TUGAS AKHIR – EE 184801

PEMANFAATAN KOMPUTASI AWAN UNTUK PENGENALAN EKSPRESI WAJAH MENGGUNAKAN AWS DEEPLENS

Gregorius Rafael
NRP 07111640000129

Dosen Pembimbing
Dr. Ir. Hendra Kusuma, M. Eng. Sc.
Ir. Tasripan, MT.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

TUGAS AKHIR – EE 184801

PEMANFAATAN KOMPUTASI AWAN UNTUK PENGENALAN EKSPRESI WAJAH MENGGUNAKAN AWS DEEPLENS

Gregorius Rafael
NRP 07111640000129

Dosen Pembimbing
Dr. Ir. Hendra Kusuma, M. Eng. Sc.
Ir. Tasripan, MT.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

FINAL PROJECT – EE 184801

THE UTILIZATION OF CLOUD COMPUTING FOR FACIAL EXPRESSION RECOGNITION USING AWS DEEPLENS

Gregorius Rafael
NRP 07111640000129

Supervisors
Dr. Ir. Hendra Kusuma, M. Eng. Sc.
Ir. Tasripan, MT.

DEPARTMENT OF ELECTRICAL ENGINEERING
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2020

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi keseluruhan Tugas akhir saya dengan judul "**PEMANFAATAN KOMPUTASI AWAN UNTUK PENGENALAN EKSPRESI WAJAH MENGGUNAKAN AWS DEEPLENS**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

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

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

Surabaya, 7 Juli 2020

Gregorius Rafael
07111640000129

Halaman ini sengaja dikosongkan

PEMANFAATAN KOMPUTASI AWAN UNTUK PENGENALAN EKSPRESI WAJAH MENGGUNAKAN AWS DEEPLENS

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik

Pada

Bidang Studi Elektronika
Departemen Teknik Elektro
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Menyetujui :

Dosen Pembimbing I

Dr. Ir. Hendra Kusuma, M. Eng. Sc.
NIP. 196409021989031003

SURABAYA
JULI, 2020

PEMANFAATAN KOMPUTASI AWAN UNTUK PENGENALAN EKSPRESI WAJAH MENGGUNAKAN AWS DEEPLENS

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik

Pada

Bidang Studi Elektronika
Departemen Teknik Elektro
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Menyetujui :

Dosen Pembimbing II

Ir. Tasripan, MT..
NIP. 196204181990031004

SURABAYA
JULI, 2020

Halaman ini sengaja dikosongkan

PEMANFAATAN KOMPUTASI AWAN UNTUK PENGENALAN EKSPRESI WAJAH MENGGUNAKAN AWS DEEPLENS

Nama mahasiswa : Gregorius Rafael
Dosen Pembimbing I : Dr. Ir. Hendra Kusuma, M. Eng. Sc.
Dosen Pembimbing II : Ir. Tasripan, MT.

Abstrak:

Komunikasi tatap-muka merupakan suatu bentuk interaksi yang sering dilakukan oleh semua orang. Akan tetapi, hal tersebut sulit untuk diimplementasikan bagi penyandang tuna netra, terutama bagi mereka untuk mengenali ekspresi wajah dari lawan bicaranya dan ekspresi wajah dipercayai mempunyai kaitan erat dalam emosi seseorang dan memberikan informasi yang lebih lengkap saat berkomunikasi. Oleh karena itu, diperlukanlah suatu alat yang bisa membaca ataupun mengenali ekspresi wajah, sehingga para penyandang tuna netra dapat mengenali ekspresi wajah dari lawan bicaranya dengan jelas dan ekspresi itu dikonversikan menjadi informasi non-verbal berupa suara yang mengindikasikan ekspresi dari lawan bicara penyandang tuna netra.

Penelitian tentang pengenalan ekspresi wajah kepada penyandang tunanetra telah banyak dilakukan. Akan tetapi, salah satu skenario kendala yang dihadapi dari beberapa penelitian sebelumnya adalah biaya pembuatan dan performa yang kurang sesuai dengan pengguna. Pada penelitian ini akan digunakan perangkat untuk komputasi awan dan pembelajaran mesin AWS DeepLens yang bertujuan untuk mencari performa dari perangkat yang menggunakan layanan komputasi awan AWS dan membandingkannya dengan performa dari alat yang menggunakan pemrograman *deep learning* tanpa komputasi awan. Semua bentuk keluaran dari perangkat akan menyampaikan informasi pengenalan ekspresi ke pengguna melalui media suara. Keberhasilan algoritma *deep learning* akan diukur dalam *confusion matrix* dengan rerata keberhasilan sebesar 73,43 % . Alat yang dikembangkan dari divais AWS DeepLens ini menggunakan sistem komputasi awan dari AWS, dengan harapan alat ini *robust* secara sistem dan *real-time* dalam penggunaannya.

Kata kunci: Komputasi Awan, Tuna Netra, Informasi Non-Verbal

Halaman ini sengaja dikosongkan

THE UTILIZATION OF CLOUD COMPUTING FOR FACIAL EXPRESSION RECOGNITION USING AWS DEEPLENS

Student Name : Gregorius Rafael
Supervisor I : Dr. Ir. Hendra Kusuma, M. Eng. Sc.
Supervisor II : Ir. Tasripan, MT.

Abstract:

Face-to-face communication is a form of interaction that are done by every people. Meanwhile, this type of communicaton is hard to be implemented for visually impaired person, especially for them to recognize their interlocutors' facial expression and It is believed that facial expression has exact relation with one's emotion and provides more detailed information when in communication. Hence, a device is needed for recognizing facial expression, so visually impaired people could recognize their interlocutors' facial expression clearly and those expressions converted into non-verbal information in type of sound that indicates it.

Researches on facial expression recognition for visually impaired people have been so much done, but one of the problems that are faced by previous researches are manufacturing costs and lack of performance that are demanded by user. This research will use a device built for cloud computing and machine learning called AWS DeepLens which purposed for looking the performance of the device using AWS cloud computing services and compare it to deep learning without clouds computing devices. The device's outputs will inform facial expression information to the user via sound. Deep learning algorithm success will be measured in confusion matrix and its average rate is 73,43%. This developed AWS DeepLens utilizes cloud computing system from AWS, with objectives of system robustness and real-time usages.

Key Words: *Cloud Computing, Visually-Impaired People, Non-Verbal Information*

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Tiada kata yang mampu menggambarkan seberapa banyak syukur yang harus penulis panjatkan kepada Tuhan Yang Maha Esa, atas segala Rahmat, Karunia, dan Petunjuk yang telah dilimpahkan-Nya, walaupun baragam tantangan dan cobaan menghadang, pada akhirnya penulis mampu menyelesaikan Tugas Akhir ini yang berjudul "**PEMANFAATAN KOMPUTASI AWAN UNTUK PENGENALAN EKSPRESI WAJAH MENGGUNAKAN AWS DEEPLENS**".

Tugas Akhir ini disusun sebagai salah satu persyaratan untuk menyelesaikan jenjang pendidikan S1 pada Bidang Studi Elektronika, Departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember. Atas selesainya penyusunan tugas akhir ini, penulis mengucapkan terima kasih kepada:

1. Tuhan Yang Maha Esa atas limpahan rahmat, karunia dan petunjuk-Nya.
2. Ibu Anastasia Lidya Maukar dan Ayah Antonius Agus Susanto Widojoko atas segala doa dan cinta yang tak henti pada penulis dalam keadaan apapun. Semoga Tuhan senantiasa melindungi dan memberkati mereka.
3. Bapak Dr. Ir. Hendra Kusuma, M. Eng. Sc. dan Bapak Ir. Tasripan, MT. selaku dosen pembimbing yang selama ini telah memberikan arahan, bimbingan dan perhatiannya selama proses penyelesaaian Tugas Akhir ini.
4. Para asisten dan teman-teman di Laboratorium Mikroelektronika dan Sistem Tertanam yang menjadi teman seperjuangan dalam proses menyelesaikan Tugas Akhir.
5. Rohman Widiyanto dan Julius Sintara yang senantiasa menjadi teman penulis dalam mengerjakan Tugas Akhir setiap saat.
6. Teman-teman pada Bidang Studi Teknik Sistem Tenaga, terutama Gokma Eunike Napitu, Divyah Laksmi, Qonitah Yumna, dan Jonathan Sebastian yang telah menjadi teman diskusi penulis dalam memberikan inspirasi dalam pemrograman dan MATLAB.
7. Teman-teman seperjuangan e56 yang telah menemani dan menorehkan cerita selama masa kuliah sampai penyusunan Tugas Akhir ini

8. Seluruh dosen, tenaga pendidik, dan karyawan Departemen Teknik Elektro ITS yang telah memberikan banyak ilmu dan menciptakan suasana belajar yang luar biasa.

Penulis telah berusaha maksimal dalam penyusunan tugas akhir ini. Namun tetap besar harapan penulis untuk menerima saran dan kritik untuk perbaikan dan pengembangan tugas akhir ini. Semoga tugas akhir ini dapat memberikat manfaat yang luas.

Surabaya, Juni 2020

Gregorius Rafael
07111640000129

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

Pada bab ini membahas mengenai hal-hal yang mendasari penelitian dengan penjelasan mendetail. Hal tersebut meliputi latar belakang, perumusan masalah, tujuan penulisan, batasan masalah, metodologi penelitian, sistematika penulisan, dan relevansi.

1.1 Latar Belakang

Dalam kehidupan sehari-hari, penyandang tunanetra menghadapi banyak tantangan dalam kehidupan mereka, seperti mencari jalan di lingkungan yang tidak dikenal, mendeteksi obyek dan orang, serta mengenali wajah beserta ekspresinya. Salah satu tantangan utama saat berinteraksi tatap muka bagi penyandang tunanetra adalah mereka tidak dapat mengenali informasi non-verbal dari lawan bicaranya, seperti ekspresi wajah, gerakan tubuh, dan gesture atau isyarat tubuh. **Ekspresi wajah diyakini memiliki kaitan yang erat dengan emosi seseorang dan mampu memberikan informasi yang lebih lengkap saat berkomunikasi**.

Penyandang tunanetra tidak bisa mendapatkan informasi tersebut karena keterbatasan penglihatannya. Manusia juga dapat memproses informasi yang biasanya diperoleh melalui penglihatan dengan menggunakan indra lain, seperti pendengaran atau peraba [1]. Untuk informasi seperti warna, tulisan, komunikasi non-verbal, atau isyarat dapat diperoleh melalui pendengaran atau sabuk *haptic*. Contoh yang paling umum adalah huruf Braille, yang banyak digunakan untuk menyampaikan informasi tertulis melalui indra peraba. Telah ada penelitian tentang menyampaikan informasi visual untuk penyandang tunanetra dengan menggunakan sabuk *haptic* [2]. Namun, dalam segi desain, perangkat yang dibuat kurang praktis karena bentuknya yang besar dan juga harganya yang cukup mahal. Faktor-faktor ini menyebabkan perangkat bantu tersebut kurang dapat dikomersilkan. Perlu dipikirkan adanya perangkat yang lebih praktis dengan harga yang terjangkau, sehingga perangkat tersebut dapat mencakup lebih banyak pengguna.

Meskipun beberapa penelitian tentang pengenalan ekspresi wajah kepada penyandang tunanetra telah dilakukan, namun masih terdapat kendala yang selalu dihadapi, yaitu yang berkaitan dengan biaya pembuatan dan desain perangkat yang kurang praktis. Pada penelitian ini akan dikembangkan

sebuah perangkat yang lebih praktis dan murah dengan performa yang lebih baik, sehingga akan lebih banyak penyandang tunanetra yang terbantu. Salah satu fitur kunci yang akan diteliti dan dikembangkan pada penelitian ini adalah pengklasifikasi emosi/ekspresi secara visual dan bentuk respon dari sistem tersebut.

Metodologi penelitian ini akan berfokus pada perencanaan pengenalan ekspresi atau emosi wajah dengan menggunakan sistem komputasi awan dimana pendekatan ini akan diimplementasikan pada sistem tertanam pada perangkat portabel (*wearable*).

Pada penelitian ini pengenalan dilakukan untuk 8 (delapan) ekspresi wajah universal yaitu bahagia (*happiness*), sedih (*sadness*), takut (*fear*), muak (*disgust*), marah (*anger*), terbelalak/terkejut (*surprise*), bingung (*confused*), dan tenang (*calm*).

1.2 Permasalahan

Perumusan masalah dari tugas akhir ini adalah :

1. Bagaimana mendapatkan metode pada komputasi awan yang digunakan untuk mengklasifikasi ekspresi/emosi pada wajah.
2. Bagaimana cara menginformasikan hasil klasifikasi ekspresi/emosi kepada penyandang tuna netra.
3. Bagaimana desain sistem beserta perangkat yang praktis dan mudah dipakai sehingga dapat digunakan penyandang tuna netra dengan baik

1.3 Tujuan dan Manfaat

Tugas akhir ini bertujuan untuk :

1. Bagaimana mendapatkan tingkat akurasi pengenalan yang baik dengan memanfaatkan komputasi awan AWS.
2. Bagaimana menentukan *services* pada AWS yang tepat, *robust*, serta ekonomis.
3. Bermanfaat untuk memberikan informasi ekspresi wajah lawan bicara penyandang tuna netra dengan media suara.

1.4 Batasan Masalah

Batasan masalah dalam tugas akhir ini adalah :

1. Penggunaan sistem computer yang terintergrasi dengan kamera keluaran AWS DeepLens.

2. Penggunaan *platform* komputasi awan milik AWS beserta *analytics services* dari AWS
3. Performa perangkat diuji pada kondisi ideal tanpa memperhatikan kendala adanya variasi pencahayaan.

1.5 Metodologi

Metodologi yang digunakan dalam menyusun penelitian tugas akhir ini adalah sebagai berikut:

1. Studi Literatur
Mengumpulkan referensi-referensi yang berhubungan dengan pengenalan ekspresi wajah, sistem komputasi awan, dan pembelajaran mesin & *deep learning* seperti buku, *paper*, serta jurnal.

2. Perancangan dan Pembuatan Sistem Komputasi Awan
Pada tahap ini dilakukan perancangan serta pembuatan sistem komputasi awan serta implementasinya kepada divais AWS DeepLens yang diperlukan untuk mendukung Tugas Akhir ini.
3. Pengujian Alat
Tahap ini berisi pengujian alat yang telah dirancang sistem pengenalan ekspresi wajah-nya untuk digunakan dan dicoba langsung oleh penyandang tuna netra untuk memperoleh kritik dan saran mengenai alat yang dirancang.
4. Analisis Data dan Evaluasi
Memberikan kesimpulan mengenai pengujian sistem komputasi awan untuk mengetahui apakah implementasi metode komputasi awan untuk pengenalan ekspresi wajah cocok digunakan.
5. Penyelesaian Laporan Tugas Akhir
Tahap ini dilakukan sebagai tahap akhir dari serangkaian pengerjaan tugas akhir ini. Juga dilakukan guna memenuhi persyaratan kelulusan mata kuliah Tugas Akhir.

1.6 Sistematika Penulisan

Sistematika penulisan dalam tugas akhir ini terdiri dari lima BAB dengan uraian sebagai berikut :

1. BAB 1 merupakan pendahuluan yang berisikan latar belakang masalah, tujuan, metodologi, sistematika penulisan, dan relevansi

2. BAB 2 berisikan kajian pustaka dan dasar teori yang membahas mengenai teori-teori penunjang yang berkaitan dengan sistem komputasi awan dan *datasheet* mengenai AWS DeepLens.
3. BAB 3 berisikan perancangan sistem komputasi awan untuk pengenalan ekspresi wajah secara visual.
4. BAB 4 berisikan pengujian performa alat AWS DeepLens dengan sistem komputasi awan yang telah dirancang kepada pengguna normal dan pengguna penyandang tuna netra.
5. BAB 5 berisikan kesimpulan dari analisis yang dilakukan dan saran untuk pengembangan lebih lanjut.

1.7 Relevansi

Penelitian ini diharapkan dapat membantu penyandang tuna netra dalam bersosialisasi dengan perangkat yang portabel. Dengan memanfaatkan teknologi komputasi awan, perangkat ini diharapkan dapat memiliki performa yang baik meskipun ukuran masih kurang portabel dan juga kurang nyaman digunakan. Akan tetapi, rancangan alat pengenal ekspresi wajah berbasis komputasi awan ini dapat menjadi referensi bagi mahasiswa dalam penelitian di bidang serupa, yaitu teknologi asistif dan komputasi awan.

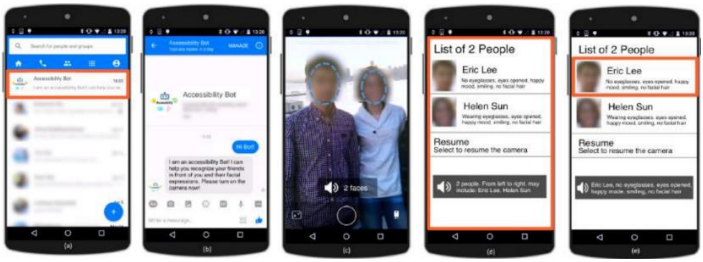
BAB 2
KAJIAN PUSTAKA DAN DASAR TEORI

Dalam bab ini akan dijelaskan teori-teori dan penelitian terkait dalam pembuatan perangkat lunak beserta perangkat keras. Kajian penelitian terkait akan disajikan secara singkat di sub-bab 2.1, sedangkan dasar teori yang berkaitan dengan Tugas Akhir ini akan diberikan di sub-bab 2.2 hingga 2.14 dan dataset yang digunakan akan dijelaskan pada sub-bab 2.15.

2.1 Kajian Pustaka dan Penelitian Terkait

Dalam sub-bab ini akan dibahas mengenai kajian pustaka dan penelitian terkait yang memuat tentang hasil penilitian yang pernah dilakukan yang berhubugnan dengan sistem pengenalan ekspresi wajah maupun pengenalan wajah.

2.1.1 *A Face Recognition Application for People with Visual Impairments: Understanding Use Beyond the Lab* [3]



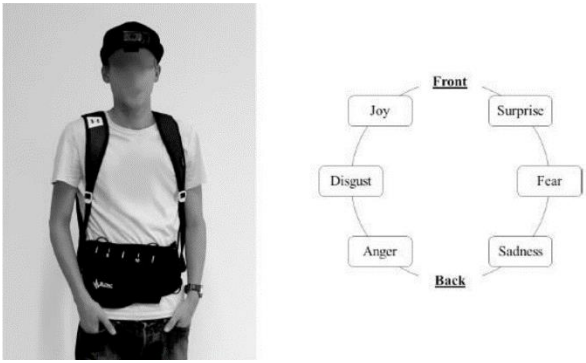
Gambar 2. 1 Penggunaan *Accessibility Bot* [3].

Dalam penelitian ini, dibuat *Accessibility bot*, sebuah bot prototipe di Facebook Messenger yang memanfaatkan algoritma computer vision dan set foto yang ditandai dari teman pengguna di Facebook untuk membantu tunanetra mengenali teman-teman mereka. *Accessibility bot* memberikan informasi kepada pengguna tentang identitas teman di kamera dan ekspresi wajah mereka. Bot pada Facebook Messenger dapat dilatih dengan menggunakan foto pengguna yang ada di Facebook tanpa harus membuat dataset tersendiri. Bot bekerja dengan TalkBack, pembaca layar pada Android. Android dipilih karena memiliki pengguna yang banyak pada

platform smartphone. Saat pengguna membuka aplikasi dan memulai berbicara dengan bot, bot akan membalas dengan perkenalan singkat, lalu menginstruksikan pengguna untuk menyalakan kamera untuk pengenalan wajah. Aplikasi ini terbatas hanya membantu penyandang tunanetra dalam bersosialisasi lewat media sosial online.

2.1.2 *Conveying Facial Expressions to Blind and Visually Impaired Persons Through a Wearable Vibrotactile Device* [4]

Pada penelitian ini dibuatlah perangkat pengenalan emosi wajah pada penyandang tunanetra menggunakan perangkat *vibrotactile* yang dapat dipakai. Emosi wajah yang dapat deteksi adalah marah, jijik, sedih, senang, terkejut, netral, dan takut. Perangkat ini menggunakan Microsoft Surface Pro 4 tablet sebagai pusat kontrol, Logitech HD Pro Webcam C920 yang dipasang pada topi sebagai pengambil gambar pada jarak pandang pengguna. Sabuk haptic digunakan untuk membedakan emosi wajah satu dan yang lainnya. Gambar yang diambil dari kamera diterima oleh FaceReader 6™ yang keluarannya adalah prediksi emosi wajah. FaceReader 6™ adalah software yang menggunakan algoritma deteksi wajah real-time yang tangguh untuk mendeteksi wajah dari video dan deep neural network yang dapat mengklasifikasikan ekspresi wajah. Meskipun memiliki performa yang sangat baik, ukuran yang cukup besar menjadikan alat ini kurang praktis saat dipakai pada kegiatan sehari-hari.



Gambar 2. 2 Sistem Substitusi Sensorik [4].

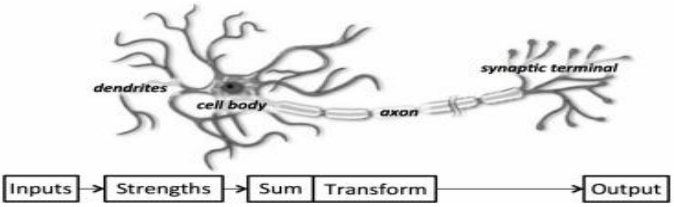
2.1.3 Pengenalan Ekspresi Wajah untuk Tunanetra menggunakan Deep Learning pada Perangkat Portabel [5]

Pada penelitian ini dibuatlah suatu perangkat pengenalan ekspresi wajah bagi penyandang tunanetra menggunakan perangkat Raspberry Pi sebagai pemroses algoritma deep learning, beserta kamera Logitech Webcam C390 yang dipasang pada topi sebagai pengambil gambar dan *headset* sederhana untuk keluaran dari perangkat tersebut. Kombinasi suara *beep* yang keluar dari *headset* tersebut akan menjadi acuan untuk ekspresi wajah atau emosi tertentu yang dideteksi oleh sistem melalui kamera.

Gambar 2. 3 Perangkat dari Sistem Pengenalan Ekspresi Wajah [5].

2.2 *Artificial Neural Network*

Artificial Neural Network adalah paradigma mengenai pemprosesan informasi yang biasanya digunakan untuk mempelajari tingkah laku suatu sistem yang kompleks pada simulasi komputer. Metode ini diilhami oleh otak manusia dalam memproses informasi. Tujuan dari *Artificial Neural Network* adalah memecahkan suatu masalah spesifik dengan alur yang sama dengan otak manusia.

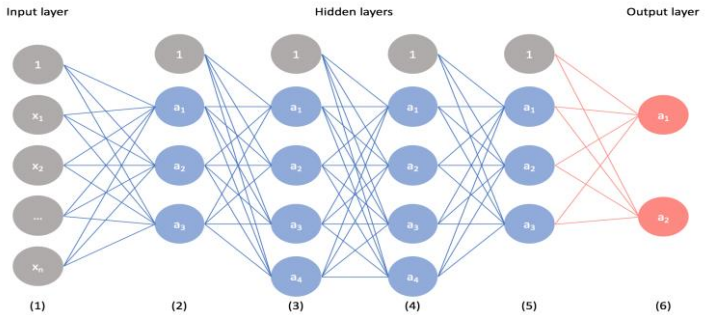


Gambar 2. 4 Deskripsi Fungsional dari Struktur Neuron [1].

Metode ini menerapkan sistem kerja antara satu neuron dengan neuron yang lain pada program komputer. Model matematis dari satu unit neuron adalah :

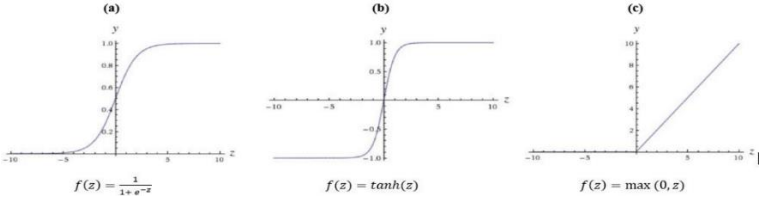
$$a = f(wp + b) \tag{2.1}$$

Masukan skalar p ditransmisikan melalui jaringan yang kekuatannya dikalikan oleh skalar *weight* (w). Hasil perkalian tersebut ditambahkan dengan skalar *bias* (b). Proses tersebut menghasilkan keluaran skalar (a) yang nilainya akan diteruskan ke neuron selanjutnya.



Gambar 2. 5 Contoh Arsitektur *Neural Network* [1].

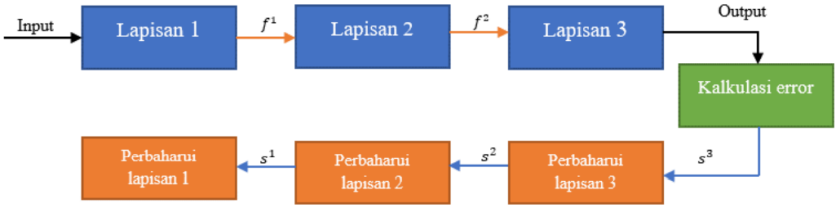
Untuk mempelajari hubungan neuron yang kompleks, diperlukan suatu fungsi yang non-linear yang disebut sebagai fungsi aktivasi. Setiap lapisan dapat memiliki fungsi aktivasi yang berbeda, sehingga setiap lapisan memiliki perilaku yang berbeda pula.



Gambar 2. 6 Contoh Fungsi Aktivasi: (a) Sigmoid (b) Tanh (c) ReLu [1].

Proses otak manusia dalam mempelajari sesuatu dapat dimodelkan oleh *neural network* menggunakan *backpropagation* atau propagasi balik. *Backpropagation* adalah propagasi balik galat sebuah prediksi model sebagai nilai acuan untuk mengganti nilai-nilai *weight* dan *bias* dengan nilai-nilai yang baru pada setiap lapisan pada arsitektur *neural network*. Galat tersebut

didapatkan dengan membandingkan keluaran model dengan target keluaran yang sudah ditentukan sebelumnya. Nilai tersebut akan digunakan untuk melakukan koreksi pada neuron dari lapisan terakhir ke lapisan pertama. Pembaruan tersebut dijalankan untuk memodelkan proses belajar pada otak manusia.



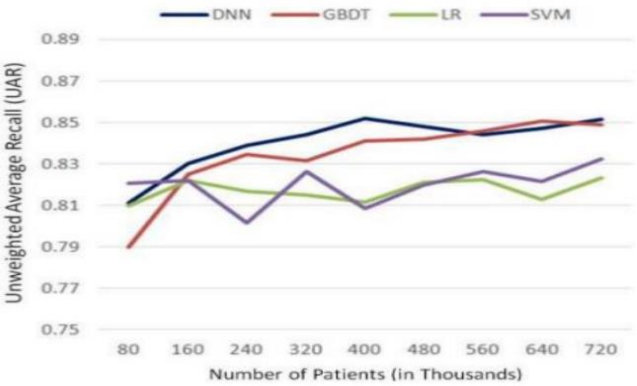
Gambar 2. 7 Diagram Proses *Backpropagation*. [5]

Untuk memperbarui *weight* dan *bias* pada setiap lapisan, diperlukan nilai sensitivitas propagasi (s^M) yang didapatkan dari lapisan setelahnya. Oleh karena itu, nilai terbaru dari suatu lapisan neuron akan mempengaruhi nilai terbaru dari lapisan sebelumnya. Lapisan terakhir akan mendapat nilai sensitivitas propagasinya dari nilai galat yang didapatkan.

2.3 *Deep Learning*

Deep learning adalah sub-bidang pembelajaran mesin yang berkaitan dengan algoritma yang diilhami oleh struktur, kinerja, serta fungsi dari otak manusia yang disebut *artificial neural network*. Pada *deep learning*, model komputer dapat belajar untuk melakukan klasifikasi secara langsung dari data apapun, seperti gambar, tulisan, maupun suara. Performa dari *deep learning* terbukti sangat baik, bahkan dapat melebihi kemampuan otak manusia. Model yang terbentuk dilatih menggunakan data yang sangat banyak dan arsitektur jaringan saraf yang memiliki banyak lapisan.

Esensi dari deep learning adalah adanya komputer yang cukup cepat dan banyaknya data untuk melatih *artificial neural network* yang cukup besar secara efektif. Salah satu keunggulan dari *deep learning* adalah skalabilitas. Ketika kita membangun *artificial neural network* yang cukup besar dan melatihnya dengan dengan semakin banyak data, performanya akan terus meningkat sebanding dengan banyaknya data. Hal ini berbeda dengan algoritma pembelajaran mesin lainnya yang akan mencapai puncak performa pada posisi tertentu.

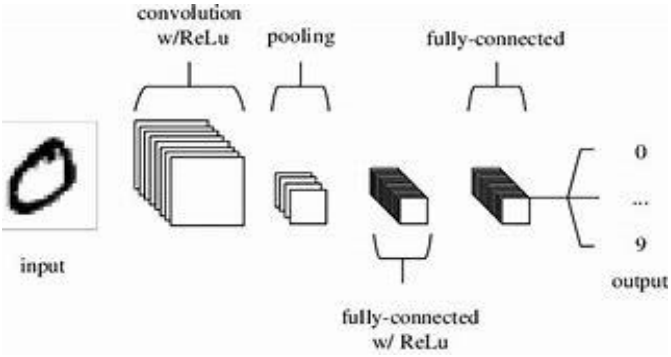


Gambar 2. 8 Performa *Deep Learning* dan Algoritma Pembelajaran Mesin lainnya (GBDT, LR, SVM) dengan Jumlah Data Latih yang Berbeda [2].

Terdapat beberapa tipe artificial neural network pada penggunaan deep learning contohnya adalah *convolutional neural network* (CNN), *recursive neural network*, *recurrent neural network*, *long short-term memory* (LSTM), dan lain-lain. Pada penelitian ini akan diteliti lebih lanjut karena telah digunakan secara umum untuk pengenalan gambar dan penggunaannya diterapkan langsung dengan sistem komputasi awan AWS dengan *service* yang disebut AWS SageMaker.

2.4 *Convolutional Neural Network*

Indra pengelihatan manusia sangatlah kompleks. Tidak hanya manusia dapat mengidentifikasi suatu objek, manusia dapat mengetahui kedalaman objek, mengetahui kontur suatu objek, dan juga memisahkan objek terhadap latar belakangnya. Hal ini dapat terjadi dikarenakan kemampuan otak dan relasi antar-neuron otak manusia yang sangat kompleks. Untuk meniru perilaku neuron-neuron tersebut, kita tidak bisa memakai arsitektur *neural network* biasanya. Oleh karena itu, ditemukannya arsitektur yang dapat menangani gambar sebagai masukan yaitu CNN (*Convolutional Neural Network*).



Gambar 2. 9 Arsitektur CNN Sederhana [6].

CNN terdiri dari 3 lapisan yaitu *convolutional*, *pooling*, dan *fully-connected*. Fungsi dasar dari CNN bisa dibagi menjadi 4 proses [6]:

1. Gambar masuk pada *input layer* yang besarnya sama dengan jumlah pixel setiap saluran warna (RGB).
2. *Convolutional layer* akan menentukan keluaran dari neuron yang akan dihubungkan ke bagian input lokal melalui perhitungan scalar antara weights dan bagian yang dihubungkan ke input. *Rectified Linear Unit* (ReLu) diterapkan pada output dari layer sebelumnya.
3. *Pooling layer* melakukan downsampling luaran layer sebelumnya sehingga mengurangi dimensi input dan mengurangi jumlah parameter dalam proses aktivasi tersebut.
4. *Fully-connected layer* arsitekturnya sama seperti neural network standar yang luarannya adalah nilai dari masing masing kelas. Lapisan ini diletakkan dibagian akhir dari arsitektur CNN, oleh karena itu dimensi masukannya lebih sederhana.

Dengan melewati proses ini, CNN dapat mengubah masukan gambar yang dimensinya kompleks menjadi lebih sederhana karena teknik *convolutional* dan *down-sampling* untuk klasifikasi objek.

2.5 VGG16

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

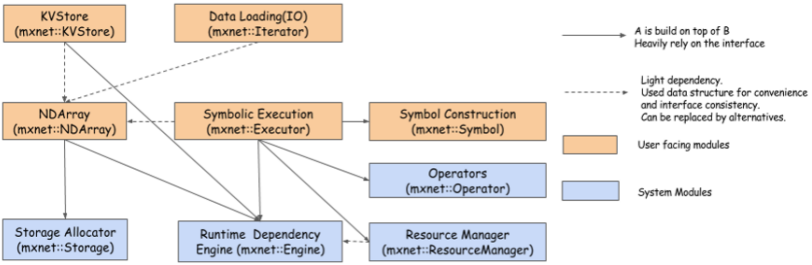
Gambar 2. 10 Arsitektur VGG16 [7].

VGG16 adalah arsitektur CNN yang menambah kedalam arsitektur menggunakan *convolutional filter* kecil yang ternyata dalam proses pengujian, performanya jauh lebih baik dari konfigurasi yang sudah ada dengan menambah kedalaman aristektur menjadi 16-19 *weight layer*. VGG16 mempunyai lebih dari 100 juta parameter, sehingga dibutuhkan komputer yang kuat untuk melatihnya [7].

2.6 MXNet

MXNet adalah *open-source library* arsitektur CNN yang didesain oleh Apache dan satu-satunya arsitektur CNN yang didukung pada *platform* AWS. MXNet dikenal juga sebagai arsitektur CNN yang bisa dipakai pada banyak bahasa pemrograman, salah satunya adalah Python. Keunggulan utama dari MXNet adalah kemampuannya sebagai *hybrid front-end*, dimana model yang dibentuk oleh arsitektur MXNet bias langsung dipanggil untuk keperluan optimalisasi proses eksekusi pelatihan, pembelajaran dan penerapan model *deep learning*. Arsitektur ini didasari pada struktur *inverted*

residual, dimana adanya jalan pintas di antara *bottleneck layer* yang tipis. Lapisan ekspansi menengah menggunakan konvolusi mendalam yang ringan untuk menapis fitur sebagai sumber non-linearitas. Hal ini dilakukan untuk kekuatan representatif dari arsitektur. Oleh karena itu, arsitektur ini memiliki parameter yang lebih sedikit dengan menjaga performanya [8].



Gambar 2. 11 Arsitektur MXNet [11].

2.7 Komputer Visi

Komputer visi adalah bidang ilmiah interdisipliner yang membahas bagaimana komputer dapat digunakan ntuk memperoleh pemahaman tingkat tinggi dari citra, baik gambar maupun video digital. Dari perspektif teknik, komputer visi adalah upaya otomatisasi tugas-tugas yang dapat dilakukan oleh sistem visual manusia.

Model yang digunakan pengelihatan mesin biasanya dikembangkan dalam fisik (radiometri, optik, dan desain sensor) dan dalam grafik komputer. Kedua bidang tersebut memodelkan bagaimana objek bergerak, bagaimana cahaya memantul dari permukaannya yang lalu mengalami pembiasan melalui lensa kamera, dan diproyeksikan ke sebuah bidang gambar [9].

2.8 Komputasi Awan

Komputasi awan atau yang biasa disebut *cloud computing* adalah suatu paradigma tentang sistem komputasi yang membahas bagaimana komputer dapat digunakan bersamaan dengan penggunaan internet yang digambarkan sebagai awan (*cloud*) sebagai sumber daya lain untuk data dan pemrograman, yang mana dapat dimanfaatkan untuk memperoleh pemahaman tingkat tinggi dari informasi dan juga sebagai sistem yang menjalankan suatu program tertentu secara otomatis. Dari perspektif teknik, ia adalah suatu komputer yang ada di internet agar sistem yang kompleks

untuk komputasi biasa tersebut menjadi lebih ringan dengan komputasi awan. Hal ini didasari atas kebutuhan untuk memiliki perangkat keras komputer, yang mana dapat diminimalisir dengan penggunaan sebuah komputer yang berupa *server* dari pusat data dan sistem komputer dari *platform* komputasi awan.

Cara kerja dari komputasi awan adalah sistem penyewaan komputer dengan kemampuan tertentu dan spesifikasi tertentu untuk tugas yang akan dijalankan, sehingga salah satu keunggulan dari komputasi awan adalah skalabilitas akan kebutuhan dan harga yang dibayarkan kepada *platform*. Keuntungan lain dari penggunaan komputasi awan adalah kemampuan untuk memilih *service* tertentu untuk penggunaan tertentu. Hal ini dapat tercapai karena setiap *platform* dari komputasi awan pasti memiliki *services* yang ditawarkan untuk pengembang sistem, sehingga pemilihan *services* akan dapat mempermudah untuk pembuatan suatu sistem, dengan biaya dan performa yang sesuai dengan harapan [10].

2.9 Keras

Keras adalah API *neuralnetwork* tingkat tinggi yang ditulis pada Bahasa pemrograman python dan mampu berjalan diatas *platform machine learning* TensorFlow, CNTK, atau Theano. Keras difokuskan untuk memfasilitasi eksperimen sederhana karena mempunyai API yang sangat sederhana dibandingkan API milik *back-end* dari Keras. Keras cocok untuk eksperiman yang cepat dan sederhana, mendukung *convolutional network* dan *recurrent network*, dan mampu berjalan pada CPU maupu GPU [11].

Gambar 2. 12 Logo Keras [11].

2.10 Amazon SageMaker

Amazon SageMaker adalah *cloud service* dari AWS (Amazon Web Services) yang memiliki kemampuan membangun, melatih, dan menerapkan model *machine learning* dengan cepat dan mudah. SageMaker menghilangkan kerja berat di setiap langkah proses pengembangan sistem

machine learning agar lebih mudah dalam mengembangkan model *machine learning* apapun berkualitas tinggi.

Service yang ditawarkan oleh Amazon SageMaker mencakup banyak proses dalam pembuatan model *machine learning*, dari *service* untuk pemberian label data kepada masing-masing dataset yang ada, kemudian pembangunan sistem *machine learning*, pelatihan dan penyesuaian pengaturan terhadap model, hingga penerapannya pada *services-services* yang lain dari AWS ataupun langsung diterapkan kepada perangkat keras tertentu. Seluruh kemudahan ini juga memiliki dukungan API dan *software* library dari TensorFlow, Keras, Scikit-learn, PyTorch, dan lain-lainnya [12].

Gambar 2. 13 Logo Amazon Sagemaker [11].

2.11 Amazon S3

Amazon S3 (Simple Storage Service) adalah *service* keluaran AWS yang berfungsi untuk penyimpanan objek yang menawarkan skalabilitas, ketersediaan data, keamanan, dan performa untuk skala industry, sehingga penggunaan *service* ini bisa dari segala ukuran dan industri untuk menyimpan dan melindungi data sebanyak apa pun untuk berbagai kasus pengunaan, seperti situs web, aplikasi perusahaan, perangkat IoT (*Internet of Things*), dan analisis *big data*.

Amazon S3 memiliki fitur manajemen yang mudah, dari pengaturan data dan konfirmasi kontrol akses yang dapat diatur agar dapat memenuhi kebutuhan bisnis, organisasi, dan kepatuhan tertentu. Amazon S3 dirancang untuk ketahanan data hingga, karena secara otomatis membuat dan menyimpan salinan dari semua data dan objek S3 di beberapa sistem lainnya [13].

Gambar 2. 14 Logo Amazon S3 [13].

2.12 AWS Lambda

AWS Lambda adalah *service* keluaran AWS yang berfungsi untuk menjalankan program-program tertentu pada sistem *cloud computer*, tanpa menyediakan komputer dan *server* tertentu, sehingga pengguna dari AWS Lambda hanya membayar layanan ini dari waktu penggunaannya untuk komputasi.

Gambar 2. 15 Logo AWS Lambda [14].

Dengan AWS Lambda, kita dapat menjalankan kode hampir untuk semua jenis aplikasi atau layanan *backend* – semua tanpa administrasi, dengan cara mengunggah kode program ke AWS Lambda dan AWS Lambda dapat menangani segala yang diperlukan untuk menjalankan dan menskalakan kode-kode program dengan ketersediaan yang disesuaikan. Pengaturan kode untuk secara otomatis memicu dari layanan AWS lainnya atau memanggilnya secara langsung dari web atau aplikasi ponsel [14].

2.13 OpenCV

Open Source Computer Vision Library (OpenCV) merupakan *library* komputer visual dan *machine learning*. OpenCV dibentuk untuk menyediakan infrastruktur umum untuk komputer visi dan mempercepat

produksi pada produk komersial. OpenCV mempermudah pengguna *machine learning* yang *open source* yang mempunyai lisensi BSD. OpenCV mempunyai ribuan library yang telah dioptimalkan dan disederhanakan sehingga lebih mudah dipakai. OpenCV mempunyai *library* seperti pendeteksi objek, mendeteksi wajah, mendeteksi gerakan, menapis warna, mengekstrak sudut, dan masih banyak lagi [15].

Gambar 2. 16 Logo OpenCV [15].

2.14 AWS DeepLens

AWS DeepLens adalah komputer *single board* keluaran AWS yang memiliki tunjangan berupa kamera yang secara umum difungsikan untuk melakukan tugas-tugas *machine learning*. Perangkat ini secara spesifikasi menyerupai Raspberry Pi, hanya saja kemampuan prosesornya dioptimasikan untuk menjalankan program-program *machine learning*. AWS DeepLens juga memungkinkan semua orang dari segala usia untuk menjelajahi komputasi, dan belajar bagaimana memprogram dalam bahasa pemrograman, seperti Python, serta belajar untuk membuat sistem *machine learning* pada perangkat keras. AWS DeepLens mampu melakukan semua yang dapat dilakukan oleh komputer *desktop*, mulai dari menjelajah internet dan memutar video definisi tinggi, hingga membuat *spreadsheet*, melakukan pemrosesan kata, dan bermain game.

Gambar 2. 17 Perangkat AWS DeepLens [16].

Kecepatan prosesor berkisar dari 700 MHz hingga 1,4 GHz dan ukuran RAM sebesar 8 GB. Kamera yang tertanam memiliki resolusi sebesar 4 Mega Pixel dengan MJPEG (Motion JPEG). DeepLens memiliki dua koneksi USB 3, dengan satu USB-nya digunakan untuk registrasi alat di *console* AWS. Untuk output video, mikro HDMI dengan kualitas 4K didukung, dengan jack lengan tip standar 3,5 mm untuk keluaran audio [16].

2.15 *Facial Expression Recogniton*-2013 (FER-2013)

FER-2013 merupakan dataset yang dibangun dari suatu penelitian dan kompetisi di Kaggle, sehingga dataset ini merupakan hasil keluaran sistem CNN yang dibangun pada tahun 2013 untuk tujuan tersebut. FER-2013 terdiri dari 35.685 gambar *grayscale* berukuran 48 x 48 pixel. FER-2013 memiliki gambar wajah-wajah yang telah dianotasi secara otomatis, sehingga wajah yang kurang atau lebih terpusat akan menempati jumlah ruang yang sama pada setiap gambarnya. Kategorisasi setiap wajah berdasarkan emosi yang ditunjukkan dalam ekspresi wajah ke salah satu dari tujuh kategori (0 = Marah, 1 = Jijik, 2 = Takut, 3 = Bahagia, 4 = Sedih, 5 = Kejutan, 6 = Netral).

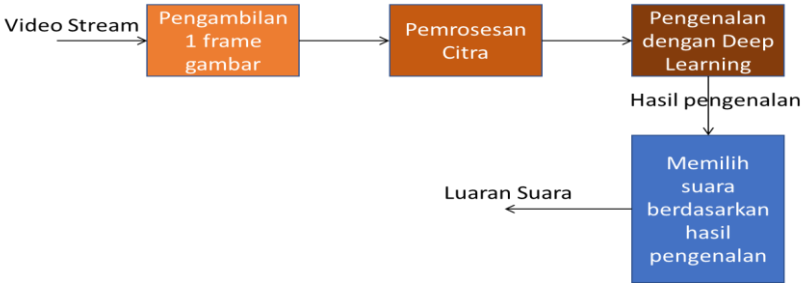
Gambar 2. 18 Contoh Gambar pada Dataset FER-2013 [17].

Set pelatihan terdiri dari 28.709 contoh. Set uji publik yang digunakan untuk kompetisi Kaggle terdiri dari 3.589 contoh. Set tes akhir, yang digunakan untuk menentukan pemenang kompetisi, terdiri dari 3.589 contoh lainnya [17].

Halaman ini sengaja dikosongkan

BAB 3
METODOLOGI PENELITIAN

Pembahasan pada bab ini mengenai perancangan sistem perangkat lunak dan keras. Proses pengerjaan yang dilakukan sesuai dengan apa yang dirancang. Proses pengenalan dan identifikasi ekspresi wajah diawali dengan pengambilan gambar wajah. Gambar yang telah diambil akan diteruskan ke *processing unit*. Sebelum dimasukan ke sistem prediksi, gambar akan diaugmentasi terlebih dahulu. Hasil prediksi akan disampaikan melalui informasi audio. Diagram blok sistem rekognisi ekspresi wajah yang telah didesain dapat dilihat pada Gambar 3.1.



Gambar 3. 1 Diagram Blok Sistem

3.1 Diagram Blok Sistem

Berikut ini adalah penjelasan dari blok diagram dari perangkat rekognisi ekspresi wajah untuk penyandang tunanetra (lihat Gambar 3.1).

1. **Pengambilan Gambar**
   Perangkat akan mengambil citra melalui kamera yang terintegrasi pada perangkat AWS DeepLens. Citra yang ditangkap adalah gambar wajah dari lawan bicara pengguna. Citra tersebut kemudian akan mengalami pengolahan lebih lanjut.
2. **Pra-pemrosesan Gambar**
   Citra yang diterima akan diaugmentasi sebelum dimasukkan pada sistem prediksi. Proses augmentasi yang dimaksud adalah dengan mengubah citra awal yang diambil menjadi *grayscale* dan dilakukan pemotongan gambar pada bagian wajah untuk fokus

pada fitur gambar yang dicari dan menghilangkan fitur-fitur gambar yang kurang relevan. Gambar yang telah dipotong akan ditapis menggunakan *histogram equalization* agar dapat meningkatkan kualitas gambar pada masukan sistem prediksi.

3. **Proses Prediksi**

   Citra yang telah diproses dimasukan pada model *deep learning* untuk mengetahui label ekspresi wajah pada gambar. Keluaran dari model adalah nilai probabilitas tingkat keyakinan (*confidence probability*) pada setiap label ekspresi wajah. Label yang memiliki tingkat keyakinan paling tinggi akan menjadi hasil prediksi.

4. **Pemilihan File Audio**

   Setiap label ekspresi wajah akan direpresentesikan pada file audio yang memiliki komposisi suara berbeda. Informasi akan disampaikan langsung dengan rekaman kata-kata yang sesuai dengan ekspresinya selama satu detik.

Tabel 3. 1 Informasi Audio

Ekspresi	Representasi Bunyi	Nama File
Senang	Suara “Senang”	Senang.wav
Marah	Suara “Marah”	Marah.wav
Sedih	Suara “Sedih”	Sedih.wav
Netral	Suara “Netral”	Netral.wav
Terkejut	Suara “Terkejut”	Terkejut.wav
Jijik	Suara “Jijik”	Jijik.wav
Takut	Suara “Takut”	Takut.wav

5. **Keluaran Suara**

   File audio yang telah dipilih berdasarkan hasil prediksi akan dimainkan. Suara akan disalurkan melalui sebuah *earphone* yang terintegrasi pada perangkat.

3.2 Perancangan *Deep Learning*

Pada penelitian ini, *deep learning* adalah kunci utama dari sistem. Oleh karena itu perlu diadakan penelitian yang lebih mendetail agar mendapat model *deep learning* dengan performa baik dan paling optimum. Untuk melatih model *deep learning* diperlukan komputer denganpada penelitian ini digunakan sebuah layanan pada AWS yang disebut AWS SageMaker. Layanan inilah yang menjadi dasar perancangan, pelatihan dan optimalisasi model *deep learning*. Dalam layanan AWS SageMaker, penentuan atas CPU dan GPU tertentu menentukan performa pelatihan yang optimal dan biaya untuk jasa komputasi pada sistem komputasi awan, sehingga pemilihan spesifikasi komputer tertentu menentukan lama waktu, kemampuan *prototyping*, serta biaya yang dikeluarkan untuk membuat model *deep learning*.

Spesifikasi komputer yang dipilih pada layanan AWS SageMaker untuk penelitian ini dapat dilihat pada Tabel 3.2.

Tabel 3. 2 Spesifikasi Komputer Uji [11]

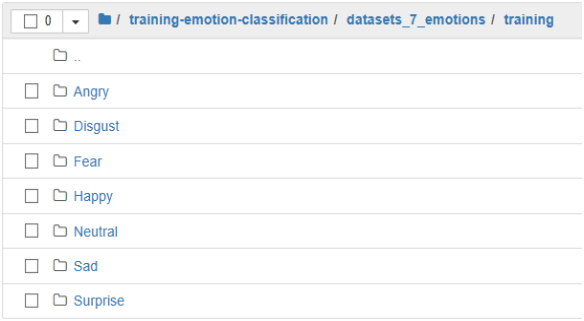
Jenis Komputer	ml.c5.2xlarge (*Computing Optimized*)
CPU	8 Core
RAM	16 GB
Memori Penyimpanan	5 GB
Jaringan Koneksi Internet	Maksimum 10 Gbps

3.2.1 Pengelolaan Dataset

Faktor terbesar dari kualitas model deep learning adalah dataset. Semakin representatif dataset yang digunakan, semakin baik performa model. Dataset yang digunakan adalah FER-2013 [18]. Dataset ini dibutuhkan untuk pelatihan algoritma *deep learning* yang akan diimplementasikan ke sistem pada penelitian ini.

Gambar-gambar yang terdapat pada dataset FER-2013 telah dilakukan *labelling* sesuai dengan jenis emosi sebanyak 7 macam, yaitu sedih, senang, takut, jijik, marah, terkejut, dan netral, dengan cara menyimpan gambar pada *folder* yang judulnya sesuai dengan ekspresi wajah. Total gambar yang ada pada dataset ini kurang lebih 28.000 gambar.

Untuk memastikan kualitas dataset, maka dilakukan pengecekan terhadap gambar satu-per-satu pada masing-masing *folder*. Gambar dengan ekspresi wajah kurang sesuai dengan judul *folder* letak penyimpanannya akan dipindahkan ke *folder* yang sesuai. Apabila terdapat gamabr dari ekspresi wajah yang ambigu, maka gambar tersebut akan dihapus. Pada tahap ini layanan Amazon SageMaker *labbeling jobs* digunakan dalam mengklasifikasikan ekspresi wajah yang terdapat pada dataset.



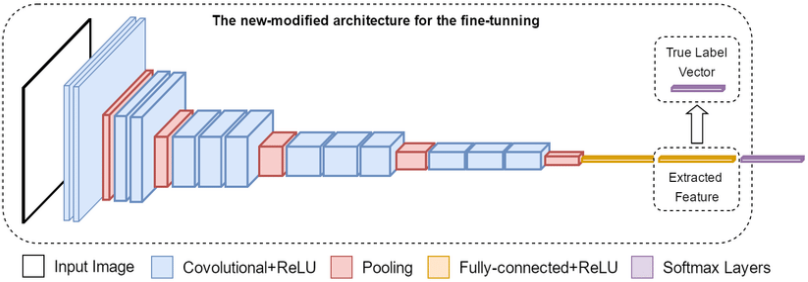
Gambar 3. 2 Struktur Penyimpanan Dataset.

3.2.2 Pelatihan Model *Deep Learning*

Pada bagian ini akan dijelaskan proses dalam melatih model *deep learning*. Dalam penelitian ini akan dilakukan pencarian performa model dengan variabel *learning rate*, *decay*, dan *epoch*. Data penting yang akan diambil adalah akurasi dan kecepatan prediksi dari model. Arsitektur CNN yang dipakai adalah VGG16 dengan *pretrained weights*. Arsitektur VGG 16 yang digunakan dimodifikasi pada bagian fungsi aktivasi, dimana fungsi aktivasi yang digunakan adalah ELU (*Exponential Linear Unit*). Pemilihan arsitektur VGG16 didasari atas ketersediaan dataset dengan resolusi kecil dan karena *library* AWS juga mendukung arsitektur VGG-16 dengan Keras. Secara spesifik, arsitektur VGG-16 dapat dilatih dengan gambar dengan

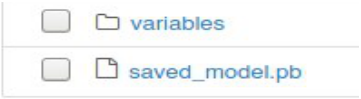
resolusi rendah, sehingga kecepatan *training* relatif lebih cepat dan bisa didapatkan hasil yang lebih baik, walaupun menggunakan fungsi aktivasi ELU yang lebih memakan waktu untuk komputasi perhitungan *training*. Untuk mendapatkan kecepatan pelatihan dan prediksi model yang lebih baik, maka digunakanlah arsitektur CNN VGG16 yang telah dimodifikasi yang dilampirkan pada Lampiran A nomor 5.

Keluaran dari arsitektur tersebut mempunyai *classes* yang cukup sederhana, sehingga keluarannya langusng digunakan sebagai acuan target keluaran. Masukan dari model *deep learning* ini adalah gambar berwarna dengan resolusi 48x48 pixel. Gambar arsitektur *deep learning* yang digunakan dapat dilihat pada gambar 3.3. Keluaran dari pelatihan model adalah *model weights* dengan format HDF5 (.h5) dan Json (.json). Keluaran tersebut harus dikonversikan menjadi *file* dengan format *Protocol Buffer* (.pb dan .pbtxt), sesuai dengan yang format keluaran pelatihan dari TensorFlow dan format yang didukung oleh sistem Amazon SageMaker. Setelah konversi hasil pelatihan, diperlukan sebuah cara untuk mengkonversikan model *deep learning* tersebut menjadi sebuah *trigger* agar dapat diproses di AWS Lambda. Hasil konversi tersebut disimpan pada *bucket* S3 yang telah dibuat secara otomatis.



Gambar 3. 3 Ilustrasi Arsitektur VGG16 termodifikasi. [7]

Gambar 3. 4 Keluaran dari Pelatihan Model *Deep Learning*.



Gambar 3. 5 Konversi Hasil Keluaran Pelatihan Deep *Learning*.

3.2.3 Kecepatan Prediksi Model *Deep Learning*

Sistem yang diajukan pada penelitian ini menggunakan CNN. CNN dikenal memerlukan kekuatan komputasi yang cukup besar sehingga dapat memperlambat kerja dari suatu sistem. Oleh karena itu, perlu dilakukan percobaan untuk menemukan model yang sesuai dengan aplikasi pada penelitian ini. Untuk memilih model yang sesuai, dilakukan uji kecepatan dari 2 arsitektur CNN yaitu VGG-16, dan VGG-16 yang dimodifikasi pada fungsi aktivasinya. Arsitektur tersebut dilatih pada sebuah dataset tanpa melakukan *tuning* pada parameter *learning*. Pengujian dilakukan dengan memprediksi 50 gambar dengan memperhatikan kecepatan dan akurasi prediksi pada komputer uji dan Jetson Nano (lihat tabel 3.3).

Tabel 3. 3 Hasil Pengujian pada Komputer Uji dan Jetson Nano

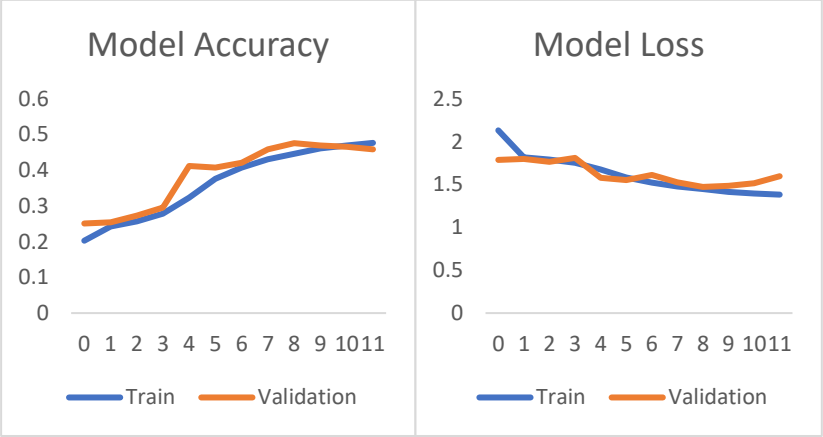
Arsitektur		**Kecepatan (detik/prediksi)**	**Akurasi**
Komputer Uji	VGG-16	0.05123	62 %
	VGG-16 Modifikasi	0,02845	70 %
Jetson Nano	VGG-16	6,218	58 %
	VGG-16 Modifikasi	1,362	65 %

Pada pengujian terhadap komputer uji, perbedaan akurasi dan kecepatan tidak terlalu signifikan. Bila kecepatan model kurang dari 0.1 detik, maka model tersebut sudah sangat baik untuk memeprediksi gambar secara kontinu. Tidak ada perbedaan akurasi secara signifikan dari kedua arsitektur tersebut. Akan tetapi, terdapat perbedaan yang sangat signifikan pada kecepatan pada Jetson Nano. Berdasarkan pengamatan, VGG-16 modifikasi sangat unggul pada kecepatan prediksi meskipun belum bisa dibilang mempunyai kecepatan yang baik. Akan tetapi, dalam aplikasi penelitian ini kecepatan tersebut sudah memenuhi. Dalam aspek akurasi, ke tiga model mempunyai kecepatan yang relatif sama. Dari kedua arsitektur, VGG-16 modifikasi yang paling sesuai untuk digunakan pada penelitian ini.

3.2.4 *Hyperparameter Tuning*

Pada penelitian ini akan dilakukan pengaturan parameter *learning*. Parameter yang akan diperhatikan adalah *learning rate* dan *epoch* maksimum. Pada pengujian ini, model VGG16 akan dilatih menggunakan dataset FER-2013. Pertama, model akan dilatih beberapa kali dengan parameter *learning rate* acak untuk mendapatkan acuan nilai *learning rate* yang optimal dengan batasan *epoch* maksimum sebesar 25 *epoch*. Setelah menemukan nilai acuan yang optimal, maka dilakukan *learning model* dengan nilai acuan yang telah ditemukan sebelumnya.

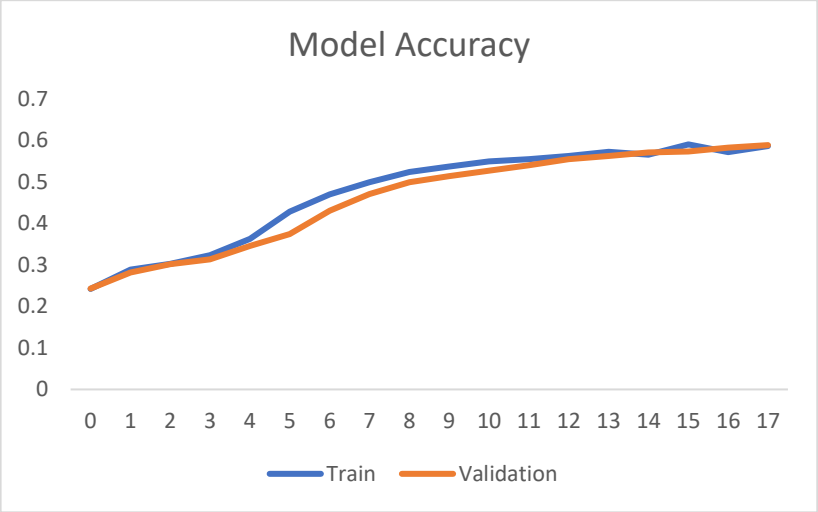
Pada gambar 3.6, terlihat bahwa *momentum* 0.6 dan *learning rate* sebesar 0.001 dapat menghasilkan nilai akurasi 47.97% dan *loss* sebesar 1.4%. Hal ini dapat dikatakan bagus, mengingat dataset yang hanya sebanyak 28.000 sampel dan terbatas hanya 11 *epoch* dari batasan *epoch* maksimum 12 *epoch.* Untuk hasil yang lebih *robust*, maka diperlukan nilai perkiraan ini agar didapatkan hasil yang lebih maksimal untuk *learning* yang menjadi model akhir yang lebih *robust*.



Gambar 3. 6 Riwayat Pembelajaran dengan *Momentum* 0.6 dan *Learning Rate* 0.001.

Setelah mendapatkan acuan nilai variabel *momentum* dan *learning rate* akan dilakukan sesi *learning* ulang. Model dari sesi *learning* ini akan menjadi model akhir yang akan diimplementasikan ke dalam sistem perangkat lunak yang dirancang apabila mencapai batas minimum akurasi yang diinginkan, yaitu 50% dari data tes. Harapan dari *hyperparameter*

tuning adalah mendapatkan model yang akurasinya lebih dari batas minimum. Gambar 3.7 adalah riwayat sesi training yang sudah dilakukan *hyperparameter tuning* dengan nilai variabel *momentum* dan *learning rate* masing masing 0,7 dan 0,0002. Akurasi data latih hampir mencapai 60 %. Bila diperhatikan, akurasi data tes sudah melebihi batas minimum yaitu mendekati ambang batas 70%. Oleh karena itu, model dari sesi *learning* ini dapat dipakai sebagai model akhir yang akan diimplementasikan pada sistem.

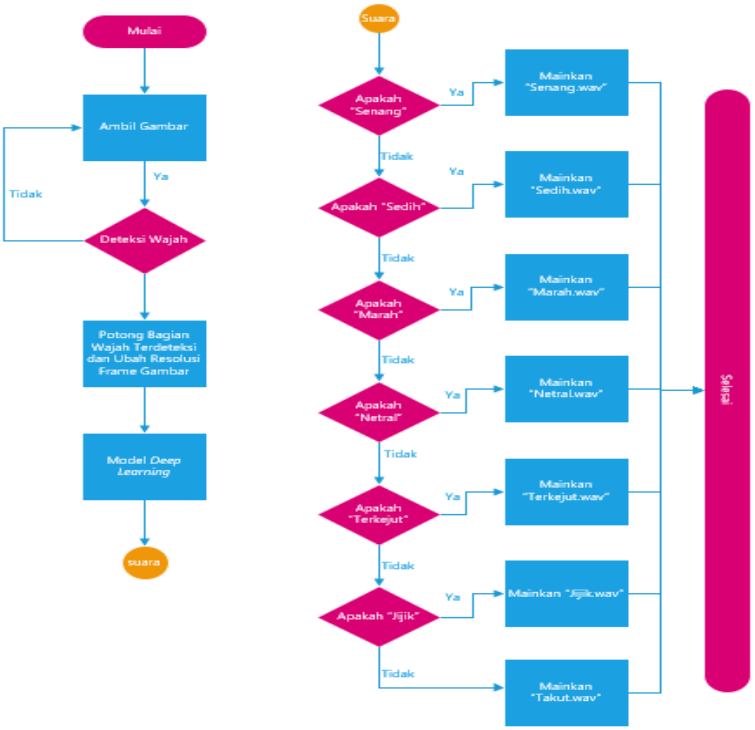


Gambar 3. 7 Hasil Akhir Model *Deep Learning* yang digunakan.

3.3 Perancangan Perangkat Lunak

Pada sub-bab ini akan dibahas bagaimana proses pembuatan perangkat lunak sistem yang diimplementasikan pada AWS DeepLens. Untuk implementasi sistem, perlu dibuat *script* dengan bahasa pemrograman python versi 3.7 pada layanan AWS Lambda, sehingga *script* yang dibuat tersebut dapat dipanggil untuk diunduh pada divais AWS DeepLens melalui layanan AWS Lambda. Proses eksekusi agar model *deep learning* dapat dipanggil instansi-nya pada AWS Lambda adalah dengan mengkonversikan hasil pelatihan dalam format .pb yang terletak di Amazon S3. Kemudian, setelah hasil pelatihan dapat dikonversikan sebagai *trigger* untuk AWS Lambda, maka instansi Amazon S3 dapat dipanggil untuk memprogram di AWS Lambda Gambar 3.9 adalah diagram alir dari algoritma *script* yang

akan dibuat pada AWS Lambda. Program yang akan dibuat tersebut akan di impor ke divais AWS DeepLens secara otomatis dengan menghubungkan divais dengan *console* AWS DeepLens dengan AWS Lambda yang ada.



Gambar 3. 8 Algoritma Program Utama pada AWS DeepLens.

Dependencies yang dibutuhkan adalah OpenCV, Numpy, Keras, dan PyAudio. Script akan berjalan otomatis pada saat AWS DeepLens menyala. Gambar yang diambil oleh kamera akan diproses pada python dengan bantuan OpenCV. OpenCV dan Haar *cascade* dapat mendeteksi wajah manusia pada gambar tersebut. Bila terdeteksi, wajah dengan luasan terbesar akan diambil dan ditapis oleh tapis *histogram equalization* sesuai bawaan OpenCV. Resolusinya diturunkan menjadi 48x48 piksel, sehingga gambar dapat masuk ke model *deep learning*. Setiap label pada keluaran *deep learning* akan diprogram dengan suara tertentu. Hasil prediksi akan disajikan

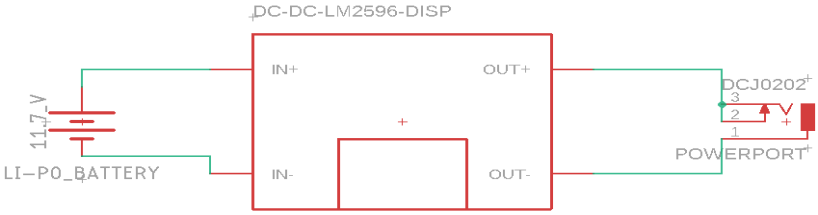
dalam bentuk file audio untuk setiap ekspresi wajah akan ditugaskan untuk file audio yang berbeda yang dibuat dengan merekam suara sesuai dengan ekspresinya dalam bentuk wav. Daripada menggunakan nada yang berbeda untuk tiap ekspresi wajah, menggunakan suara ekspresi secara langsung terasa lebih langsung tanpa perlu meningat-ingat nada yang dikodekan secara spesifik. Tabel 3.1 menunjukan representasi audio pada setiap ekspresi wajah yang dirancang untuk penelitian ini.

3.4 Perancangan Perangkat Keras

Perangkat keras pada penilitian ini berupa desain elektronika suplai daya dan skenario penggunaannya. Pada subbab ini akan dijelaskan rancangan awal dari perangkat ini.

3.4.1 Spesifikasi Elektronika

AWS DeepLens sebagai kontroller utama pada sistem ini. Hal ini dikarenakan AWS DeepLens memiliki performa yang baik dan reliabel dalam urusan *deep learning*, integrasi antara sistem komputasi awan AWS dengan perangkat keras, dan kamera yang telah terintegrasi dengan perangkat. Pada *audio jack* 3.5 mm akan dipasang *earphone* atau *headphone*.



Gambar 3. 9 Rangkaian pada Sumber Listrik.

Sumber listrik yang terdapat bersama dengan perangkat AWS DeepLens memiliki adaptor dari listrik AC 220 Volt/50 Hertz menjadi listrik DC 12 Volt dengan arus maksimum 2 Ampere, sehingga untuk perangkat ini sumber listriknya dimodifikasi menggunakan baterai Li-Po 3 sel sebesar 11,7 Volt yang arusnya diturunkan oleh *buck converter*. Tegangan yang dibutuhkan untuk menjalankan AWS DeepLens adalah sebesar 12 Volt dengan toleransi sebesar 5 persennya, sehingga AWS DeepLens dapat berjalan dengan tegangan minimum 11,4 Volt hingga 12,6 Volt. Tegangan LiPo yang sebesar 11,7 Volt. Keluaran dari *buck converter* akan dipasang *adaptor jack* yang sesuai dengan *port power* pada AWS DeepLens.

3.4.2 Skenario Pemakaian

Perangkat berbentuk penyangga yang menyerupai rompi. Citra yang ditangkap digunakan sebagai input utama dari sistem ini sehingga penempatannya sangat diperhatikan. *Earphone* digunakan sebagai penyampaian informasi yang didapatkan oleh sistem. Informasi tersebut berupa nada yang diatur sedemikian rupa agar penyandang tunanetra dapat membedakan emosi satu dengan yang lainnya.

Gambar 3. 10 Skenario Penggunaan Perangkat.

Halaman ini sengaja dikosongkan

BAB 4
HASIL PENGUJIAN

Pada penelitian ini, perangkat akan diujikan kepada dua kategori pengguna, yakni pengguna dengan pengelihatan normal dan tunanetra. Perangkat yang digunakan adalah Nvidia Jetson Nano dan kamera Logitech C-925e dengan program yang sama seperti pada AWS DeepLens, hanya hasil pelatihan model *deep learning* yang digunakan pada pengujian berasal dari Amazon SageMaker. Pengujian ini diharapkan dapat menemukan korelasi dari kategori pengguna alat dengan kemampuan menerima informasi suara. Skenario pengujian akan diperhatikan agar dapat menganalisa data hasil pengujian secara komprehensif.

4.1 Pengujian Perangkat pada Pengguna dengan Penglihatan Normal

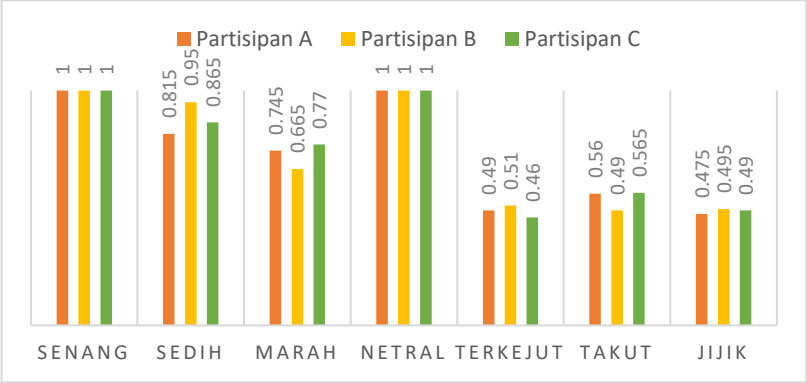
Dalam sub-bab ini, alat akan diujikan kepada pengguna dengan pengelihatan normal pada kondisi lingkungan yang berbeda (ruangan terbuka dan ruangan tertutup).

4.1.1. Pengujian Sistem di dalam Ruangan

Dalam sub-bab ini akan diujikan penggunaan sistem di dalam ruangan. Dalam pengujian ini, diundang 3 partisipan sebagai pengguna perangkat dan dua orang merangkap sebagai ekspressor. Partisipan yang bertindak sebagai ekspresor akan memperagakan 7 ekspresi wajah sebanyak 2 kali. Pengguna perangkat akan menebak ekspresi yang dilakukan oleh 2 orang ekspresor.

Gambar 4. 1 Pengujian Sistem dalam Ruangan.

Sebelum memulai pengujian, partisipan akan ditunjukan informasi suara pada setiap ekspresi dan gambar ekspresi wajah yang ada pada dataset sebagai referensi dalam mempragakan tiap ekspresi wajah. Ekspresor akan duduk bersebelahan tanpa kontak mata dengan pengguna perangkat. Pengguna perangkat akan menyebutkan informasi ekspresi wajah yang terdeteksi melalui *earphone*. Persentase keberhasilan dan kesalahan pengguna perangkat akan dicatat. Data yang telah diambil dari setiap pengguna perangkat akan dijadikan satu dan dipresentasikan dengan *confusion matrix*.



Gambar 4. 2 Diagram Persentase Keberhasilan Prediksi pada Pengujian di dalam Ruangan.

	Happy	Sad	Angry	Neutral	Surprised	Fear	Disgust
Happy	1.00	–	–	–	–	–	–
Sad	–	0.88	0.12	–	–	–	–
Angry	–	0.11	0.73	0.03	0.13	–	–
Neutral	–	–	–	1.00	–	–	–
Surprised	0.13	–	0.23	–	0.49	–	0.15
Fear	–	0.09	–	0.13	–	0.54	0.24
Disgust	–	0.22	–	0.10	0.04	0.16	0.48

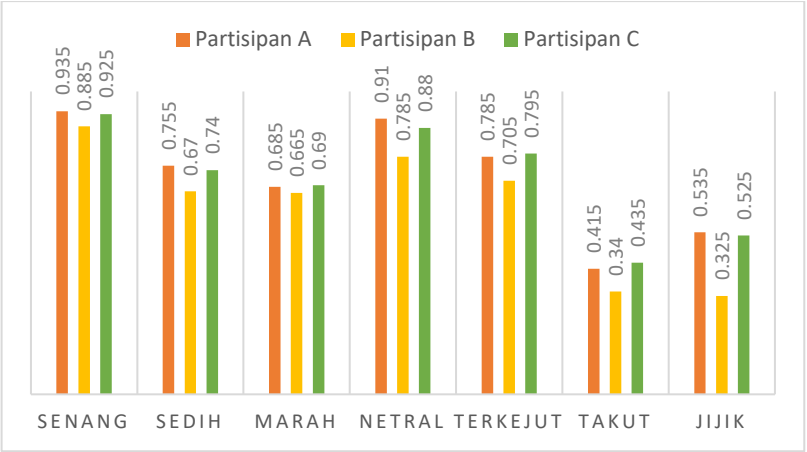
Gambar 4. 3 *Confusion Matrix* dari Hasil Pengujian Sistem dalam Ruangan.

Harapan dari pencahayaan yang bagus di dalam ruangan tanpa penggunaan bantuan pencahayaan lainnya adalah agar mendapat akurasi prediksi yang baik. Pada Gambar 4.2, persentasi keberhasilan ketiga subjek tidak jauh berbeda. Perbedaan yang paling terlihat ada pada ekspresi terkejut, jijik, dan takut, akan tetapi perbedaan tersebut tidak terlalu berpengaruh ke performa perangkat secara keseluruhan. Pada Gambar 4.3, ekspresi netral dan senyum tidak pernah mengalami kesalahan saat prediksi. Ekspresi terkejut memiliki kesalahan prediksi terbanyak pada pengujian ini. Keselahan prediksi pada setiap ekspresi dicatat dengan harapan menemukan suatu pola kesalahan dan penyebabnya. Pada kondisi dalam ruangan, sistem mempunyai akurasi prediksi total 72,95 % dan dapat dikatakan sistem telah berjalan dengan optimal.

4.1.2 Pengujian Sistem di luar Ruangan

Dalam sub-bab ini sistem akan diuji di luar ruangan untuk mensimulasikan keadaan lingkungan dengan pencahayaan yang tidak optimum dan konsisten. Dalam pengujian ini, diundang 3 partisipan sebagai pengguna perangkat dan dua orang merangkap sebagai ekspressor. Partisipan yang bertindak sebagai ekspresor akan memperagakan 7 ekspresi wajah sebanyak 2 kali. Pengguna perangkat akan menebak ekspresi yang dilakukan oleh 2 orang ekspresor. Pengguna perangkat akan menyebutkan informasi ekspresi wajah yang terdeteksi melalui *earphone*. Data yang telah diambil dari setiap pengguna perangkat akan dijadikan satu dan dipresentasikan dengan *confusion matrix*.

Gambar 4. 4 Pengujian Sistem di luar Ruangan.



Gambar 4. 5 Diagram Persentase Keberhasilan Prediksi pada Pengujian di luar Ruangan.

	Happy	Sad	Angry	Neutral	Surprised	Fear	Disgust
Happy	0.92	–	–	0.07	–	0.01	–
Sad	–	0.72	0.13	–	–	–	0.15
Angry	–	0.07	0.68	–	0.20	0.05	–
Neutral	–	–	–	0.86	–	0.06	0.08
Surprised	–	–	–	0.18	0.76	0.02	0.04
Fear	–	0.36	–	0.10	0.14	0.40	–
Disgust	–	–	0.27	–	0.13	0.14	0.46

Gambar 4. 6 *Confusion Matrix* dari Hasil Pengujian Sistem luar Ruangan.

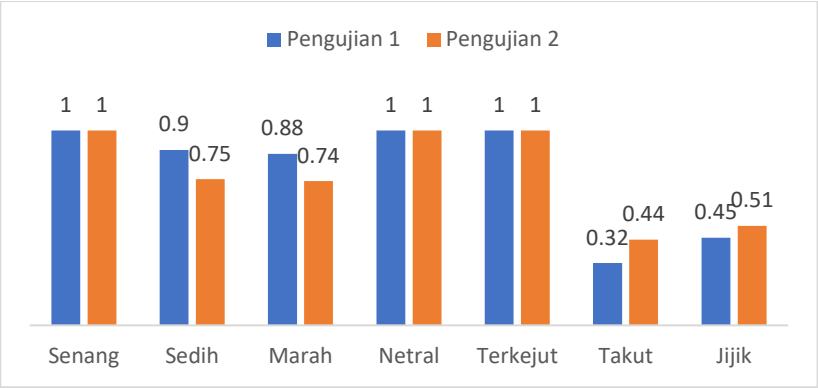
Pada Gambar 4.5, tingkat keberhasilan prediksi dari kedua pengguna perangkat ada perbedaan di beberapa kategori ekspresi. Akan tetapi kedua subjek memiliki *trend* keberhasilan yang serupa. Dilihat pada Gambar 4.6, ekspresi senyum memiliki tingkat kesalahan prediksi paling rendah, kemudian diikuti oleh ekspresi netral. Ekspresi takut memiliki kesalahan

prediksi terbanyak pada pengujian ini. Akan tetapi, ekspresi jijik mengalami peningkatan kecil dibandingkan dengan pengujian di dalam ruangan. Untuk ekspresi lainnya, terdapat kesalahan prediksi tetapi masih pada batas wajar. Pada kondisi luar ruangan, sistem mempunyai akurasi prediksi total 68,50 % dan dapat dikatakan sistem berjalan dengan baik.

4.2 Pengujian Alat pada Pengguna Tunanetra

Dalam sub-bab ini, alat akan diuji ke pengguna tunanetra pada kondisi lingkungan tertutup dan diukur akurasi prediksinya. Pengujian ini dilakukan oleh seorang penyandang tunanetra sebagai pengguna perangkat dan seorang dengan pengelihatan normal sebagai objek untuk direkognisi ekspresi wajahnya. Partisipan akan memperagakan 7 ekspresi wajah sebanyak 2 kali kepada pengguna perangkat. Perangkat akan dipakai di ruangan tertutup tanpa tambahan lampu. Sebelum pengujian dimulai, pengguna perangkat dilatih terlebih dahulu untuk mengenali informasi suara dari earphone selama 10 menit. Partisipan yang berperan sebagai peraga ekspresi wajah akan duduk di sebelah pengguna perangkat. Peraga akan diperlihatkan beberapa contoh ekspresi wajah yang ada pada dataset sebagai referensi. Pengguna perangkat akan menebak ekspresi wajah peraga berdasarkan infromasi suara yang didengar. Penguji akan mencatat setiap tebakan salah maupun benar agar dapat menyajikan data akhir sebagai *confusion matrix*. Setelah pengujian diadakan sesi timbal balik pengguna tentang perangkat yang diuji.

Gambar 4. 7 Pengujian Sistem pada Pengguna Tunanetra.



Gambar 4. 8 Diagram Persentase Keberhasilan Prediksi pada Pengujian kepada Pengguna Tunanetra.

	Happy	Sad	Angry	Neutral	Surprised	Fear	Disgust
Happy	1.00	–	–	–	–	–	–
Sad	–	0.83	–	0.06	0.08	0.03	–
Angry	–	0.13	0.81	0.06	–	–	–
Neutral	–	–	–	1.00	–	–	–
Surprised	–	–	–	–	1.00	–	–
Fear	–	–	0.08	0.16	0.22	0.38	0.14
Disgust	–	–	0.09	0.07	0.12	0.24	0.48

Gambar 4. 9 *Confusion Matrix* dari Hasil Pengujian Sistem pada Pengguna Tunanetra.

Pada gambar 4.8, subjek memiliki tingkat keberhasilan yang relatif sama antara pengujiannya. Akan tetapi, terdapat perbedaan yang signifikan pada pengenalan ekspresi takut dan jijik. Setelah diadakan tanya jawab pada subjek, dijelaskan bahwa subjek kurang menguasai pada pengenalan informasi melalui suara. Namun, subjek merasa lebih nyaman setelah pengambilan data pada ekspresor kedua. Dilihat dari Gambar 4.9, pengguna

berhasil menebak ekspresi wajah dengan benar sebesar 78,50% dengan kondisi ruangan yang alami tanpa adanya tambahan pencahayaan. Ekspresi takut menjadi ekspresi wajah yang memiliki kesalahan yang paling besar. Pada ekspresi jijik dan takut, galat dari jawaban selalu berada di ekspresi marah. Hal ini diduga dikarenakan ekspresi marah dan jijik memiliki fitur wajah yang mirip atau keterbatasan dataset pada ekspresi jijik dan takut. Ekspresi lainnya memiliki kesalahan yang masuk dalam batas toleransi. Secara kesuluruhan perangkat dapat dikatakan berfungsi dengan baik.

Halaman ini sengaja dikosongkan

BAB 5
KESIMPULAN

5.1 Kesimpulan

Dari hasil penelitian dan analisis komprehensif berkaitan dengan sistem dan alat, sementara ini bisa didapatkan beberapa kesimpulan, diantaranya:

1. Rerata sistem pengenalan ekspresi wajah yang diujikan mencapai pengenalan minimum 73,43 % dari model yang dilatih pada Amazon SageMaker.
2. Infrastruktur komputasi awan AWS, spesifik pada Amazon SageMaker, AWS Lambda, dan divais AWS DeepLens hanya mendukung model *deep learning* terbatas.
3. AWS DeepLens memiliki kelebihannya sebagai divais terintegrasi. namun, AWS DeepLens hanya dapat bekerja dengan *platform* AWS dan bergantung hanya pada layanan AWS Lambda.
4. Diperlukan pengetahuan dan pengalaman yang komprehensif untuk mengembangkan dan mengimplementasikan sistem komputasi awan secara efektif.

5.2 Saran

1. Akurasi sistem dapat ditingkatkan dengan melakukan pelatihan sistem dengan dataset yang lebih banyak.
2. Dalam pemilihan *platform* komputasi awan, dapat dipertimbangkan untuk memilih *platform* yang sekiranya sesuai dengan kemudahan integrasinya dan keterbukaannya.
3. Suara keluaran dari sistem dapat dibuat lebih *robust* dengan kode-kode suara atau nada-nada sederhana
4. *Tuning Parameter* dapat diperdalam pada bagian *learning rate* dan *decay*.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] Buduma, Nikhil, dan Nicholas Locasio, “Fundamentals of Deep Learning”, USA, O’Reilly, 2017

[2] Hung, C.-Y., Chen, W.-C., Lai, P.-T., Lin, C.-H., & Lee, C.-C., “Comparing deep neural network and other machine learning algorithms for stroke prediction in a large-scale population-based electronic medical claims database, ” 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2017

[3] Jin, Yongsik, Jonghong Kim, Bumwhi Kim, Rammohan Mallipeddi, dan Minho Lee, “Smart Cane: Face Recognition System for Blind”, Confrence on Human-Agent Interaction, Daegu, Oktober, 2015

[4] Buimer HP, Bittner M, Kostelijk T, van der Geest TM, Nemri A, van Wezel RJA, et al. "Conveying Facial Expressions to Blind and Visually-Impaired Persons through a Wearable Vibrotactile Device", *PLoS one*, Maret 2018.

[5] Fahrudin, Hasby., "Pengenalan Ekspresi Wajah untuk Tunanetra menggunakan Deep Learning Pada Perangkat Portabel," Tugas Ahir Departemen Teknik Elektro ITS , 2019.

[6] Nash, Ryan, dan Keiron O'Shea , "An Introduction to Convolutional Neural Network", ResearchGate, Desember, 2015

[7] Simonyan, Karen, dan Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", arXiv:1409.1556, September, 2014

[8] Apache MXNet Developer Team, "MXNet Getting Started Guide," Available: https://mxnet.apache.org/. [Accessed : 1-Juni-2020].

[9] Szeliski, Richard, "Computer Vision: Algorithm and Applications", Springer, 2010

[10] Erl, Thomas., Zaigham Mahmood, dan Ricardo Puttini, "Cloud Computing: Concepts, Technology & Architecture," *Pearson*, Mei, 2013.

[11] Keras, “Keras Documentation”. [Online]. Available: https://keras.io/. [Accessed: 18-May-2019].

[12] AWS Developer Team, "Amazon SageMaker Developer Guide," Available: https://docs.aws.amazon.com/sagemaker. [Accessed : 10-April-2020].

[13] AWS Developer Team, "Amazon S3 Developer Guide," Available: https://docs.aws.amazon.com/s3/index.html. [Accessed : 02-May-2020].

[14] AWS Developer Team, "AWS Lambda Developer Guide," Available: https://docs.aws.amazon.com/lambda. [Accessed : 17-April-2020].

[15] OpenCV Developer Team, “About OpenCV,” 2017. [Online]. Available: http://opencv.org/about.html. [Accessed: 21-May-2019]

[16] AWS Developer Team, "AWS DeepLens Developer Guide," Available: https://docs.aws.amazon.com/deeplens/dg/. [Accessed : 18-April-2020]

[17] Kaggle, "Challenges in Representation Learning: Facial Expression Recognition Challenge," Available: https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/. [Accessed: 13-May-2020]

LAMPIRAN A

1. Program Latih pada Amazon SageMaker

```
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation,
Flatten, BatchNormalization
from keras.layers import Conv2D, MaxPooling2D

from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint,
EarlyStopping, ReduceLROnPlateau
import os

num_classes = 7
img_rows, img_cols = 48, 48
batch_size = 32

dir_path = os.getcwd()

train_data_dir = dir_path +
'/datasets_7_emotions/training'
validation_data_dir = dir_path +
'/datasets_7_emotions/validation'

train_data_generator = ImageDataGenerator(rescale=1. /
255,
rotation_range=30, shear_range=0.3,zoom_range=0.3,
width_shift_range=0.4,
height_shift_range=0.4,horizontal_flip=True,
fill_mode='nearest')

validation_data_generator = ImageDataGenerator(rescale=1.
/ 255)

train_generator =
train_data_generator.flow_from_directory(train_data_dir,
color_mode='grayscale', target_size=(img_rows, img_cols),
batch_size=batch_size, class_mode='categorical',
shuffle=True)
```

```
validation_generator =
validation_data_generator.flow_from_directory(validation_d
ata_dir,
color_mode='grayscale', target_size=(img_rows, img_cols),
batch_size=batch_size,
class_mode='categorical',shuffle=True)

model = Sequential()


# Feature Learning Layer 0
model.add(Conv2D(32, (3, 3), padding='same',
kernel_initializer='he_normal', input_shape=(img_rows,
img_cols, 1)))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Conv2D(32, (3, 3), padding='same',
kernel_initializer='he_normal', input_shape=(img_rows,
img_cols, 1)))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))

# Feature Learning Layer 1
model.add(Conv2D(64, (3, 3), padding='same',
kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Conv2D(64, (3, 3), padding='same',
kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))

# Feature Learning Layer 2
model.add(Conv2D(128, (3, 3), padding='same',
kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), padding='same',
kernel_initializer='he_normal'))
```

```
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))
# Feature Learning Layer 3
model.add(Conv2D(256, (3, 3), padding='same',
kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Conv2D(256, (3, 3), padding='same',
kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))
# Feature Learning Layer 4
model.add(Flatten())
model.add(Dense(64, kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

# Feature Learning Layer 5
model.add(Dense(64, kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

# Feature Learning Layer 6
model.add(Dense(num_classes,
kernel_initializer='he_normal'))
model.add(Activation('softmax'))

print(model.summary())
```

2. Program pemanggilan dataset pada Amazon SageMaker

```
import os
import zipfile
dir_path = os.getcwd()

file = dir_path + ‘/datasets_7_emotions.zip’
```

```
path = dir_path
if os.path.isdir(dir_path + '/datasets_7_emotions'):
    print ('Datasets already extracted in ' + dir_path +
'/datasets_7_emotions')
else:
    if os.path.isfile(file):
        with zipfile.ZipFile(file, 'r') as zip_ref:
            zip_ref.extractall(path)
    else:
        print ('File ' + 'dataset_7_emotions.zip' + ' not
exist!')
```

3. Program konversi model pada Amazon SageMaker

```
import os
import boto3, re
import tarfile
import sagemaker

from time import gmtime, strftime

from tensorflow.python.util import deprecation
deprecation._PRINT_DEPRECATION_WARNINGS = False

from tensorflow.python.keras.models import model_from_json
from tensorflow.python.saved_model import builder
from tensorflow.python.saved_model.signature_def_utils
import predict_signature_def
from tensorflow.python.saved_model import tag_constants
from tensorflow.python.keras import backend as K

from sagemaker import get_execution_role
from sagemaker.tensorflow.serving import Model

role = get_execution_role()
json_file = 'emotion_classification_vgg_7_emotions.json'

def load_json_model(path):
    json_file = open(path)
    loaded_model_json = json_file.read()
    json_file.close()
    loaded_model = model_from_json(loaded_model_json)
```

```
    return loaded_model

dir_path = os.getcwd()
model_json_path = dir_path + ‘/keras_model’ + ‘/’ +
json_file

if os.path.isdir(dir_path + ‘/keras_model’):
    !ls keras_model

    if os.path.isfile(model_json_path):
        loaded_model = load_json_model(model_json_path)
    elif os.path.isfile(dir_path + ‘/’ + json_file):
        print (‘Moving \’’+ json_file +’\’ to
\’/keras_model\’ directory..’)
        !mv $json_file keras_model

        loaded_model = load_json_model(model_json_path)
    else:
        print(‘Please do the training process first to get
\’model.json\’ file!’)
else:
    if os.path.isfile(dir_path + ‘/’ + json_file):
        print (‘Directory ‘ + dir_path + ‘/keras_model’ +
‘ not exist!’)
        print (‘Creating \’/keras_model\’ directory..’)
        !mkdir keras_model

        print (‘Moving \’’+ json_file +’\’ to
\’/keras_model\’ directory..’)
        !mv $json_file keras_model

        loaded_model = load_json_model(model_json_path)
    else:
        print(‘Please do the training process first to get
\’’+ json_file +’\’ file!’)

def load_h5_model(path):
    loaded_model.load_weights(path)
    print(“Loaded model from disk.”)

dir_path = os.getcwd()
model_h5_path = dir_path + ‘/keras_model’ + ‘/’ + h5_file
```

```
if os.path.isdir(dir_path + '/keras_model'):
    !ls keras_model

    if os.path.isfile(model_h5_path):
        load_h5_model(model_h5_path)
    elif os.path.isfile(dir_path + '/' + h5_file):
        print ('Moving \''+ h5_file + '\' to
\'/keras_model\' directory..')
        !mv $h5_file keras_model

        load_h5_model(model_h5_path)
    else:
        print('Please do the training process first to get
\'emotion_classification_vgg_weight.h5\' file!')
else:
    if os.path.isfile(dir_path + '/' + h5_file):
        print ('Directory ' + dir_path + '/keras_model' +
' not exist!')
        print ('Creating \'/keras_model\' directory..')
        !mkdir keras_model

        print ('Moving \''+ h5_file +'\' to
\'/keras_model\' directory..')
        !mv $h5_file keras_model

        load_h5_model(model_h5_path)
    else:
        print('Please do the training process first to get
\''+ h5_file +'\' file!')

# Note: This directory structure will need to be followed
– see notes for the next section
model_version = ,1'
export_dir = ,export/Servo/' + model_version

# Build the Protocol Buffer SavedModel at 'export_dir'
builder = builder.SavedModelBuilder(export_dir)

# Create prediction signature to be used by TensorFlow
Serving Predict API
signature = predict_signature_def(inputs={"inputs":
loaded_model.input}, outputs={"score":
loaded_model.output})
```

```
with K.get_session() as sess:
    # Save the meta graph and variables
    builder.add_meta_graph_and_variables(sess=sess,
tags=[tag_constants.SERVING],

signature_def_map={“serving_default”: signature})
    builder.save()

! ls export
! ls export/Servo
! ls export/Servo/1
! ls export/Servo/1/variables

with tarfile.open(‘model.tar.gz’, mode=’w:gz’) as archive:
    archive.add(‘export’, recursive=True)

sagemaker_session = sagemaker.Session()

inputs =
sagemaker_session.upload_data(path=’model.tar.gz’,
key_prefix=’model’)
! touch train.py

sagemaker_model = Model(model_data = ‘s3://’ +
sagemaker_session.default_bucket() +
‘/model/model.tar.gz’,
                        role = role,
                        framework_version = ‘1.15’,
                        entry_point = ‘train.py’)

timestamp_prefix = strftime(“%Y-%m-%d-%H-%M-%S”, gmtime())
endpoint_name = ‘tensorflow-inference-‘ + timestamp_prefix

predictor = sagemaker_model.deploy(initial_instance_count
= 1,

instance_type = ‘ml.m4.xlarge’,

endpoint_name = endpoint_name)

from sagemaker.tensorflow.model import TensorFlowPredictor
```

```
predictor = TensorFlowPredictor(endpoint_name,
sagemaker_session)
```

4. Tabel tipe *ML Instances* pada Amazon SageMaker

Tipe *Instance*	**vCPU**	**GPU**	**RAM (GB)**	**Memori GPU (GB)**	**Performa Jaringan**
Standar					
ml.t2.medium	2	-	4	-	Rendah
ml.t2.large	2	-	8	-	Rendah
ml.t2.xlarge	4	-	16	-	Menengah
ml.t2.2xlarge	8	-	32	-	Menengah
ml.t3.medium	2	-	4	-	Rendah
ml.t3.large	2	-	8	-	Rendah
ml.t3.xlarge	4	-	16	-	Rendah
ml.t3.2xlarge	8	-	32	-	Rendah
ml.m5.large	2	-	8	-	Tinggi
ml.m5.xlarge	4	-	16	-	Tinggi
ml.m5.2xlarge	8	-	32	-	Tinggi
ml.m5.4xlarge	16	-	64	-	Tinggi
ml.m5.12xlarge	48	-	192	-	10 Gbps
ml.m5.24xlarge	96	-	384	-	25 Gbps
ml.m4.xlarge	4	-	16	-	Tinggi
ml.m4.4xlarge	16	-	64	-	Tinggi
ml.m4.10xlarge	40	-	160	-	10 Gbps
ml.m4.16xlarge	64	-	256	-	25 Gbps
ml.m5d.large	2	-	8	-	10 Gbps
ml.m5d.xlarge	4	-	16	-	10 Gbps
ml.m5d.2xlarge	8	-	32	-	10 Gbps
ml.m5d.4xlarge	16	-	64	-	10 Gbps
ml.m5d.8xlarge	32	-	128	-	10 Gbps
ml.m5d.12xlarge	48	-	192	-	10 Gbps
ml.m5d.24xlarge	96	-	384	-	25 Gbps
Memori Teroptimasi					
ml.r5.large	2	-	16	-	10 Gbps
ml.r5.xlarge	4	-	32	-	10 Gbps
ml.r5.2xlarge	8	-	64	-	10 Gbps
ml.r5.4xlarge	16	-	128	-	10 Gbps
ml.r5.12xlarge	48	-	384	-	10 Gbps

ml.r5.24xlarge	96	-	768	-	25 Gbps
ml.r5d.large	2	-	16	-	10 Gbps
ml.r5d.xlarge	4	-	32	-	10 Gbps
ml.r5d.2xlarge	8	-	64	-	10 Gbps
ml.r5d.4xlarge	16	-	128	-	10 Gbps
ml.r5d.8xlarge	32	-	256	-	10 Gbps
ml.r5d.12xlarge	48	-	384	-	10 Gbps
ml.r5d.16xlarge	64	-	512	-	20 Gbps
ml.r5d.24xlarge	96	-	768	-	25 Gbps
Komputasi Teroptimasi					
ml.c5.large	2	-	4	-	10 Gbps
ml.c5.xlarge	4	-	8	-	10 Gbps
ml.c5.2xlarge	8	-	16	-	10 Gbps
ml.c5.4xlarge	16	-	32	-	10 Gbps
ml.c5.9xlarge	36	-	72	-	10 Gbps
ml.c5.18xlarge	72	-	144	-	25 Gbps
ml.c5d.xlarge	4	-	8	-	10 Gbps
ml.c5d.2xlarge	8	-	16	-	10 Gbps
ml.c5d.4xlarge	16	-	32	-	10 Gbps
ml.c5d.9xlarge	36	-	72	-	10 Gbps
ml.c5d.18xlarge	72	-	144	-	25 Gbps
ml.c4.large	2	-	3.75	-	Menengah
ml.c4.xlarge	4	-	7.5	-	Tinggi
ml.c4.2xlarge	8	-	15	-	Tinggi
ml.c4.4xlarge	16	-	30	-	Tinggi
ml.c4.8xlarge	36	-	60	-	10 Gbps
Percepatan Komputasi					
ml.p3.2xlarge	8	1xV100	61	16	10 Gbps
ml.p3.8xlarge	32	4xV100	244	64	10 Gbps
ml.p3.16xlarge	64	8xV100	488	128	25 Gbps
ml.p3dn.24xlarge	96	8xV100	768	256	100 Gbps
ml.p2.xlarge	4	1xK80	61	12	Tinggi
ml.p2.8xlarge	32	8xK80	488	96	10 Gbps
ml.p2.16xlarge	64	16xK80	732	192	25 Gbps
ml.g4dn.xlarge	4	1xT4	16	16	25 Gbps
ml.g4dn.2xlarge	8	1xT4	32	16	25 Gbps
ml.g4dn.4xlarge	16	1xT4	64	16	25 Gbps
ml.g4dn.8xlarge	32	1xT4	128	16	50 Gbps
ml.g4dn.12xlarge	48	4xT4	192	64	50 Gbps

ml.g4dn.16xlarge	64	1xT4	256	16	50 Gbps
ml.inf1.xlarge	4	1xT4	8	16	25 Gbps
ml.inf1.2xlarge	8	1xT4	16	16	25 Gbps
ml.inf1.6xlarge	24	1xT4	48	16	25 Gbps
ml.inf1.24xlarge	96	1xT4	192	16	100 Gbps

5. Detail Arsitektur VGG-16 yang dimodifikasi

```
Layer (type)                 Output Shape
=================================================
conv2d (Conv2D)              (None, 48, 48, 32)
activation (elu)             (None, 48, 48, 32)
batch_normalization          (None, 48, 48, 32)
-------------------------------------------------
conv2d_1 (Conv2D)            (None, 48, 48, 32)
activation_1 (elu)           (None, 48, 48, 32)
batch_normalization_1        (None, 48, 48, 32)
-------------------------------------------------
max_pooling2d                (None, 24, 24, 32)
dropout (Dropout)            (None, 24, 24, 32)
_________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 64)
activation_2 (elu)           (None, 24, 24, 64)
batch_normalization_2        (None, 24, 24, 64)
-------------------------------------------------
conv2d_3 (Conv2D)            (None, 24, 24, 64)
activation_3 (elu)           (None, 24, 24, 64)
batch_normalization_3        (None, 24, 24, 64)
-------------------------------------------------
max_pooling2d_1              (None, 12, 12, 64)
dropout_1 (Dropout)          (None, 12, 12, 64)
_________________________________________________
conv2d_4 (Conv2D)            (None, 12, 12, 128)
activation_4 (elu)           (None, 12, 12, 128)
batch_normalization_4        (None, 12, 12, 128)
-------------------------------------------------
conv2d_5 (Conv2D)            (None, 12, 12, 128)
activation_5 (elu)           (None, 12, 12, 128)
batch_normalization_5        (None, 12, 12, 128)
-------------------------------------------------
max_pooling2d_2              (None, 6, 6, 128)
dropout_2 (Dropout)          (None, 6, 6, 128)
_________________________________________________
conv2d_6 (Conv2D)            (None, 6, 6, 256)
activation_6 (elu)           (None, 6, 6, 256)
```

```
batch_normalization_6 (None, 6, 6, 256)
-----------------------------------------
conv2d_7 (Conv2D)     (None, 6, 6, 256)
activation_7 (elu)    (None, 6, 6, 256)
batch_normalization_7 (None, 6, 6, 256)
-----------------------------------------
max_pooling2d_3       (None, 3, 3, 256)
dropout_3 (Dropout)   (None, 3, 3, 256)
_________________________________________
flatten (Flatten)        (None, 2304)
_________________________________________
dense (Dense)             (None, 64)
activation_8 (elu)        (None, 64)
batch_normalization_8     (None, 64)
-----------------------------------------
dropout_4 (Dropout)       (None, 64)
_________________________________________
dense_1 (Dense)           (None, 64)
activation_9 (elu)        (None, 64)
batch_normalization_9     (None, 64)
-----------------------------------------
dropout_5 (Dropout)       (None, 64)
_________________________________________
dense_2 (Dense)            (None, 7)
-----------------------------------------
activation_10 (elu)        (None, 7)
=========================================
```

LAMPIRAN B

1. Dokumentasi pengujian pada ruang tertutup pada pengguna normal

2. Dokumentasi pengujian pada ruang terbuka pada pengguna normal

3. Dokumentasi pengujian pada pengguna tunanetra

Halaman ini sengaja dikosongkan

BIODATA PENULIS

Gregorius Rafael Widojoko, dilahirkan di Surabaya, 18 Januari 1999. Penulis adalah putra dari pasangan Anastasia Lidya Maukar dan Antonius Agus Susanto Widojoko. Penulis memulai jenjang pendidikan SD Vita School Surabaya, SMPK 4 BPK Penabur Jakarta, dan SMA Kolese Kanisius Jakarta hingga lulus tahun 2016. Pada tahun yang sama penulis melanjutkan pendidikan ke jenjang perguruan tinggi dan diterima di Institut Teknologi Sepuluh Nopember Surabaya di Departemen Teknik Elektro. Semasa kuliah, penulis aktif sebagai anggota UKM Bridge ITS, anggota Marine Solar Boat Team (MSBT) ITS, anggota KMK ITS, anggota IEEE-SB ITS, dan asisten Laboratorium Mikroelektronika dan Sistem Tertanam. Penulis dapat dihubungi melalui email : grafaelw@outlook.com