidak sama, akan dilakukan *image scaling* untuk memastikan keseluruhan gambar dari

dataset memiliki ukuran lebar dan tinggi yang sama yaitu 480 px. Ada beberapa teknik dalam melakukan *image scaling*, yaitu dengan menggunakan *OpenCV library* maupun dengan menggunakan *software* pengolah gambar lainnya. Hasil dari *image scaling* adalah keseluruhan dari ukuran gambar pada *dataset* memiliki lebar dan tinggi yang sama yaitu 480 px.

3. Mean and Standard Deviation of Input Data



Gambar 3.8: Mean (kiri) dan Standard Deviation (kanan)

Tahap selanjutnya adalah untuk melihat *mean image* yang diperoleh dengan mengambil nilai rata-rata untuk setiap piksel di semua contoh pada proses *training data*. Misalnya, *mean image* dari 100 gambar pertama dalam *dataset* dapat dilihat pada Gambar 3.8 bagian kiri. Dapat ditarik kesimpulan bahwa wajah tersebut sedikit sejajar dengan pusat dan memiliki ukuran yang sebanding. Kemudian, nilai dari standar deviasi ditunjukkan pada Gambar 3.8 sebelah kanan. Nilai varians yang lebih tinggi menunjukkan warna yang lebih putih, sehingga dapat dilihat bahwa gambar tersebut sangat bervariasi di bagian batas jika dibandingkan dengan bagian tengah. Kemudian, nilai dari *mean* dan *standard deviation* akan digunakan pada saat proses *training data*.

4. Data Augmentation

Data Augmentation:

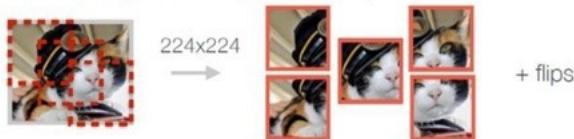
- a. No augmentation (= 1 image)



- b. Flip augmentation (= 2 images)



- c. Crop+Flip augmentation (= 10 images)

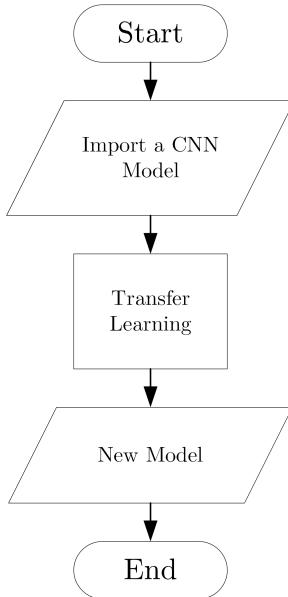


73

Gambar 3.9: Ilustrasi pada proses *data augmentation*

Teknik *pre-processing* selanjutnya adalah melakukan augmentasi pada data yang sudah ada dengan versi gambar yang sudah dilakukan proses augmentasi. Ilustrasi pada proses *data augmentation* dapat dilihat pada gambar 3.9. Penskalaan, rotasi, dan transformasi adalah proses yang sering dilakukan. Hal ini dilakukan untuk meng-explos *neural network* ke berbagai variasi, agar dapat mengenali fitur yang akan dilakukan proses *training*. Hal ini dapat membantu *neural network* untuk dapat mengenali karakteristik yang tidak diinginkan dalam *dataset*.

3.5 Building a CNN Model

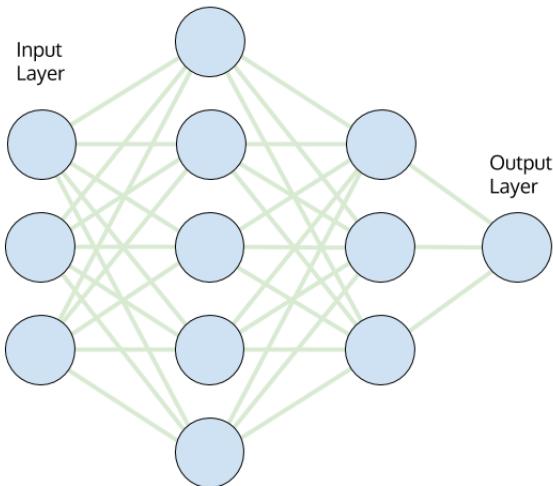


Gambar 3.10: Diagram alir proses *building a CNN Model*

Untuk membuat *CNN Model* yang digunakan untuk *Image Classification*, maka metode yang digunakan adalah dengan menggunakan *Transfer Learning*. *Transfer learning* adalah teknik yang sangat efisien untuk melakukan proses *training* atau *retrain* pada *neural network*. Gambar 3.10 menunjukkan diagram alir proses *building a CNN Model*.

Neural network yang telah dilatih secara penuh mengambil nilai *input* ke dalam *layer* awal dan kemudian secara berurutan memberikan atau mengumpulkan informasi ini ke depan (ambil secara bersamaan mengubahnya) hingga beberapa *layer* kedua hingga terakhir telah membangun representasi tingkat tinggi dari *input* yang dapat dengan mudah ditransformasikan menjadi nilai akhir. Proses pelatihan dari model secara penuh melibatkan pengoptimalan bobot dan bias yang digunakan pada setiap koneksi, yang berlabel

warna hijau pada gambar 3.11 berikut.



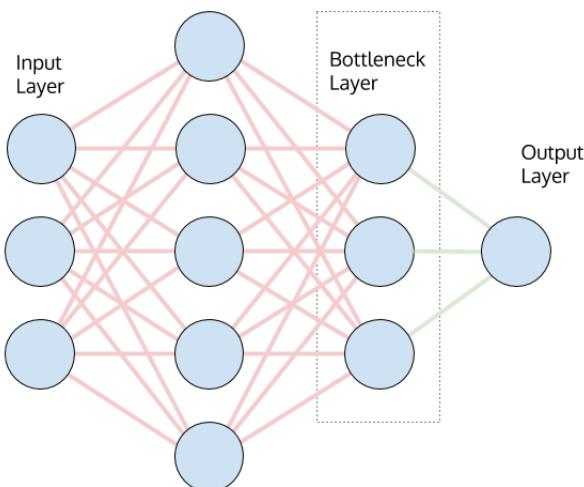
Gambar 3.11: Arsitektur model untuk *standard neural network*, dengan warna hijau menunjukkan proses *training* untuk semua bobot dan bias.

Layer kedua hingga terakhir disebut *bottleneck layer*. *Bottleneck layer* mendorong nilai dalam model regresi, atau probabilitas *softmax* dalam model klasifikasi, ke dalam *final network layer*.

Dalam *transfer learning*, proses akan dimulai dengan bobot yang sudah dilatih sebelumnya untuk seluruh *network*. Kemudian, akan dilakukan perbaikan bobot hingga ke *layer* terakhir dan membiarkan bobot dalam *layer* tersebut berubah saat melakukan proses pembelajaran pada data baru. Seperti yang terlihat pada gambar 3.12, koneksi akan tetap dijaga agar koneksi yang ditunjukkan dengan warna merah tetap, dan sekarang hanya melatih ulang *layer* terakhir dengan koneksi berwarna hijau. *Transfer learning* memberikan dua keuntungan, yaitu :

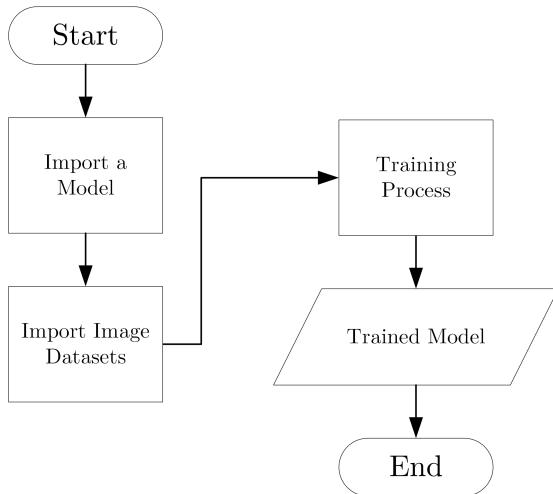
1. *Training* pada data baru memakan waktu yang lebih cepat daripada memulai dari awal.
2. Masalah dapat dipecahkan dengan menggunakan *training data* yang lebih sedikit daripada membangun model dari awal.

Dengan hanya melakukan *retraining* pada *layer* terakhir, optimasi komputasi dapat dilakukan dengan optimasi komputasi yang jauh lebih murah. Hal ini sangat kontras dengan *open source models* seperti *Inception-v3* yang berisi 25 juta parameter dan dilatih dengan menggunakan *hardware* terbaik di kelasnya. Hasilnya, *network* ini memiliki parameter yang tepat dan *bottleneck layer* dengan representasi data *input* yang sangat dioptimalkan. Meskipun kemungkinan akan mengalami kesulitan untuk melatih model berkinerja tinggi dari awal dengan sumber daya komputasi dan data yang sangat terbatas, *transfer learning* dapat dilakukan untuk memecahkan permasalahan yang akan dihadapi.



Gambar 3.12: Arsitektur model untuk *transfer-learning neural network model*, dengan warna merah menunjukkan bobot dan bias yang tetap, sedangkan warna hijau menunjukkan proses *training* pada bobot dan bias di *final layer*.

3.6 Training Process



Gambar 3.13: Diagram alir *training process*

Setelah *network model* disusun dan *dataset* sudah melalui tahap *pre-processing*, maka *network* tersebut siap untuk melakukan proses *training*. Diagram alir *training process* ditunjukkan pada gambar 3.13. Untuk memulai proses ini, bobot awal dipilih secara acak. Kemudian, proses *learning* atau *training* dimulai. Pada tahap ini, akan digunakan *dataset* untuk secara bertahap dapat meningkatkan kemampuan model yang digunakan untuk memprediksi apakah citra wajah yang diberikan adalah sakit atau tidak sakit.

Secara khusus, rumus untuk garis lurus ditunjukkan pada persamaan 3.6.1

$$y = mx + b \quad (3.6.1)$$

di mana x adalah *input*, m adalah kemiringan garis itu, dan b adalah *y-intercept*, dan y adalah nilai garis pada posisi x . Nilai-nilai yang dimiliki untuk disesuaikan atau dilatih adalah m dan b . Tidak ada cara lain untuk mempengaruhi posisi garis, karena satu-satunya variabel lain adalah x yang berarti *input* dan y yang berarti *output*.

Dalam *machine learning*, ada banyak hal karena mungkin ada

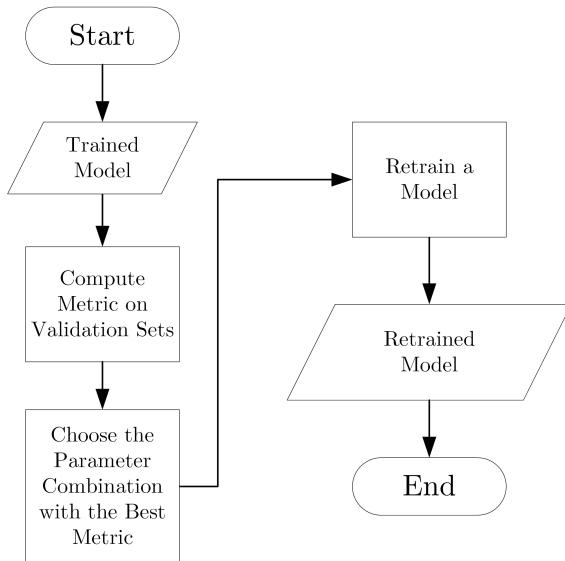
banyak fitur. Pengumpulan nilai-nilai m ini biasanya dibentuk menjadi sebuah matriks, yang akan ditunjukkan sebagai W , untuk matriks bobot. Demikian untuk b juga akan diatur dan menyebutnya dengan bias.

$$Weights = \begin{bmatrix} m_{1,1} & m_{1,2} \\ m_{2,1} & m_{2,2} \\ m_{3,1} & m_{3,2} \end{bmatrix}$$

$$Biases = \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \\ b_{3,1} & b_{3,2} \end{bmatrix}$$

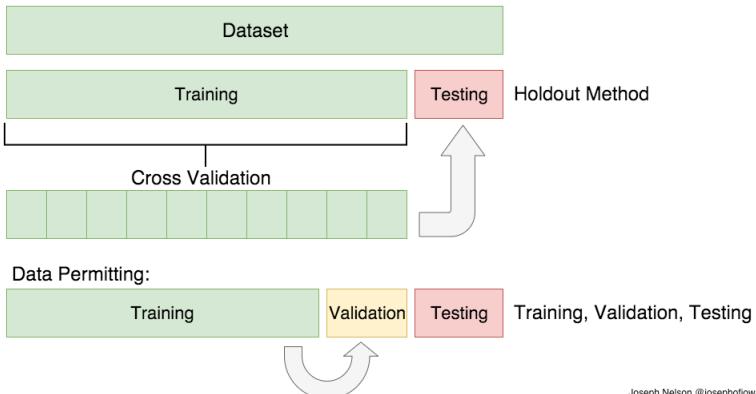
Training process menginisialisasi beberapa nilai acak untuk label wajah dan tidak sakit dari model, lalu memprediksi *output* dengan nilai-nilai itu, kemudian membandingkannya dengan prediksi model dan kemudian menyesuaikan nilai-nilai sehingga akan cocok dengan prediksi yang dibuat sebelumnya. Nilai dalam W dan b dapat disesuaikan sehingga akan memiliki prediksi yang lebih benar. Proses ini kemudian berulang dan setiap siklus pembaruan disebut dengan satu *training step*.

3.7 Validation Process



Gambar 3.14: Diagram alir *validation process*

Setelah proses pelatihan selesai, langkah selanjutnya adalah untuk memeriksa apakah model yang telah dilatih apakah cukup baik dengan menggunakan teknik validasi. Gambar 3.14 menunjukkan diagram alir proses validasi. Pada proses inilah, *dataset* yang telah dipisahkan pada proses sebelumnya akan berperan. Evaluasi memungkinkan pengujian model terhadap data yang belum pernah dilihat dan digunakan untuk pelatihan dan dimaksudkan untuk mewakili bagaimana model dapat menyelesaikan permasalahan yang akan diselesaikan. Tahapan pada validasi akan membantu untuk menemukan parameter terbaik untuk model prediktif dan mencegah dari *overfitting*.



Gambar 3.15: Visualisasi dari *Train/Test split* dan *Cross Validation*.

Cross validation atau merupakan teknik pengujian *out-of-sample* adalah salah satu dari berbagai teknik validasi model yang serupa untuk menilai bagaimana hasil analisis statistik akan digeneralisasi ke *dataset* yang independen. Visualisasi dari *Train/Test split* dan *Cross Validation* ditunjukkan pada gambar 3.15. Hal ini terutama digunakan dalam pengaturan di mana tujuannya adalah prediksi dan memperkirakan seberapa akurat model prodeksi tersebut jika dibandingkan dengan kondisi pada saat di dunia nyata. Dalam masalah prediksi, suatu model biasanya diberikan *dataset* dari data yang tidak diketahui (atau data yang terlihat pertama kali) dibandingkan dengan model yang diuji (disebut dengan *validation set* atau *testing set*). Tujuan dari *cross validation* adalah untuk menguji kemampuan model untuk memprediksi data baru yang tidak digunakan dalam memperkirakannya, untuk menandai masalah seperti *overfitting* atau *selection bias* dan untuk memberikan wawasan tentang bagaimana model akan digeneralisasi ke suatu *dataset* independen atau dataset yang tidak dikenal, misalnya permasalahan dari dunia nyata.

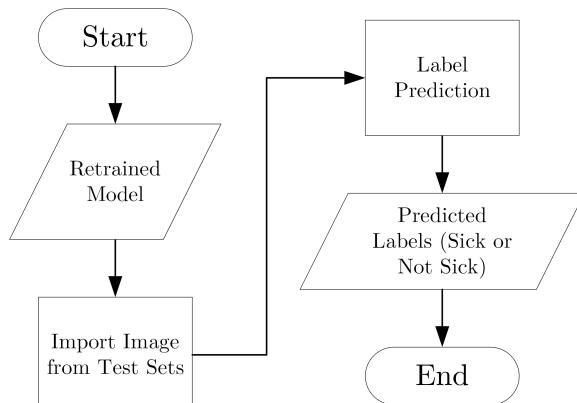
Pada tugas akhir ini, metode dari *cross validation* yang digunakan adalah metode *hold-out*. Dalam metode hold-out, secara acak akan ditetapkan dan ditentukan titik data ke dua set, yaitu d_0 dan d_1 , biasanya disebut dengan *training set* dan *test set*. Ukuran dari *training set* biasanya lebih besar daripada *test set*. Proses

yang dilakukan adalah dengan melatih (membuat model) dan juga melakukan pengujian (mengevaluasi kinerjanya).

Dalam *cross validation*, hasil dari beberapa pengujian model kemudian dirata-rata. Pada metode *hold-out*, secara terpisah, melibatkan satu kali proses. Hal ini harus dijalankan dengan berhati-hati, karena tanpa melakukan rata-rata dari beberapa pengujian, satu kesalahan dapat menyebabkan hasil yang tidak valid. Indikator akurasi prediktif (F^*) akan cenderung tidak stabil karena tidak akan dihaluskan oleh beberapa iterasi. Demikian pula, indikator peran spesifik yang digunakan oleh berbagai variabel prediktor, seperti nilai koefisien regresi akan cenderung tidak stabil. Metode *hold-out* dapat disebut sebagai jenis yang paling sederhana dari *cross-validation*.

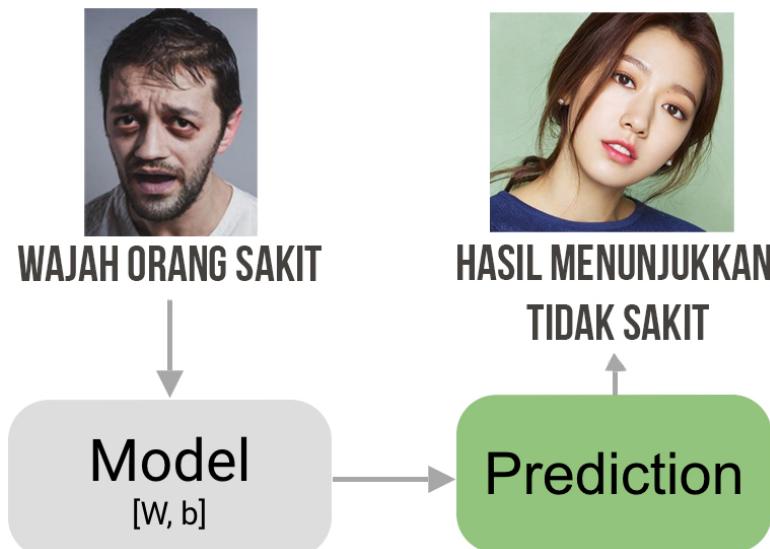
Metode *hold-out* sangat baik untuk digunakan ketika *dataset* yang dimiliki sangat besar, waktu yang terbatas, serta kemampuan komputasi dari *hardware* yang juga terbatas. Hal yang perlu diingat bahwa karena *cross validation* menggunakan beberapa *train/test split test*, maka akan membutuhkan lebih banyak daya komputasi dan waktu yang digunakan daripada menggunakan metode *hold-out*.

3.8 Testing Process



Gambar 3.16: Diagram alir *testing process*

Machine Learning pada dasarnya adalah menggunakan data untuk menjawab pertanyaan. Jadi, *testing data* atau melakukan prediksi adalah langkah terakhir untuk menjawab pertanyaan. Tahap ini adalah titik di mana nilai dari *machine learning* direalisasikan. Pada proses ini, model dapat digunakan untuk memprediksi hasil dari apa yang diinginkan. Diagram alir *testing process* ditunjukkan pada gambar 3.16. Sedangkan tahapan melakukan prediksi dari *test set* ditunjukkan pada gambar 3.17



Gambar 3.17: Tahapan melakukan prediksi dari *test set*.

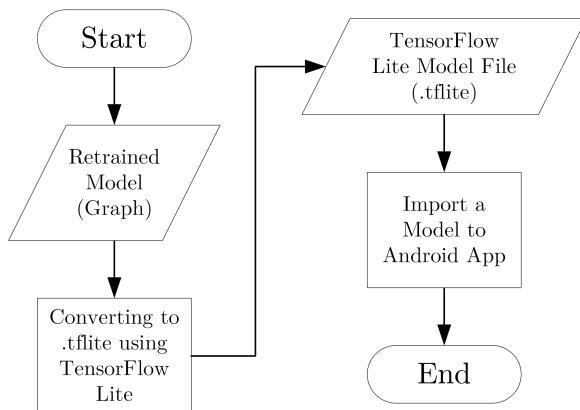
Untuk dapat dikatakan sebuah model yang baik, model tersebut harus memiliki kriteria sebagai berikut :

1. Model yang memiliki performa dan akurasi terbaik pada *test set*, dalam hal ini adalah kriteria dari performa.
2. Model yang dapat melakukan kinerja dengan baik di berbagai metrik kinerja, dalam hal ini adalah kriteria dari ketangguhan atau kekokohan.

- Model yang memiliki skor terbaik pada tahap *cross validation*, dalam hal ini adalah kriteria dari konsistensi.
- Model yang dapat menyelesaikan permasalahan yang terjadi di dunia nyata, dalam hal ini adalah kriteria dari fungsi atau kebermanfaatan.

3.9 Deploying Model to Android

Setelah mendapatkan model yang tepat untuk melakukan *image classification*, maka langkah selanjutnya adalah melakukan *deploying* ke *Android*. Pada tahap ini, akan digunakan *TensorFlow Lite* sebagai *library* yang akan digunakan pada *Android platform*. Gambar 3.18 menunjukkan diagram alir dari tahapan *deploying model to Android*.



Gambar 3.18: Diagram alir proses *deploying model to Android*

Alur kerja pada tahap ini adalah :

- Pick a model*

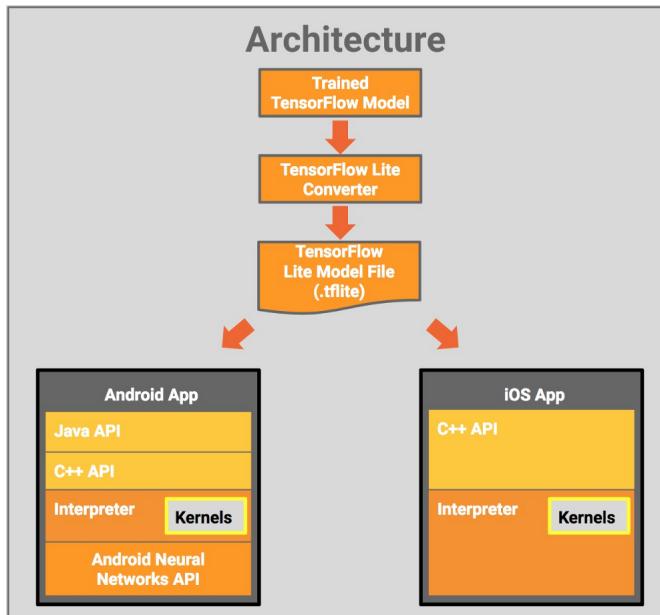
Hasil dari proses *training* dengan menggunakan *TensorFlow* akan menghasilkan file grafik dengan ekstensi *.pb. File grafik ini digunakan untuk melakukan *image classification* pada *Android*.

- Convert the model*

Langkah selanjutnya adalah merubah file grafik berekstensi *.pb ke dalam format *TensorFlow Lite* yang berekstensi *.tflite atau *.lite untuk dapat digunakan di *Android*.

3. Deploy to mobile devices

Kemudian, model tersebut dijalankan dengan menggunakan *TensorFlow Lite interpreter*, yaitu *APIs* untuk membaca dan menjalankan file dari *TensorFlow Lite* yang sudah didukung dengan berbagai macam bahasa pemrograman. Pada tugas akhir ini, bahasa pemrograman yang digunakan untuk membuat aplikasi *Android* adalah *Kotlin*. Arsitektur *TensorFlow Lite* dapat dilihat pada gambar 3.19.



Gambar 3.19: Arsitektur *TensorFlow Lite*.

BAB 4

PENGUJIAN DAN ANALISA

Pada bab ini dipaparkan hasil pengujian serta analisa dari desain sistem dan implementasi. Pengujian dilakukan guna mengetahui performa dari model yang telah dibuat, mengetahui performa dari aplikasi pada platform *Android*, dan mengujinya pada orang sakit. Pengujian dilakukan dengan membagi menjadi beberapa bagian yaitu sebagai berikut :

1. Pengujian jenis model
2. Pengujian performa aplikasi
3. Pengujian pada orang sakit

4.1 Pengujian Jenis Model

Pengujian pada jenis model bertujuan untuk mengetahui performa dan akurasi dari setiap model yang dihasilkan.

4.1.1 Inception-v3

Spesifikasi model *Inception-v3* ditunjukkan pada tabel 4.1.

Spesifikasi Model Inception-v3	
Bottleneck Tensor Name	pool_3/_reshape
Bottleneck Tensor Size	2048
Input Width	299
Input Height	299
Input Depth	3
Resized Input Tensor Name	ResizeBilinear
Input Mean	128
Input Standard Deviation	128

Tabel 4.1: Tabel spesifikasi dari model *Inception-v3*

Setelah dilakukan serangkaian proses *training* pada model *Inception-v3*, didapatkan hasil akhir yaitu *output file size* sebesar 83,3 MB dan nilai dari *final test accuracy* sebesar 99,0 %.

4.1.2 Inception-v4

Spesifikasi model *Inception-v4* ditunjukkan pada tabel 4.2.

Spesifikasi Model Inception-v4	
Bottleneck Tensor Name	InceptionV4/Logits/MatMul
Bottleneck Tensor Size	1001
Input Width	299
Input Height	299
Input Depth	3
Resized Input Tensor Name	InputImage
Input Mean	128
Input Standard Deviation	128

Tabel 4.2: Tabel spesifikasi dari model *Inception-v4*

Setelah dilakukan serangkaian proses *training* pada model *Inception-v3*, didapatkan hasil akhir yaitu *output file size* sebesar 163 MB dan nilai dari *final test accuracy* sebesar 97,9 %.

4.1.3 Inception-ResNet-v2

Spesifikasi model *Inception-ResNet-v2* ditunjukkan pada tabel 4.3.

Spesifikasi Model Inception-ResNet-v2	
Bottleneck Tensor Name	InceptionResnetV2/Logits/MatMul
Bottleneck Tensor Size	1001
Input Width	299
Input Height	299
Input Depth	3
Resized Input Tensor Name	InputImage
Input Mean	128
Input Standard Deviation	128

Tabel 4.3: Tabel spesifikasi dari model *Inception-ResNet-v2*

Setelah dilakukan serangkaian proses *training* pada model *Inception-ResNet-v2*, didapatkan hasil akhir yaitu *output file size* sebesar 213 MB dan nilai dari *final test accuracy* sebesar 97,6 %.

4.1.4 ResNet-v2-152

Spesifikasi model *ResNet-v2-152* ditunjukkan pada tabel 4.4.

Spesifikasi Model ResNet-v2-152	
Bottleneck Tensor Name	input/BottleneckInputPlaceholder
Bottleneck Tensor Size	2048
Input Width	224
Input Height	224
Input Depth	3
Resized Input Tensor Name	Placeholder
Input Mean	128
Input Standard Deviation	128

Tabel 4.4: Tabel spesifikasi dari model *ResNet-v2-152*

Setelah dilakukan serangkaian proses *training* pada model *ResNet-v2-152*, didapatkan hasil akhir yaitu *output file size* sebesar 223 MB dan nilai dari *final test accuracy* sebesar 95,4 %.

4.1.5 MobileNetV1

Spesifikasi model *MobileNetV1* ditunjukkan pada tabel 4.5.

Spesifikasi Model MobileNetV1 1.4 224	
Bottleneck Tensor Name	MobileNetV1/Predictions/Reshape
Bottleneck Tensor Size	1001
Input Width	224
Input Height	224
Input Depth	3
Resized Input Tensor Name	input
Input Mean	127,5
Input Standard Deviation	127,5

Tabel 4.5: Tabel spesifikasi dari model *MobileNetV1 1.4 224*

Setelah dilakukan serangkaian proses *training* pada model *MobileNetV1 1.14 224*, didapatkan hasil akhir yaitu *output file size* sebesar 16,3 MB dan nilai dari *final test accuracy* sebesar 98,4 %.

4.1.6 MobileNetV2

Spesifikasi model *MobileNetV2* ditunjukkan pada tabel 4.6.

Spesifikasi Model MobileNetV2 1.4 224	
Bottleneck Tensor Name	MobilenetV2/Predictions/Reshape_1
Bottleneck Tensor Size	1001
Input Width	224
Input Height	224
Input Depth	3
Resized Input Tensor Name	input
Input Mean	128
Input Standard Deviation	128

Tabel 4.6: Tabel spesifikasi dari model *MobileNetV2 1.4 224*

Setelah dilakukan serangkaian proses *training* pada model *MobileNetV2 1.14 224*, didapatkan hasil akhir yaitu *output file size* sebesar 23,3 MB dan nilai dari *final test accuracy* sebesar 91,5 %.

4.1.7 MobileFaceNets

Spesifikasi model *MobileFaceNets* ditunjukkan pada tabel 4.7.

Spesifikasi Model MobileFaceNets	
Bottleneck Tensor Name	MobileFaceNet/Logits/SpatialSqueeze
Bottleneck Tensor Size	128
Input Width	112
Input Height	112
Input Depth	3
Resized Input Tensor Name	img_inputs
Input Mean	128
Input Standard Deviation	128

Tabel 4.7: Tabel spesifikasi dari model *MobileFaceNets*

Setelah dilakukan serangkaian proses *training* pada model *MobileFaceNets*, didapatkan hasil akhir yaitu *output file size* sebesar 5,68 MB dan nilai dari *final test accuracy* sebesar 93,8 %.

4.1.8 SqueezeNet

Spesifikasi model *SqueezeNet* ditunjukkan pada tabel 4.8.

Spesifikasi Model SqueezeNet	
Bottleneck Tensor Name	global_average_pooling2d_1/Mean
Bottleneck Tensor Size	1000
Input Width	227
Input Height	227
Input Depth	3
Resized Input Tensor Name	input_1
Input Mean	128
Input Standard Deviation	128

Tabel 4.8: Tabel spesifikasi dari model *SqueezeNet*

Setelah dilakukan serangkaian proses *training* pada model *SqueezeNet*, didapatkan hasil akhir yaitu *output file size* sebesar 4,74 MB dan nilai dari *final test accuracy* sebesar 48,2 %.

4.1.9 NasNet Mobile

Spesifikasi model *NasNet Mobile* ditunjukkan pada tabel 4.9.

Spesifikasi Model NasNet Mobile	
Bottleneck Tensor Name	input/BottleneckInputPlaceholder
Bottleneck Tensor Size	1056
Input Width	224
Input Height	224
Input Depth	3
Resized Input Tensor Name	Placeholder
Input Mean	128
Input Standard Deviation	128

Tabel 4.9: Tabel spesifikasi dari model *NasNet Mobile*

Setelah dilakukan serangkaian proses *training* pada model *NasNet Mobile*, didapatkan hasil akhir yaitu *output file size* sebesar 16,9 MB dan nilai dari *final test accuracy* sebesar 93 %.

4.1.10 DenseNet

Spesifikasi model *DenseNet* ditunjukkan pada tabel 4.10.

Spesifikasi Model DenseNet	
Bottleneck Tensor Name	densenet169/predictions/Reshape
Bottleneck Tensor Size	1000
Input Width	224
Input Height	224
Input Depth	3
Resized Input Tensor Name	input
Input Mean	128
Input Standard Deviation	128

Tabel 4.10: Tabel spesifikasi dari model *DenseNet*

Setelah dilakukan serangkaian proses *training* pada model *DenseNet*, didapatkan hasil akhir yaitu *output file size* sebesar 55 MB dan nilai dari *final test accuracy* sebesar 90 %.

4.1.11 ShuffleNetV2

Spesifikasi model *ShuffleNetV2* ditunjukkan pada tabel 4.11.

Spesifikasi Model ShuffleNetV2	
Bottleneck Tensor Name	linear/output
Bottleneck Tensor Size	1000
Input Width	224
Input Height	224
Input Depth	3
Resized Input Tensor Name	input
Input Mean	128
Input Standard Deviation	128

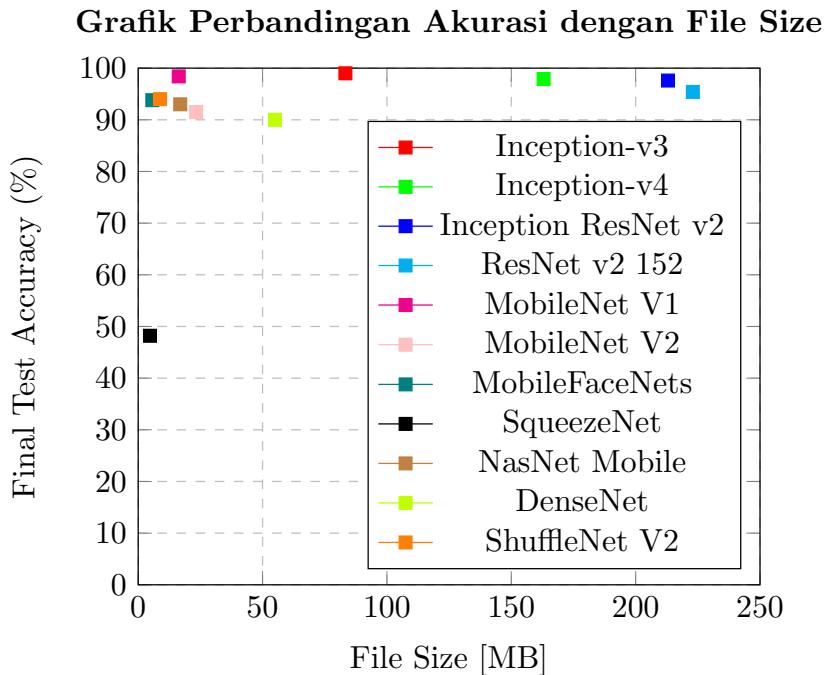
Tabel 4.11: Tabel spesifikasi dari model *ShuffleNetV2*

Setelah dilakukan serangkaian proses *training* pada model *ShuffleNetV2*, didapatkan hasil akhir yaitu *output file size* sebesar 8,75 MB dan nilai dari *final test accuracy* sebesar 94 %.

Tabel 4.12 adalah tabel perbandingan dari *output file size* dan *final test accuracy* dari total 11 model yang diuji. Tahap ini masih merupakan hasil tahap *training*. Sedangkan gambar 4.1 merupakan grafik perbandingan akurasi dengan *file size*. Proses selanjutnya setelah mendapatkan hasil yang ditunjukkan pada setiap model adalah melakukan proses memprediksi atau melakukan *testing data*.

No	Model	Output File Size	Accuracy
1	Inception-v3	83.3 MB	99.0 %
2	Inception-v4	163 MB	97.9 %
3	Inception ResNet v2	213 MB	97.6 %
4	ResNet-v2-152	223 MB	95.4 %
5	MobileNet V1 1.0 224	16.3 MB	98.4 %
6	MobileNet V2 1.4 224	23.3 MB	91.5 %
7	MobileFaceNets	5.68 MB	93.8 %
8	SqueezeNet	4.74 MB	48.2 %
9	Nasnet Mobile	16.9 MB	93.0 %
10	DenseNet	55 MB	90.0 %
11	ShuffleNet V2	8.75 MB	94.0 %

Tabel 4.12: Tabel perbandingan dari berbagai model yang digunakan



Gambar 4.1: Grafik Perbandingan Akurasi dengan File Size

4.1.12 Perbandingan Hasil Prediksi

Setelah mendapatkan hasil pada tabel 4.12 terdapat dua parameter yang digunakan yaitu *output file size* dan *final test accuracy*. Parameter tersebut digunakan untuk melakukan justifikasi dalam pemilihan model yang terbaik. Pada parameter *output file size*, model yang diinginkan adalah model dengan ukuran sekecil mungkin, karena penggunaannya adalah pada *Android*. Sedangkan untuk *final test accuracy*, tingkat akurasi yang diinginkan adalah model dengan tingkat akurasi yang tinggi sehingga kemampuan model tersebut untuk melakukan proses prediksi akan semakin tepat dengan kondisi yang sesuai pada dunia nyata.

Setelah membandingkan parameter *output file size* dan *final test accuracy*, pada tahap selanjutnya akan dilakukan proses membandingkan nilai atau hasil dari prediksi pada setiap model yang telah dibuat. Semakin tinggi tingkat akurasi dan kebenaran dalam prediksi, maka model tersebut sangat bagus dan sesuai dengan permasalahan yang ingin diselesaikan oleh model yang telah dibuat. Tujuan dari tahapan membandingkan ini adalah untuk memilih model yang terbaik dan apakah model yang telah dibuat dapat benar-benar dapat menyelesaikan permasalahan *image classification*. Sehingga, dari hasil percobaan ini dapat menentukan model terbaik yang dapat menyelesaikan permasalahan dan juga telah diuji dengan menggunakan *testing set*.

Hasil Pengujian dengan Model *Inception-v3*

Pada hasil pengujian dengan menggunakan satu gambar orang tidak sakit yang ditunjukkan pada gambar 4.2, didapatkan hasil skor yaitu :

1. tidak sakit (*score* : 0,670)
2. sakit (*score* : 0,330)

Dari hasil tersebut, model *Inception-v3* dapat memprediksi gambar dengan benar, serta *evaluation time* dari model ini sebesar 2,359 detik.



Gambar 4.2: Hasil uji dari model *Inception-v3*

Hasil Pengujian dengan Model *Inception-v4*

Pada hasil pengujian dengan menggunakan satu gambar orang tidak sakit yang ditunjukkan pada gambar 4.3, didapatkan hasil skor yaitu :

1. tidak sakit (*score* : 0,778)
2. sakit (*score* : 0,222)

Dari hasil tersebut, model *Inception-v4* dapat memprediksi gambar dengan benar, serta *evaluation time* dari model ini sebesar 12,309 detik.



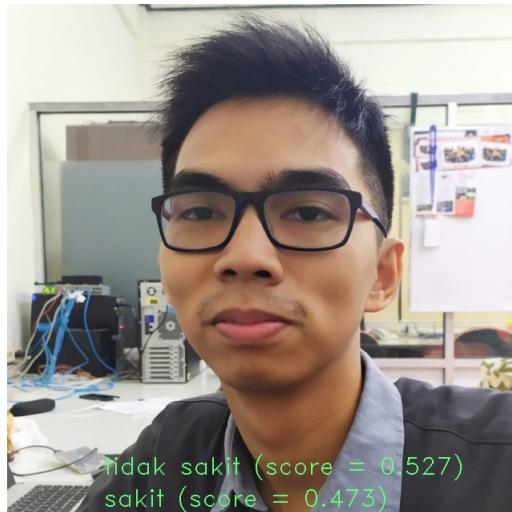
Gambar 4.3: Hasil uji dari model *Inception-v4*

Hasil Pengujian dengan Model *Inception-ResNet-v2*

Pada hasil pengujian dengan menggunakan satu gambar orang tidak sakit yang ditunjukkan pada gambar 4.4, didapatkan hasil skor yaitu :

1. tidak sakit (*score* : 0,527)
2. sakit (*score* : 0,473)

Dari hasil tersebut, model *Inception-ResNet-v2* dapat memprediksi gambar dengan benar, serta *evaluation time* dari model ini sebesar 14,557 detik



Gambar 4.4: Hasil uji dari model *Inception-ResNet-v2*

Hasil Pengujian dengan Model *ResNet-v2-152*

Pada hasil pengujian dengan menggunakan satu gambar orang tidak sakit yang ditunjukkan pada gambar 4.5, didapatkan hasil skor yaitu :

1. tidak sakit (*score* : 0,032)
2. sakit (*score* : 0,968)

Dari hasil tersebut, model *ResNet-v2-152* tidak dapat memprediksi gambar dengan benar, serta *evaluation time* dari model ini sebesar 15,231 detik



Gambar 4.5: Hasil uji dari model *ResNet-v2-152*

Hasil Pengujian dengan Model *MobileNetV1*

Pada hasil pengujian dengan menggunakan satu gambar orang tidak sakit yang ditunjukkan pada gambar 4.6, didapatkan hasil skor yaitu :

1. tidak sakit (*score* : 0,714)
2. sakit (*score* : 0,286)

Dari hasil tersebut, model *MobileNetV1* dapat memprediksi gambar dengan benar, serta *evaluation time* dari model ini sebesar 2,370 detik



Gambar 4.6: Hasil uji dari model *MobileNetV1*

Hasil Pengujian dengan Model *MobileNetV2*



Gambar 4.7: Hasil uji dari model *MobileNetV2*

Pada hasil pengujian dengan menggunakan satu gambar orang tidak sakit yang ditunjukkan pada gambar 4.7, didapatkan hasil skor yaitu :

1. tidak sakit (*score* : 0,877)
2. sakit (*score* : 0,123)

Dari hasil tersebut, model *MobileNetV2* dapat memprediksi gambar dengan benar, serta *evaluation time* dari model ini sebesar 2,915 detik

Hasil Pengujian dengan Model *MobileFaceNets*



Gambar 4.8: Hasil uji dari model *MobileFaceNets*

Pada hasil pengujian dengan menggunakan satu gambar orang tidak sakit yang ditunjukkan pada gambar 4.8, didapatkan hasil skor yaitu :

1. tidak sakit (*score* : 0,804)
2. sakit (*score* : 0,196)

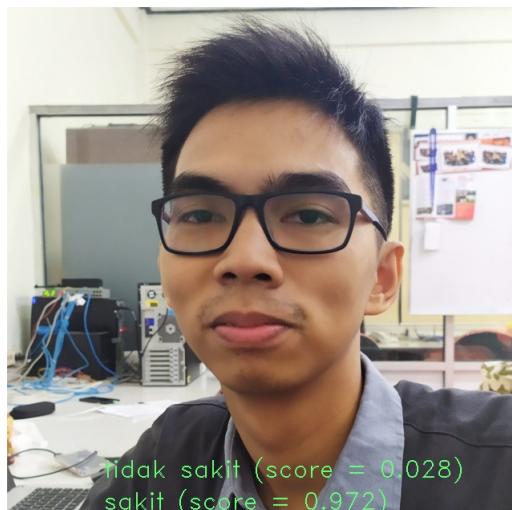
Dari hasil tersebut, model *MobileNetV1* tidak dapat memprediksi gambar dengan benar, serta *evaluation time* dari model ini sebesar 2,746 detik

Hasil Pengujian dengan Model *SqueezeNet*

Pada hasil pengujian dengan menggunakan satu gambar orang tidak sakit yang ditunjukkan pada gambar 4.9, didapatkan hasil skor yaitu :

1. tidak sakit (*score* : 0,028)
2. sakit (*score* : 0,972)

Dari hasil tersebut, model *MobileNetV1* tidak dapat memprediksi gambar dengan benar, serta *evaluation time* dari model ini sebesar 1,576 detik



Gambar 4.9: Hasil uji dari model *SqueezeNet*

Hasil Pengujian dengan Model *NasNet Mobile*

Pada hasil pengujian dengan menggunakan satu gambar orang tidak sakit yang ditunjukkan pada gambar 4.10, didapatkan hasil skor yaitu :

1. tidak sakit (*score* : 0,987)
2. sakit (*score* : 0,013)

Dari hasil tersebut, model *NasNet Mobile* tidak dapat memprediksi gambar dengan benar, serta *evaluation time* dari model ini sebesar 3,343 detik



Gambar 4.10: Hasil uji dari model *NasNet Mobile*

Hasil Pengujian dengan Model *DenseNet*

Pada hasil pengujian dengan menggunakan satu gambar orang tidak sakit yang ditunjukkan pada gambar 4.11, didapatkan hasil skor yaitu :

1. tidak sakit (*score* : 0,971)
2. sakit (*score* : 0,029)

Dari hasil tersebut, model *DenseNet* dapat memprediksi gambar dengan benar, serta *evaluation time* dari model ini sebesar 4,997 detik



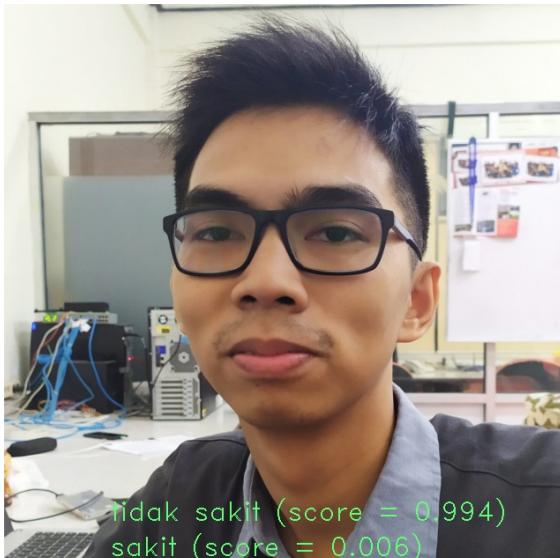
Gambar 4.11: Hasil uji dari model *DenseNet*

Hasil Pengujian dengan Model *ShuffleNetV2*

Pada hasil pengujian dengan menggunakan satu gambar orang tidak sakit yang ditunjukkan pada gambar 4.12, didapatkan hasil skor yaitu :

1. tidak sakit (*score* : 0,994)
2. sakit (*score* : 0,006)

Dari hasil tersebut, model *ShuffleNetV2* dapat memprediksi gambar dengan benar, serta *evaluation time* dari model ini sebesar 2,068 detik



Gambar 4.12: Hasil uji dari model *ShuffleNetV2*

Setelah melakukan *testing* dengan menggunakan gambar pada *testing set* pada keseluruhan model, maka tabel 4.13 menunjukkan ringkasan dari perbandingan hasil *testing* yang telah dilakukan.

Parameter yang akan dibandingkan yaitu *evaluation time*, besar skor dari hasil prediksi, serta benar atau salahnya model tersebut dalam melakukan prediksi pada kondisi nyata.

Pengujian yang ditunjukkan pada tabel 4.12 dan tabel 4.13 dilakukan dengan menggunakan *Laptop* yang memiliki spesifikasi seperti pada tabel 4.14.

No	Model	Evaluation Time	Top-1 Accuracy (%)	Testing Results
1	Inception-v3	2,359 s	67,0 %	Correct
2	Inception-v4	12,309 s	77,8 %	Correct
3	Inception ResNet v2	14,557 s	52,7 %	Correct
4	ResNet-v2-152	15,231 s	3,2 %	Incorrect
5	MobileNet V1 1.0 224	2,370 s	71,4 %	Correct
6	MobileNet V2 1.4 224	2,915 s	87,7 %	Correct
7	MobileFaceNets	2,746 s	80,4 %	Correct
8	SqueezeNet	1,576 s	2,8 %	Incorrect
9	Nasnet Mobile	3,343 s	98,7 %	Correct
10	DenseNet	4,997 s	97,1 %	Correct
11	ShuffleNet V2	2,068 s	99,4 %	Correct

Tabel 4.13: Tabel perbandingan evaluation time, *Top-1 Accuracy*, dan testing results dari berbagai model yang digunakan

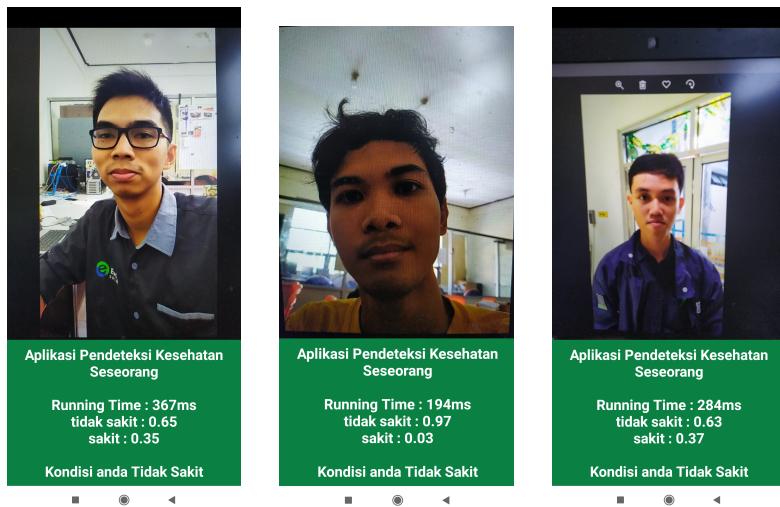
Spesifikasi Laptop ASUS ROG Strix Hero Edition G503VM	
Processor	Intel Core i7-7700HQ
RAM	16 GB DDR4
Storage	256 GB SSD + 1 TB HDD
Graphics Card	NVIDIA GeForce GTX 1060 6 GB
Operating System	Windows 10 Home 64 Bit

Tabel 4.14: Tabel spesifikasi Laptop yang digunakan

4.2 Pengujian Performa Aplikasi

Dari hasil pengujian kepada beberapa orang, maka bisa didapatkan *performance time* dan skor untuk prediksi apakah orang tersebut sakit atau tidak sakit.

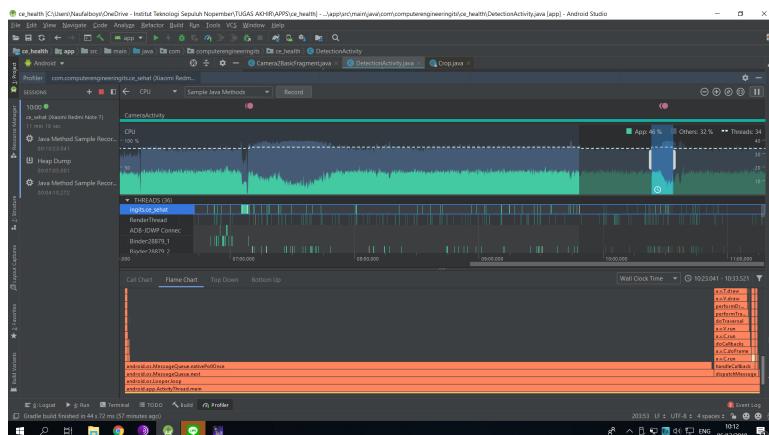
4.2.1 Pengujian Tanpa Menggunakan *Face Detection*



Gambar 4.13: Hasil pengujian pada aplikasi *Android*

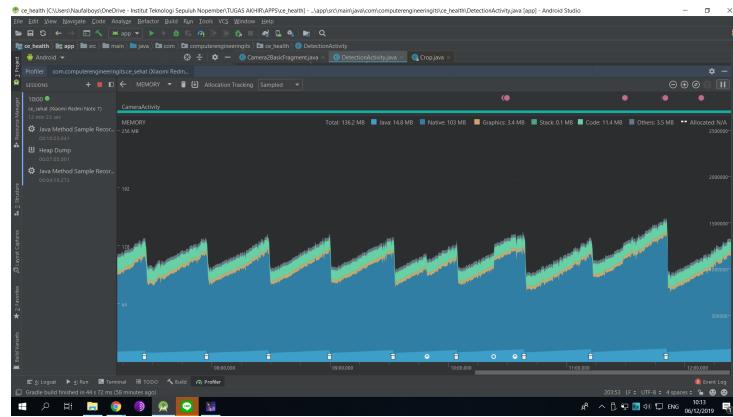
Dari gambar 4.13, dapat dilihat berbagai parameter seperti *running time* dan juga skor atau *confidence score*. Hal ini dapat dipengaruhi oleh beberapa faktor, seperti jenis kelamin, usia, dan juga faktor pencahayaan.

Langkah selanjutnya adalah melakukan pengujian performa aplikasi dengan menggunakan *Android Profiler* untuk menguji *CPU Usage* yang ditunjukkan pada gambar 4.14 . Berikut adalah gambar yang menunjukkan pengujian performa aplikasi untuk menguji *CPU Usage* pada aplikasi.

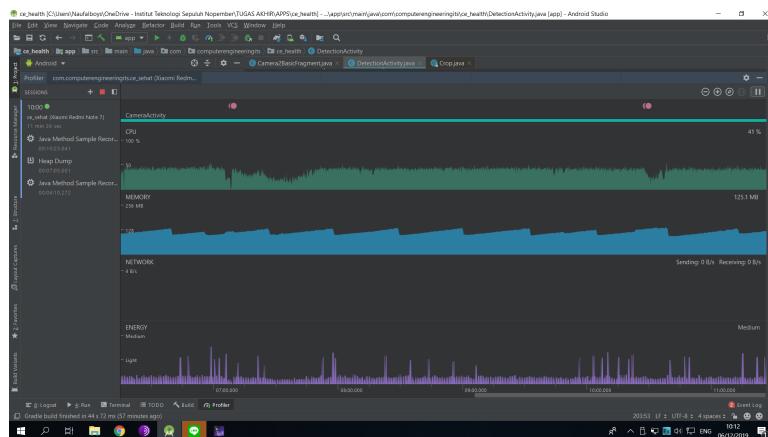


Gambar 4.14: Hasil pengujian performa *CPU Usage* tanpa menggunakan *face detection*

Selain melihat tingkat persentase dari *CPU Usage*, pada tugas akhir ini juga akan dilakukan pengujian performa untuk menguji *Memory Usage* yang dapat dilihat pada gambar 4.15. Berikut adalah gambar yang menunjukkan pengujian performa aplikasi untuk menguji *Memory Usage* pada aplikasi.



Gambar 4.15: Hasil pengujian performa *Memory Usage* tanpa menggunakan *face detection*



Gambar 4.16: Hasil pengujian performa tanpa menggunakan *face detection*

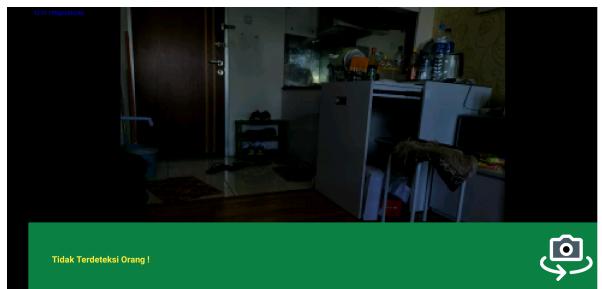
Selain *CPU Usage* dan *Memory Usage*, dilakukan pengujian juga terhadap *Memory Usage* pada aplikasi tersebut. Hasil pengujian performa dengan menggunakan *Android Profiler* ditunjukkan pada gambar 4.16. Dapat dilihat bahwa *CPU Usage*, *Memory*

Usage, maupun *Energy Usage* yang digunakan pada aplikasi tanpa menggunakan *face detection* relatif lebih ringan, sehingga aplikasi ini dapat mencapai *framerate* sebesar 30 fps.

4.2.2 Pengujian Menggunakan *Face Detection*

Pada pengujian dengan menggunakan *Face Detection*, teknik yang digunakan adalah dengan memanfaatkan metode *Haar Cascade Classifier* sehingga kamera dapat mengenali bagian wajah. Jika aplikasi tidak mendeteksi wajah atau wajah yang terdeteksi lebih dari satu, maka aplikasi tidak akan melakukan proses klasifikasi. Sedangkan jika aplikasi mendeteksi satu wajah, maka akan dilakukan proses klasifikasi pada wajah.

Gambar 4.17 menunjukkan tampilan antarmuka jika aplikasi tidak mendeteksi wajah orang. Sedangkan gambar 4.18 menunjukkan tampilan antarmuka jika aplikasi mendeteksi wajah orang.



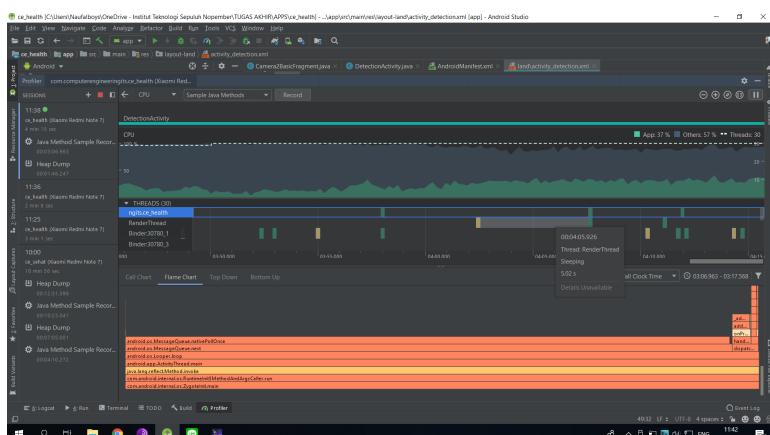
Gambar 4.17: Tampilan aplikasi ketika wajah tidak terdeteksi



Gambar 4.18: Tampilan aplikasi ketika wajah terdeteksi

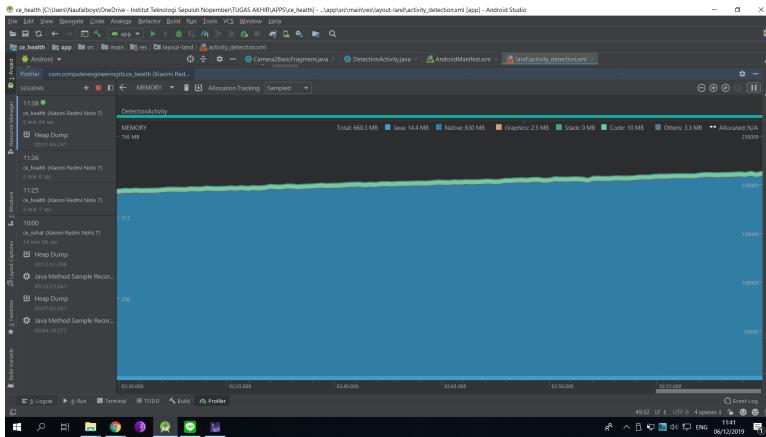
Langkah selanjutnya adalah melakukan pengujian performa aplikasi dengan menggunakan *Android Profiler* untuk menguji *CPU*

Usage. Gambar 4.19 adalah gambar yang menunjukkan pengujian performa aplikasi untuk menguji *CPU Usage* pada aplikasi.

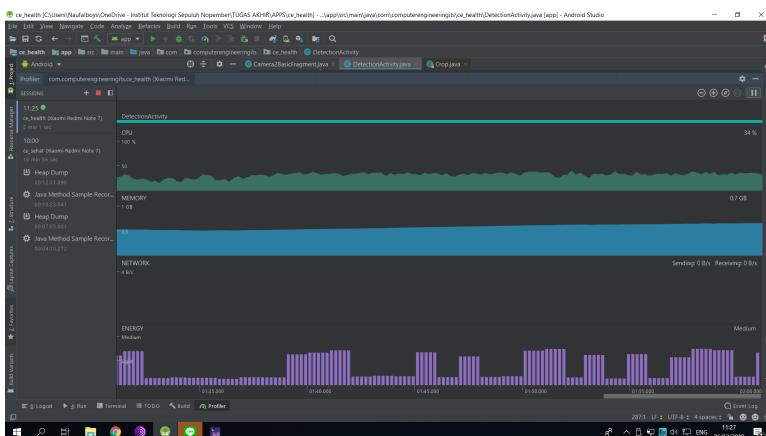


Gambar 4.19: Hasil pengujian performa *CPU Usage* dengan menggunakan *face detection*

Selain melihat tingkat persentase dari *CPU Usage*, pada tugas akhir ini juga akan dilakukan pengujian performa untuk menguji *Memory Usage*. Gambar 4.20 menunjukkan pengujian performa aplikasi untuk menguji *Memory Usage* pada aplikasi.



Gambar 4.20: Hasil pengujian performa *Memory Usage* dengan menggunakan *face detection*



Gambar 4.21: Hasil pengujian performa dengan menggunakan *face detection*

Selain *CPU Usage* dan *Memory Usage*, dilakukan pengujian juga terhadap *Memory Usage* pada aplikasi tersebut. Hasil pengujian performa dengan menggunakan *Android Profiler* ditunjukkan pa-

da gambar 4.21. Dapat dilihat bahwa *CPU Usage*, *Memory Usage*, maupun *Energy Usage* yang digunakan pada aplikasi dengan menggunakan *face detection* relatif lebih berat untuk dijalankan, sehingga aplikasi ini hanya dapat mencapai *framerate* sebesar kurang dari 30 fps ketika tidak ada wajah terdeteksi dan kurang dari 15 fps ketika ada wajah terdeteksi.

4.2.3 Perbandingan Performa Aplikasi

Pada bagian ini, akan dilakukan proses perbandingan dari hasil pengujian tanpa menggunakan *Face Detection* dengan pengujian menggunakan *Face Detection*. Perbandingan yang dilakukan adalah dengan melakukan perbandingan penggunaan *resource* pada *android* seperti *CPU*, *Memory*, dan *Energy*.

No	Metode	<i>CPU Usage</i>	<i>Memory Usage</i>	<i>Energy Usage</i>
1	<i>Dengan Face Detection</i>	41 %	700 MB	Medium
2	<i>Tanpa Face Detection</i>	34 %	125.1 MB	Light

Tabel 4.15: Tabel perbandingan performa aplikasi dengan menggunakan *face detection* dan tanpa *face detection*

Setelah dilakukan pengujian perbandingan performa aplikasi seperti pada tabel 4.15, didapatkan hasil bahwa aplikasi dengan menggunakan algoritma *face detection* memakan *resource* yang lebih besar daripada tanpa menggunakan algoritma *face detection*. Dari hasil pengujian, aplikasi yang menerapkan algoritma *face detection* juga mendapatkan akurasi yang cukup tinggi, namun sisi negatifnya adalah karena penggunaan *resource* yang besar sehingga sering terjadi *frame rate drops* karena hanya mencapai kurang dari 15 *fps*.

4.3 Pengujian Pada Orang Sakit

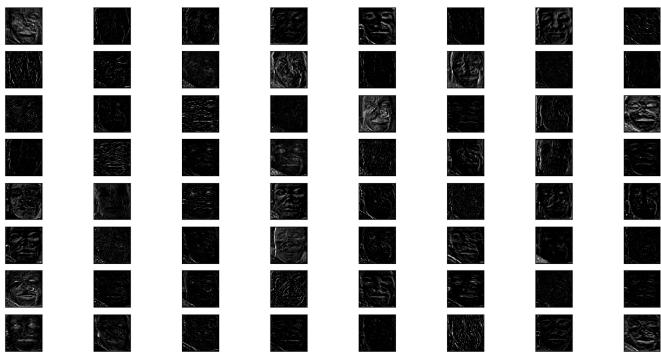
Pada bagian ini, akan dilakukan pengujian kepada orang sakit. Jumlah orang yang akan diuji adalah lima orang dan semuanya sedang dalam kondisi sakit.

4.3.1 Feature Maps Dataset Orang Sakit

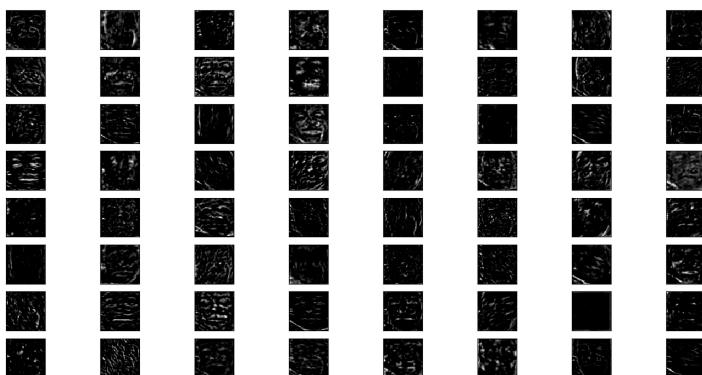
Feature maps dapat juga disebut dengan *activation maps*. Setelah filter diekstraksi dari gambar, filter ini adalah bagian kecil dari gambar yang akan memiliki fitur berbeda. Jumlah filter yang digunakan pada input harus membuat jumlah yang sama dari *feature maps*. Jadi gambar input dengan 6 filter akan memiliki 6 *feature maps*. Gambar 4.22, 4.23, 4.24, dan 4.25 merupakan gambar *feature maps* citra wajah orang sakit pada Stage 1 hingga Stage 4 dengan menggunakan arsitektur *ShuffleNetV2*.



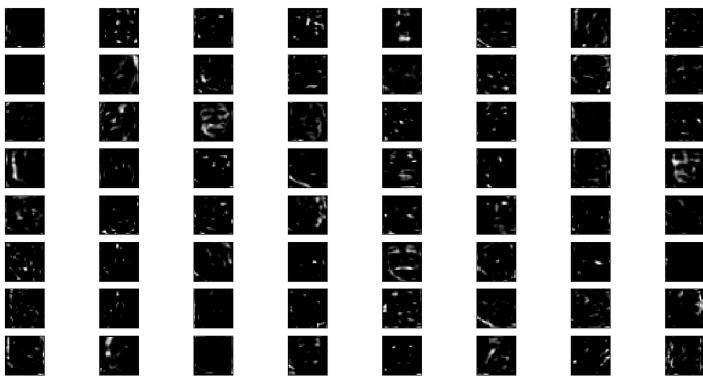
Gambar 4.22: *Feature maps* citra wajah orang sakit pada Stage 1 dengan menggunakan arsitektur *ShuffleNetV2*.



Gambar 4.23: *Feature maps* citra wajah orang sakit pada Stage 2 dengan menggunakan arsitektur *ShuffleNetV2*.



Gambar 4.24: *Feature maps* citra wajah orang sakit pada Stage 3 dengan menggunakan arsitektur *ShuffleNetV2*.



Gambar 4.25: *Feature maps* citra wajah orang sakit pada Stage 4 dengan menggunakan arsitektur *ShuffleNetV2*.

4.3.2 Pengujian Pada Orang Sakit

Pengujian pada orang sakit dilakukan kepada lima orang yang menderita flu batuk, sakit kepala, dan sakit kejang.



Gambar 4.26: Hasil pengujian terhadap tiga orang sakit

Pada gambar 4.26 di atas, gambar sebelah kiri merupakan orang yang terkena sakit kepala dan sakit kejang. Hasil percobaan menunjukkan bahwa orang tersebut sakit dengan nilai *confidence* sebesar 0.8149 atau 81,49 %.

Pada gambar 4.26 di atas, gambar sebelah tengah merupakan

kan orang yang menderita flu. Hasil percobaan menunjukkan bahwa orang tersebut sakit dengan nilai *confidence* sebesar 0.7436 atau 74,36 %.

Pada gambar 4.26 di atas, gambar sebelah kanan merupakan orang yang menderita flu dan batuk. Hasil percobaan menunjukkan bahwa orang tersebut sakit dengan nilai *confidence* sebesar 0.9841 atau 98,41 %.



Gambar 4.27: Hasil pengujian terhadap dua orang sakit

Pada gambar 4.27 di atas, gambar sebelah kiri merupakan orang yang menderita flu. Hasil percobaan menunjukkan bahwa orang tersebut sakit dengan nilai *confidence* sebesar 0.9940 atau 99,40 %.

Pada gambar 4.27 di atas, gambar sebelah kanan merupakan orang yang menderita flu dan batuk. Hasil percobaan menunjukkan bahwa orang tersebut sakit dengan nilai *confidence* sebesar 0.9760 atau 97,60 %.

BAB 5

PENUTUP

5.1 Kesimpulan

Dalam penelitian ini, telah diimplementasikan serangkaian prosedur untuk mendeteksi kondisi kesehatan seseorang melalui citra wajah dengan menggunakan *Deep Convolutional Neural Network*. Berdasarkan hasil pengujian dari model yang dipilih yaitu *ShuffleNetV2*, tingkat akurasi sistem dalam mendeteksi wajah orang sakit atau tidak sakit mencapai 94 % berdasarkan data pada tabel 4.12. Selain itu, model ini juga berhasil melakukan prediksi dengan tepat pada gambar dengan *evaluation time* sebesar 2,068 detik dan mendapatkan *Top-1 Accuracy* sebesar 57,7 %. Model *ShuffleNetV2* dipilih karena ukuran model yang kecil, *evaluation time* yang cepat serta kemampuan prediksi yang cukup tinggi. Kemudian setelah memilih model *ShuffleNetV2*, maka selanjutnya akan dilakukan *Deploying to Android* dengan menggunakan *TensorFlow Lite*. Pada aplikasi *android*, penambahan algoritma *face detection* dengan menggunakan *Haar Cascade Classifier* sangat efektif untuk proses mendeteksi wajah, sehingga akurasi juga semakin meningkat. Namun, penggunaan algoritma *Haar Cascade Classifier* juga berpengaruh pada *fps* yang dicapai oleh aplikasi, yaitu kurang dari 15 *fps* jika aplikasi mendeteksi sebuah wajah dan melakukan proses klasifikasi.

5.2 Saran

Demi pengembangan lebih lanjut mengenai tugas akhir ini, terdapat beberapa saran sebagai langkah lanjutan yaitu sebagai berikut :

1. Memperbanyak *dataset* citra wajah orang sakit. Karena pada tugas akhir ini, *dataset* citra wajah orang sakit sangat terbatas, sehingga terkadang aplikasi ini kurang bisa mendeteksi kondisi kesehatan seseorang dengan tepat. Permasalahan yang dihadapi oleh penulis adalah kurangnya referensi atau sumber dari *dataset* citra wajah orang sakit.
2. Penambahan algoritma deteksi wajah yang lebih optimal. *Haar Cascade Classifier* adalah algoritma yang digunakan pada tu-

gas akhir ini untuk melakukan proses pendekripsi pada wajah, namun hasilnya masih kurang optimal. Sehingga diperlukan algoritma deteksi wajah yang lebih optimal.

3. Sistem yang diajukan masih membutuhkan pengujian pada kondisi lingkungan yang berbeda-beda hingga kondisi ekstrem, seperti faktor pencahayaan dan umur.
4. Penambahan dan perbaikan tampilan antarmuka pada aplikasi dan fitur yang disediakan, seperti penambahan fitur untuk menyimpan gambar setelah aplikasi tersebut dapat mendekripsi wajah dan melakukan proses klasifikasi.

DAFTAR PUSTAKA

- [1] “Artificial intelligence in business: Using ai in your company.” <https://www.datamation.com/artificial-intelligence/artificial-intelligence-in-business.html>. Accessed: 2019-04-21. (Dikutip pada halaman xi, 6).
- [2] “Machine learning.” https://vas3k.com/blog/machine_learning/. Accessed: 2019-04-21. (Dikutip pada halaman xi, 7).
- [3] “Artificial intelligence, machine learning, dan deep learning.” <https://www.zdnet.com/article/ai-is-revolutionizing-customer-service/>. Accessed: 2019-04-22. (Dikutip pada halaman xi, 9).
- [4] “The differences between artificial and biological neural networks.” <https://towardsdatascience.com/the-differences-between-artificial-and-biological-neural-networks-a8b46db828b7>. Accessed: 2019-04-22. (Dikutip pada halaman xi, 10).
- [5] “Artificial neural network.” https://en.wikipedia.org/wiki/Artificial_neural_network. Accessed: 2019-04-22. (Dikutip pada halaman xi, 12).
- [6] “Deep learning: Feedforward neural network.” <https://towardsdatascience.com/deep-learning-feedforward-neural-network-26a6705dbdc7>. Accessed: 2019-04-22. (Dikutip pada halaman xi, 12).
- [7] “A guide to receptive field arithmetic for convolutional neural networks.” <https://medium.com/mlreview/a-guide-to-receptive-field-arithmetic-for-convolutional-neural-networks-e0f514068807>. (Dikutip pada halaman xi, 16, 20).
- [8] “Multi-class neural networks: Softmax.” <https://developers.google.com/machine-learning/>

- crash-course/multi-class-neural-networks/softmax. Accessed: 2019-04-22. (Dikutip pada halaman xi, 21).
- [9] “A primer into neural networks.” <https://medium.com/datadriveninvestor/a-primer-into-neural-networks-a0bc02d8513b>. Accessed: 2019-04-22. (Dikutip pada halaman xi, 22).
- [10] “Tensorflow.” https://www.tensorflow.org/beta/guide/effective_tf2. Accessed: 2019-04-23. (Dikutip pada halaman xi, 23, 24).
- [11] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, June 2016. (Dikutip pada halaman xi, 25).
- [12] C. Szegedy, S. Ioffe, and V. Vanhoucke, “Inception-v4, inception-resnet and the impact of residual connections on learning,” CoRR, vol. abs/1602.07261, 2016. (Dikutip pada halaman xi, 26, 27).
- [13] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” arXiv preprint arXiv:1512.03385, 2015. (Dikutip pada halaman xi, 27).
- [14] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilennets: Efficient convolutional neural networks for mobile vision applications,” CoRR, vol. abs/1704.04861, 2017. (Dikutip pada halaman xi, 28).
- [15] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, “Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation,” CoRR, vol. abs/1801.04381, 2018. (Dikutip pada halaman xi, 29).
- [16] S. Chen, Y. Liu, X. Gao, and Z. Han, “Mobilefacenets: Efficient for accurate real-time face verification on mobile devices,”

- CoRR, vol. abs/1804.07573, 2018. (Dikutip pada halaman xi, 30).
- [17] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, “SqueezeNet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size,” CoRR, vol. abs/1602.07360, 2016. (Dikutip pada halaman xi, 30, 31).
 - [18] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, “Learning transferable architectures for scalable image recognition,” CoRR, vol. abs/1707.07012, 2017. (Dikutip pada halaman xi, 31, 32).
 - [19] G. Huang, Z. Liu, and K. Q. Weinberger, “Densely connected convolutional networks,” CoRR, vol. abs/1608.06993, 2016. (Dikutip pada halaman xi, 32).
 - [20] X. Zhang, X. Zhou, M. Lin, and J. Sun, “Shufflenet: An extremely efficient convolutional neural network for mobile devices,” CoRR, vol. abs/1707.01083, 2017. (Dikutip pada halaman xi, 33, 34).
 - [21] N. Ma, X. Zhang, H. Zheng, and J. Sun, “Shufflenet V2: practical guidelines for efficient CNN architecture design,” CoRR, vol. abs/1807.11164, 2018. (Dikutip pada halaman xi, 33, 34).
 - [22] M. de Bruijne, “Machine learning approaches in medical image analysis : from detection to diagnosis,” Medical Image Analysis, vol. 33, pp. 94–97, 2016. (Dikutip pada halaman 1).
 - [23] C. Williams, “A brief introduction to artificial intelligence,” in Proceedings OCEANS '83, pp. 94–99, Aug 1983. (Dikutip pada halaman 5).
 - [24] J. R. Koza, F. H. Bennett, D. Andre, and M. A. Keane, “Automated design of both the topology and sizing of analog electrical circuits using genetic programming,” in Artificial Intelligence in Design '96, pp. 151–170, Springer Netherlands, 1996. (Dikutip pada halaman 6).

- [25] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities,” Proceedings of the National Academy of Sciences, vol. 79, no. 8, pp. 2554–2558, 1982. (Dikutip pada halaman 10).
- [26] D. Graupe, Principles of Artificial Neural Networks. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 2nd ed., 2007. (Dikutip pada halaman 10).
- [27] K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” Biological Cybernetics, vol. 36, pp. 193–202, Apr. 1980. (Dikutip pada halaman 16).
- [28] D. H. Hubel and T. N. Wiesel, “Receptive fields and functional architecture of monkey striate cortex,” The Journal of Physiology, vol. 195, pp. 215–243, Mar. 1968. (Dikutip pada halaman 16).
- [29] R. Thornhill and S. W. Gangestad, “Facial sexual dimorphism, developmental stability, and susceptibility to disease in men and women,” Evolution and Human Behavior, vol. 27, pp. 131–144, Mar. 2006. (Dikutip pada halaman 35).
- [30] L. A. Zebrowitz and G. Rhodes, “Sensitivity to “bad genes” and the anomalous face overgeneralization effect: Cue validity, cue utilization, and accuracy in judging intelligence and health,” Journal of Nonverbal Behavior, vol. 28, no. 3, pp. 167–185, 2004. (Dikutip pada halaman 35).
- [31] N. Pound, D. W. Lawson, A. M. Toma, S. Richmond, A. I. Zhurov, and I. S. Penton-Voak, “Facial fluctuating asymmetry is not associated with childhood ill-health in a large british cohort study,” Proceedings of the Royal Society B: Biological Sciences, vol. 281, pp. 20141639–20141639, Aug. 2014. (Dikutip pada halaman 35).
- [32] V. Coetzee, D. I. Perrett, and I. D. Stephen, “Facial adiposity: A cue to health?,” Perception, vol. 38, pp. 1700–1711, Jan. 2009. (Dikutip pada halaman 36).

- [33] C. I. Fisher, A. C. Hahn, L. M. DeBruine, and B. C. Jones, “Integrating shape cues of adiposity and color information when judging facial health and attractiveness,” *Perception*, vol. 43, pp. 499–508, Jan. 2014. (Dikutip pada halaman 36).
- [34] A. A. Volk, J. M. Lukjanczuk, and V. L. Quinsey, “Influence of infant and child facial cues of low body weight on adults' ratings of adoption preference, cuteness, and health,” *Infant Mental Health Journal*, vol. 26, no. 5, pp. 459–469, 2005. (Dikutip pada halaman 36).
- [35] M. J. Rantala, V. Coetzee, F. R. Moore, I. Skrinda, S. Kecko, T. Krama, I. Kivleniece, and I. Krams, “Adiposity, compared with masculinity, serves as a more valid cue to immunocompetence in human mate choice,” *Proceedings of the Royal Society B: Biological Sciences*, vol. 280, pp. 20122495–20122495, Nov. 2012. (Dikutip pada halaman 36).
- [36] V. Coetzee, J. Chen, D. I. Perrett, and I. D. Stephen, “Deciphering faces: Quantifiable visual cues to weight,” *Perception*, vol. 39, pp. 51–61, Jan. 2010. (Dikutip pada halaman 37).
- [37] K. Wolffhechel, A. C. Hahn, H. Jarmer, C. I. Fisher, B. C. Jones, and L. M. DeBruine, “Testing the utility of a data-driven approach for assessing BMI from face images,” *PLOS ONE*, vol. 10, p. e0140347, Oct. 2015. (Dikutip pada halaman 37).
- [38] I. J. Holzleitner, D. W. Hunter, B. P. Tiddeman, A. Seck, D. E. Re, and D. I. Perrett, “Men's facial masculinity: When (body) size matters,” *Perception*, vol. 43, pp. 1191–1202, Jan. 2014. (Dikutip pada halaman 37).
- [39] I. M. L. Scott, N. Pound, I. D. Stephen, A. P. Clark, and I. S. Penton-Voak, “Does masculinity matter? the contribution of masculine face shape to male attractiveness in humans,” *PLoS ONE*, vol. 5, p. e13585, Oct. 2010. (Dikutip pada halaman 37).
- [40] B. Fink, L. Bunse, P. J. Matts, and D. D'Emiliano, “Visible skin colouration predicts perception of male facial age, health

- and attractiveness,” International Journal of Cosmetic Science, vol. 34, pp. 307–310, May 2012. (Dikutip pada halaman 37).
- [41] N. Samson, B. Fink, and P. Matts, “Interaction of skin color distribution and skin surface topography cues in the perception of female facial age and health,” Journal of Cosmetic Dermatology, vol. 10, pp. 78–84, Feb. 2011. (Dikutip pada halaman 37).
- [42] I. D. Stephen, M. J. L. Smith, M. R. Stirrat, and D. I. Perrett, “Facial skin coloration affects perceived health of human faces,” International Journal of Primatology, vol. 30, pp. 845–857, Oct. 2009. (Dikutip pada halaman 37, 38).
- [43] B. Fink and P. Matts, “The effects of skin colour distribution and topography cues on the perception of female facial age and health,” Journal of the European Academy of Dermatology and Venereology, vol. 22, pp. 493–498, Apr. 2008. (Dikutip pada halaman 37).
- [44] P. J. Matts, B. Fink, K. Grammer, and M. Burquest, “Color homogeneity and visual perception of age, health, and attractiveness of female facial skin,” Journal of the American Academy of Dermatology, vol. 57, pp. 977–984, Dec. 2007. (Dikutip pada halaman 37).
- [45] B. Fink, P. Matts, D. D’Emiliano, L. Bunse, B. Weege, and S. Röder, “Colour homogeneity and visual perception of age, health and attractiveness of male facial skin,” Journal of the European Academy of Dermatology and Venereology, pp. no-no, Nov. 2011. (Dikutip pada halaman 37).
- [46] A. L. Jones, R. S. S. Kramer, and R. Ward, “Signals of personality and health: The contributions of facial shape, skin texture, and viewing angle.,” Journal of Experimental Psychology: Human Perception and Performance, vol. 38, no. 6, pp. 1353–1361, 2012. (Dikutip pada halaman 38).
- [47] E. L. Abel and M. L. Kruger, “Smile intensity in photographs predicts longevity,” Psychological Science, vol. 21, pp. 542–544, Feb. 2010. (Dikutip pada halaman 38).

- [48] S. D. Pressman and S. Cohen, “Does positive affect influence health?,” *Psychological Bulletin*, vol. 131, pp. 925–971, Nov. 2005. (Dikutip pada halaman 38).
- [49] T. Sundelin, M. Lekander, G. Kecklund, E. J. W. V. Someren, A. Olsson, and J. Axelsson, “Cues of fatigue: Effects of sleep deprivation on facial appearance,” *Sleep*, vol. 36, pp. 1355–1360, Sept. 2013. (Dikutip pada halaman 38).
- [50] M. Kawulok, M. E. Celebi, and B. Smolka, eds., *Advances in Face Detection and Facial Image Analysis*. Springer International Publishing, 2016. (Dikutip pada halaman 42).
- [51] K. K. Pal and K. S. Sudeep, “Preprocessing for image classification by convolutional neural networks,” in *2016 IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, pp. 1778–1781, May 2016. (Dikutip pada halaman 45).

Halaman ini sengaja dikosongkan

BIOGRAFI PENULIS



Naufal Reyhan Fadhil, lahir pada 15 Oktober 1996 di Kabupaten Jember, Provinsi Jawa Timur. Penulis merupakan anak pertama dari dua bersaudara. Penulis menyelesaikan pendidikan dasar pada SDN Jember Lor III pada tahun 2009. Pada tahun 2012, penulis menyelesaikan tingkat menengah pada SMP Negeri 2 Jember. Penulis menyelesaikan tingkat atas pada SMA Negeri 1 Jember pada tahun 2015. Penulis kemudian melanjutkan pendidikan Strata Satu pada Departemen Teknik Komputer Fakultas Teknologi Elektro Institut Teknologi Sepuluh Nopember Surabaya. Selama masa kuliah, penulis aktif dalam berbagai kepanitiaan maupun organisasi. Pada tahun 2017, penulis mendapatkan amanah sebagai wakil ketua MAGE (*Multimedia And Game Event*) yang diselenggarakan oleh Departemen Teknik Komputer FTE ITS. Di tahun yang sama, penulis mendapatkan amanah sebagai Ketua UKM IFLS. Kemudian pada tahun 2018, penulis mendapatkan amanah sebagai Ketua LMB ITS dan menjabat hingga pertengahan tahun 2019. Selain itu, penulis juga aktif menjadi asisten laboratorium B401 Komputasi Multimedia dan pernah menjabat sebagai koordinator dalam bidang *Internet of Things Development*. Penulis memiliki hobby *travelling*, *hiking*, dan juga *videography*. Untuk kepentingan pada penulis terkait kritik, saran, atau pertanyaan mengenai tugas akhir ini dapat menghubungi melalui *E-Mail* : naufalfadhil15@gmail.com

Halaman ini sengaja dikosongkan