PENGENALAN WAJAH MENGGUNAKAN MOBILENETV2

SKRIPSI

Oleh:
IDZNI SHABRINA
NIM. 16650017

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2023

PENGENALAN WAJAH MENGGUNAKAN MOBILENETV2

SKRIPSI

Diajukan kepada:

Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh :

IDZNI SHABRINA
NIM. 16650017

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2023

HALAMAN PERSETUJUAN

PENGENALAN WAJAH MENGGUNAKAN MOBILENETV2

SKRIPSI

Oleh :
IDZNI SHABRINA
NIM. 16650017

Telah Diperiksa dan Disetujui untuk Diuji:
Tanggal: 14 Juni 2023

Pembimbing I,	Pembimbing II,
Dr. M. Amin Hariyadi	Dr. Cahyo Crysdian
NIP. 19670018 200501 1 001	NIP. 19740424 200901 1 008

Mengetahui,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Fachrul Kurniawan, M.MT, IPM
NIP. 19771020 200912 1 001

HALAMAN PENGESAHAN

PENGENALAN WAJAH MENGGUNAKAN MOBILENETV2

SKRIPSI

Oleh :
IDZNI SHABRINA
NIM. 16650017

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal: 27 Juni 2023

Susunan Dewan Penguji

Ketua Penguji	: Roro Inda Melani, M.T, M.Sc NIP. 19780925 200501 2 008	( )
Anggota Penguji I	: Irwan Budi Santoso, M.Kom NIP. 19770103 201101 1 004	( )
Anggota Penguji II	: Dr. M. Amin Hariyadi NIP. 19670018 200501 1 001	( )
Anggota Penguji III	: Dr. Cahyo Crysdian NIP. 19740424 200901 1 008	( )

Mengetahui dan Mengesahkan,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Fachrul Kurniawan, M.MT, IPM
NIP. 19771020 200912 1 001

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:

Nama	: Idzni Shabrina
NIM	: 16650017
Fakultas / Program Studi	: Sains dan Teknologi / Teknik Informatika
Judul Skripsi	: Pengenalan Wajah Menggunakan MobileNetV2

Menyatakan dengan sebenarnya bahwa Skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil alihan data, tulisan, atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka. Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini merupakan hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 27 Juni 2023
Yang membuat pernyataan

Idzni Shabrina
NIM. 16650017

KATA PENGANTAR

Puji syukur penulis haturkan kepada Allah SWT atas rahmat dan hidayah-Nya penelitian yang berjudul **"Pengenalan Wajah Menggunakan MobileNetV2"** dapat terselsesaikan. Sholawat dan salam semoga tetap terlimpahkan kepada Nabi Muhammad SAW yang telah membimbing umat Islam.

Keberhasilan penelitian ini tidak lepas dari bantuan berbagai pihak, penulis mengucapkan terimakasih kepada:

1. Prof. Dr. H. M. Zainuddin, MA selaku Rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang.
2. Dr. Sri Harini, M.Si selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
3. Dr. Fachrul Kurniawan ST., M.MT ., IPM, selaku ketua Program Studi Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang.
4. Dr. M. Amin Hariyadi selaku dosen pembimbing I yang telah membimbing dan mengarahkan penulisan skripsi.
5. Dr. Cahyo Crysdian selaku dosen pembimbing II yang telah membimbing dan mengarahkan penulisan skripsi.
6. Roro Inda Melani, M.T, M.Sc selaku dosen penguji I yang telah memberikan masukan dalam pengerjaan skripsi.
7. Irwan Budi Santoso, M.Kom selaku dosen penguji II yang telah memberikan masukan dalam pengerjaan skripsi.

8. Seluruh dosen Program Studi Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang yang telah mengajarkan ilmu pengetahuan selama perkuliahan.
9. Seluruh staf Program Studi Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang yang telah membantu proses studi.
10. Seluruh ustadz ustadzah Universitas Islam Negeri Maulana Malik Ibrahim Malang yang telah mengajarkan ilmu keagamaan.
11. Kedua orang tua, keluarga dan teman-teman penulis yang telah memberi dukungan hingga terselesaikan skripsi ini.
12. Semua pihak yang terlibat dalam penulisan skripsi ini.

Penulis menyadari bahwa kemampuan dan ilmu pengetahuan yang penulis miliki adalah terbatas sehingga skripsi ini belum sempurna. Penulis perlu belajar lebih banyak lagi dan diperlukan kritik dan saran untuk penelitian ini dapat terus diperbaiki dan dikembangkan. Demikian yang dapat penulis sampaikan, semoga skripsi ini dapat bermanfaat kepada para pembaca khususnya penulis sendiri.

Malang, 27 Juni 2023

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Shabrina, Idzni. 2023. **Pengenalan Wajah Menggunakan MobileNetV2**. Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (I) Dr. M. Amin Hariyadi (II) Dr. Cahyo Crysdian.

Kata kunci: pengenalan wajah, MobileNetV2.

Pengenalan wajah dapat memudahkan pekerjaan manusia karena penggunaannya praktis dan fleksibel. Saat ini telah terdapat banyak metode pengenalan wajah. Namun penelitian untuk mengembangkan metode yang sudah ada maupun menemukan metode yang baru terus dilakukan oleh para peneliti untuk menghasilkan pengenalan wajah yang lebih baik. Pada penelitian ini akan digunakan arsitektur MobileNetV2 untuk pengenalan wajah. Dilakukan beberapa skenario uji coba dan digunakan *confusion matrix* untuk mengukur peforma model dengan arsitektur MobileNetV2 jika digunakan sebagai pengenalan wajah. Hasil pengukuran peforma model dari beberapa skenario uji coba, ditemukan peforma terbaik dengan nilai *weighted average precision* sebesar 0,95 dan nilai *weighted average recall, f1-score* serta akurasi sebesar 0,94. *Transfer learning* menghasilkan nilai *precision, recall, f1-score* dan akurasi sebesar 0,94. *Fine and tuning* mampu meningkatkan peforma model, model mengalami peningkatan nilai *precision, recall, f1-score* dan akurasi menjadi 0,95.

ABSTRACT

Shabrina, Idzni. 2023. **Face Recognition Using MobileNetV2**. Thesis. Informatics Engineering Department, Faculty of Science and Technology, Universitas Islam Negeri Maulana Malik Ibrahim Malang. Advisors: (I) Dr. M. Amin Hariyadi (II) Dr. Cahyo Crysdian.

Face recognition can facilitate human work because its use is practical and flexible. Nowadays, there are many methods of face recognition. However, the researcher did this research to develop existing methods and to find new methods continuously in order to produce better face recognition. In this research, the MobileNetV2 architecture will be used for face recognition. Several trial scenarios were carried out, and a confusion matrix was used to measure the performance of the model with the MobileNetV2 architecture when used as face recognition. The result shows the best performance with a weighted average precision value of 0.95, a weighted average recall value, an f1-score, and an accuracy of 0.94. Transfer learning produces precision, recall, f1-score, and accuracy values 0.94. Fine and tuning can improve the performance of the model. The model has increased the precision, recall, f1-score, and accuracy value to 0.95.

Key words: Face recognition, MobileNetV2.

مستخلص البحث

صبرنا، إذني. ٢٠٢٣. التعرف على الوجوه باستخدام بنية MobileNetV2. البحث الجامعي. قسم الهندسة المعلوماتية، كلية العلوم والتكنولوجيا بجامعة مولانا مالك إبراهيم الإسلامية الحكومية مالانج. المشرف الأول: د. أمين هريادي. المشرف الثاني: د. جهيو كريسديان.

الكلمات الرئيسية: التعرف على الوجه، MobileNetV2.

يمكن التعرف على الوجه تسهيل العمل البشري لأن استخدامه عملي ومرن. حاليا، هناك العديد من طرق التعرف على الوجه. ومع ذلك، يستمر الباحثون في إجراء الأبحاث لتطوير الطرق الحالية وإيجاد طريقة جديدة لإنتاج التعرف على الوجه بشكل أفضل. في هذا البحث، سيتم استخدام بنية MobileNetV2 للتعرف على الوجه. تم تنفيذ العديد من سيناريوهات الاختبار وتم استخدام مصفوفة الارتباك لقياس أداء النموذج باستخدام بنية MobileNetV2 عند استخدامها كالتعرف على الوجه. وجدت نتائج قياس أداء النموذج أفضل أداء من عدة سيناريوهات الاختبار بمتوسط قيمة الضبط يبلغ ٠.٩٥ ومتوسط قيمة الاستدعاء ودرجة ف١ وقيمة الدقة ٠.٩٤. أنتج نقل التعلم قيمة الضبط والاستدعاء ودرجة ف١ وقيمة الدقة تبلغ ٠.٩٤. يمكن أن يؤدي الصقل والضبط إلى تحسين أداء النموذج، وقد زاد النموذج من قيمة الضبط والاستدعاء ودرجة ف١ والدقة إلى ٠.٩٥.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pengenalan wajah dapat digunakan dalam menjaga keamanan perangkat *smartphone* dan *tablet.* Pengenalan wajah juga dapat digunakan pada keamanan layanan bank *online*. Contoh lain penggunaan pengenalan wajah dalam kehidupan sehari-hari yaitu sistem kehadiran karyawan, pemantauan jalan raya, *boarding* kereta api, dan lain-lain. Pengenalan wajah dapat memudahkan pekerjaan manusia karena penggunaannya praktis dan fleksibel. Selain itu pengenalan wajah memiliki akurasi tinggi sehingga dapat meminimalisir terjadinya kecurangan.

Pengenalan wajah adalah salah satu teknologi *biometric* yaitu teknologi menggunakan fisik dan perilaku manusia untuk autentifikasi. Teknologi *biometric* mungkin dilakukan karena *s*etiap manusia memiliki karakteristik yang berbeda-beda sesuai dengan Alqur'an surat Al-Hujurat ayat 13.

يَٰٓأَيُّهَا ٱلنَّاسُ إِنَّا خَلَقۡنَٰكُم مِّن ذَكَرٖ وَأُنثَىٰ وَجَعَلۡنَٰكُمۡ شُعُوبٗا وَقَبَآئِلَ لِتَعَارَفُوٓاْۚ إِنَّ أَكۡرَمَكُمۡ عِندَ ٱللَّهِ أَتۡقَىٰكُمۡۚ إِنَّ ٱللَّهَ عَلِيمٌ خَبِيرٞ

*"Hai manusia, sesungguhnya Kami menciptakan kamu dari seorang laki-laki dan seorang perempuan dan menjadikan kamu berbangsa-bangsa dan bersuku-suku supaya kamu saling kenal-mengenal. Sesungguhnya orang yang paling mulia di antara kamu di sisi Allah ialah orang yang paling takwa di antara kamu. Sesungguhnya Allah Maha Mengetahui lagi Maha Mengenal."(*QS. Al-Hujurat:13)

Pengenalan wajah memiliki beberapa keunggulan dibandingkan dengan metode *biometric* lainnya yaitu tidak mencolok dan tidak mengganggu, dapat ditangkap dari jarak jauh dan secara tersembunyi. Di antara enam atribut *biometric* yang dipertimbangkan oleh Hietmeyer, fitur wajah mencetak kompatibilitas tertinggi dalam sistem *Travel Readable Travel Documents* (MRTD) berdasarkan sejumlah faktor evaluasi, seperti pendaftaran, pembaruan, persyaratan mesin, dan persepsi publik (Li dan Anil, 2011:1).

Saat ini telah terdapat banyak metode pengenalan wajah. Namun penelitian untuk mengembangkan metode yang sudah ada maupun menemukan metode yang baru terus dilakukan oleh para peneliti untuk menghasilkan pengenalan wajah yang lebih baik. Untuk saat ini, salah satu metode pengenalan wajah yang memiliki hasil akurasi yang baik adalah *Convolutional Neural Network* (CNN). Penelitian mengenai sistem kehadiran berbasis *face recognition* menggunakan *machine learning* yang dilakukan oleh Damale dan Pathak (2018) menggunakan tiga metode yang berbeda yaitu SVM, MLP, dan CNN, menunjukkan hasil akurasi SVM 87%, akurasi 86,5% pada MLP, dan CNN menghasilkan akurasi terbaik yaitu 98% .

Terdapat berbagai macam arsitektur CNN seperti AlexNet, VGGNet, ResNet, MobileNet dan EfficientNet. Penelitian mengenai arsitektur CNN terus berkembang, berbagai inovasi dilakukan sehingga dihasilkan arsitektur CNN yang lebih ringan atau memiliki peforma lebih baik daripada arsitektur-arsitektur CNN pendahulunya. Pada VGG16 misalnya, memiliki total parameter sejumlah 138 juta dan akurasi terbaik 90,1%, sedangkan MobileNetV2 dengan akurasi yang sama, hanya memiliki 3,5 juta parameter.

Berdasarkan permasalahan-permasalahan yang ada dan penelitian-penelitian sebelumnya oleh para peneliti, maka akan dibangun pengenalan wajah dengan metode *MobileNetV2*.

1.2 Pernyataan Masalah

Berdasarkan latar belakang maka pernyataan masalah yang dibuat adalah berapa peforma *MobileNetV2* untuk pengenalan wajah?

1.3 Batasan Masalah

Batasan-batasan dalam penelitian ini adalah sebagai berikut.

1. Kamera mendeteksi wajah yang tidak tertutup dan menghadap ke kamera
2. Pengambilan gambar dilakukan dengan pencahayaan yang terang
3. Jarak pengambilan gambar maksimal satu meter

1.4 Tujuan Penelitian

Tujuan penelitian ini adalah mengetahui peforma *MobileNetV2* untuk pengenalan wajah.

1.5 Manfaat Penelitian

Pengenalan wajah yang dibuat dapat mengenali identitas dengan baik dan memacu penelitian mengenai pengenalan wajah untuk dapat terus diperbaiki.

1.6 Sistematika Penulisan

Berikut adalah sistematika penelitian ini:

BAB I PENDAHULUAN

Pendahuluan berisikan Latar Belakang, Pernyataan Masalah, Tujuan, Manfaat Penelitian, Batasan Masalah Dan Sistematika Penelitian

BAB II STUDI PUSTAKA

Studi Pustaka membahas beberapa teori yang mendasari penelitian ini yaitu *Face recognition*, *Haar Cascade, Convolutional Neural Network*, *MobileNet, Confusion Matrix.*

BAB III DESAIN SISTEM DAN IMPLEMENTASI

Bab ini menjelaskan perancangan sistem dan implementasi rancangan sistem pengenalan wajah menggunakan *MobileNetV2.*

BAB IV UJI COBA DAN PEMBAHASAN

Menjelaskan langkah-langkah pengujian pengenalan wajah menggunakan *MobilenetV2.*

BAB V PENUTUP

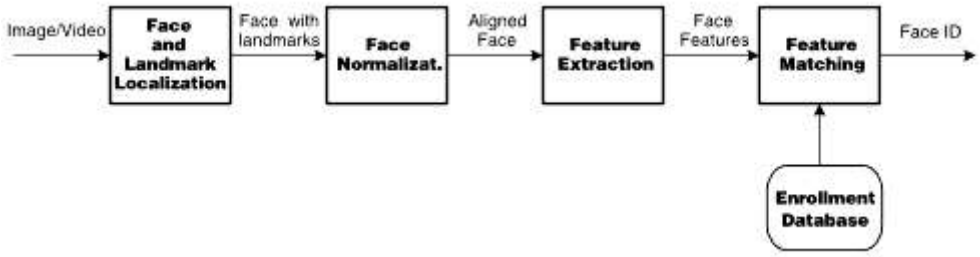
Berisi kesimpulan dari penelitian dan pembahasan mengenai peforma pengenalan wajah yang dihasilkan serta saran untuk pengembangan pengenalan wajah menggunakan MobileNetV2 di masa depan.

BAB II
STUDI PUSTAKA

2.1 *Face Recognition*

Face recognition adalah salah satu teknologi biometrik. Sistem *face recognition* digunakan untuk mengidentifikasi wajah yang terdapat dalam gambar atau video secara otomatis (Li dan Jain, 2011:2). Kinerja *face recognition* bergantung pada beberapa macam faktor seperti pencahayaan, pose wajah, ekspresi, rentang usia, rambut, pakaian yang digunakan di wajah, dan gerakan.

Alur kerja *face recognition* terdiri dari empat tahap yaitu lokalisasi wajah dan *landmark*, normalisasi wajah, ekstrasi fitur, dan pencocokan fitur (Li dan Jain, 2011:4).



Gambar 2.1 Alur kerja pengenalan wajah

Deteksi wajah memisahkan area wajah dengan gambar latar. Deteksi wajah memberikan perkiraan mengenai lokasi dan skala dari setiap wajah yang terdeteksi. Deteksi wajah memberikan estimasi kasar mengenai lokasi dan skala dari wajah, sedangkan *face landmarking* memberikan estimasi mengenai lokasi *facial landmark* seperti mata, hidung, mulut, dan guratan wajah.

Normalisasi wajah dilakukan untuk menormalisaasi gambar wajah secara geometris dan fotometrik. Normalisasi dibutuhkan untuk dapat megenali wajah dengan berbagai variasi pose dan pencahayaan. Pada normalisasi secara geometris gambar wajah dinormalisasi sesuai dengan properti geometri seperti ukuran dan pose. Pada normalisasi fotometrik gambar wajah dinormalisasi berdasarkan properti seperti pencahayaan dan *gray scale.*

Ekstrasi fitur wajah dilakukan pada gambar yang telah dinormalisasi untuk mengekstrasi informasi yang paling menonjol yang berguna untuk membedakan wajah orang, dan kuat terhadap variasi geometris dan fotometrik. Ekstrasi fitur digunakan untuk mencocokkan wajah.

Pada pencocokkan wajah, fitur yang diekstrasi dari gambar input dicocokkan dengan daftar wajah di database. Akurasi sistem pengenalan wajah bergantung pada fitur yang diekstrasi untuk merepresentasikan wajah. Dan ekstrasi fitur bergantung pada lokalisasi dan normalisasi.

Pengenalan wajah dapat dilakukan dengan berbagai macam metode yang menghasilkan tingkat akurasi yang berbeda-beda. Telah terdapat beberapa penelitian mengenai pengenalan wajah seperti penelitian yang dilakukan oleh Lukas dkk (2016), menggunakan kombinasi *Discrete Wavelet Transforms* (DWT) dan *Discrete Cosine Transform* (DCT) untuk ekstrasi fitur dari wajah kemudian diikuti dengan mengaplikasikan *Radial Basis Function* (RBF) untuk klasifikasi objek-objek wajah. Akurasi yang didapatkan mencapai 82%. Dan didapati kegagalan dalam mengenali wajah karena sistem salah mengenali wajah seorang siswa sebagai siswa lainnya

Pengenalan wajah juga dapat dilakukan dengan otomatis di mana pengguna cukup duduk di ruang kelas kemudian kamera menangkap gambar pengguna seperti pada penelitian yang dilakukan oleh Jayant dan Borra (2016) menggunakan deteksi wajah dan pengenalan wajah dengan memodifikasi algoritma Viola-Jones untuk deteksi wajah dan algoritma *alignment-free partial face recognition* untuk pengenalan wajah. Metode yang digunakan menghasilkan akurasi yang baik untuk deteksi wajah namun akurasi yang buruk untuk pengenalan wajah.

Eigenface untuk proses pengenalan wajah di sistem dan Euclidean distance digunakan untuk menghitung jarak antara gambar input dan gambar training oleh Kurniawan dkk. (2017) menghasilkan akurasi 86,67%. Namun ditemukan anomali yaitu semakin banyak gambar training memperburuk akurasi sistem, dikarenakan variance yang tinggi pada matriks Eigenface. Dan sistem yang dihasilkan tidak *robust* terhadap perubahan cahaya.

Penelitian pengenalan wajah dengan deteksi wajah menggunakan algoritma Viola dan Jones dan ekstrasi fitur menggunakan CNN yang dilakukan Bhattacharya dkk. (2018) didapati hasil yang memuaskan terhadap ekspresi wajah yang berbeda, pencahayaan, dan pose. Namun masih didapati sistem terkadang gagal mengenali wajah.

Berbagai metode dapat digunakan untuk membagun pengenalan wajah seperti penelitian yang dilakukan oleh Damale dan Pathak (2018) menggunakan tiga metode yang berbeda yaitu SVM, MLP, dan CNN. DNN digunakan untuk deteksi wajah. Pada SVM dan MLP digunakan PCA dan LDA untuk ekstrasi fitur. Pada CNN, gambar langsung dimasukkan ke CNN sebagai vektor fitur. Hasil

akurasi SVM 87%, akurasi 86,5% pada MLP, dan CNN menghasilkan akurasi terbaik yaitu 98%.

Penelitian yang dilakukan oleh Sanli dan Ilgen (2018) menggunakan *Haar filtered AdaBoost* digunakan untuk deteksi wajah manusia secara real time, *Principal Component Analysis* (PCA) dan Local Binary Pattern (LBPH) untuk mengidentifikasi wajah yang terdeteksi. Akurasi yang didapatkan antara 75% hingga 95%.

2.2 *Haar Cascade*

Haar-cascade adalah metode yang ditemukan oleh Viola dan Jones di mana melatih machine learning untuk deteksi objek di dalam gambar. Nama metode ini terdiri dari dua kata yaitu Haar dan *Cascade*. Fitur Haar adalah sebuah kotak yang dipisah menjadi dua, tiga, atau empat kotak. Di mana setiap kotak bewarna hitam atau putih. *Haar-cascade* perlu dilatih dengan berbagai gambar positif dan negatif. Tujuan dari metode ini adalah untuk mengekstrasi kombinasi dari fitur-fitur yang merepresentasikan wajah. Gambar positif mengandung objek yang harus dikenali yaitu wajah, sedangkan gambar negatif merepresentasikan sebuah gambar tanpa wajah. Machine learning ini membutuhkan gambar grayscale. Intensitas warna abu-abu akan digunakan untuk mendeteksi fitur yang direpresentasikan. Fitur-fitur tersebut dapat ditemukan dengan menghitung jumlah piksel yang bewarna gelap di sebuah wilayah dikurangi dengan jumlah piksel yang bewarna terang. Kombinasi fitur yang diekstrasi dari pelatihan akan digunakan untuk deteksi wajah di gambar. Untuk mendeteksi wajah di gambar yang tidak dikenali, akan dicari kombinasi fitur-fiturnya. Fitur dicoba untuk dicocokkan hanya dalam blok piksel yang

ditentukan oleh skala. Setiap fitur dari kombinasi akan dicoba untuk dicocokkan satu per satu dalam blok. Jika salah satu dari fitur-fitur tidak ditemukan di blok tersebut maka pencarian akan dihentikan. Fitur-fitur lainnya tidak akan dicoba karena mesin telah menyimpulkan bahwa tidak terdapat wajah di boks tersebut. Kemudian pencarian akan dilanjutkan ke blok berikutnya, demikian seterusnya. Metode ini menguji semua blok piksel dengan cara mengalir ke bawah atau *cascade*. Setelah menguji blok terakhir, skala blok akan ditingkatkan dan proses deteksi wajah akan dimulai kembali. Proses ini diulang beberapa kali dengan skala yang berbeda untuk mendeteksi wajah dengan ukuran yang berbeda.

2.3 *Convolutional Neural Network*

Convolutional Neural Network (CNN) adalah salah satu kelas *deep feed-forward artificial neural network*. CNN terdiri atas satu lapisan masukan, satu lapisan keluaran, dan beberapa lapisan tersembunyi (Suyanto, 2018:308). Lapisan tersembunyi terdiri atas *convolutional layers, pooling layers, normalization layers*, ReLU *layer*, *fully connected layers, dan loss layer*. CNN menggunakan arsitektur tiga dimensi: lebar (width), tinggi (height), dan dalam (depth). Lebar dan tinggi menyatakan dimensigambar sedangkan *depth* menyatakan RGB (*Red, Green, Blue*). CNN memiliki empat *hyperparameter* (parameter yang harus ditentukan oleh pengguna) yaitu: jumlah filter K, ukuran bidang reseptif atau *spatial extent* (tingkat spasial), lebar langkah atau *stride* S, dan *zero padding* P (Suyanto, 2018:319).

Convolution layer membandingkan filter ke sebuah kotak dari gambar dengan ukuran yang sama. Proses ini dilakukan berulang kali, membandigkan blok berikutnya sampai ke seluruh bagian gambar (Delbiaggio, 2017:18). Untuk menentukan pergerakan pindah dari satu blok ke blok lainnya, perlu memperhatikan langkah atau *stride*. *Stride* mempresentasikan pergeseran jumlah piksel dari posisi terakhir. Gambar dapat ditambahkan padding zero yaitu menambahkan beberapa piksel tambahan mengelilingi gambar dengan nilai 0 untuk mengatur ukuran dari output. Setiap perbandingan filter dengan kotak dari gambar akan menghasilkan satu angka. Dengan demikian setelah membandingkan akan didapatkan sebuah kotak baru dengan ukuran yang lebih kecil. Ukuran kotak yang dihasilkan bergantung pada beberapa faktor: ukuran gambar, ukuran filter, nilai *stride*, dan ukuran *padding*. Berikut adalah formula yang dapat digunakan untuk memprediksi ukuran dari kotak yang dihasilkan:

$$O = \frac{(w - k + 2p)}{s} + 1 \qquad (2.1)$$

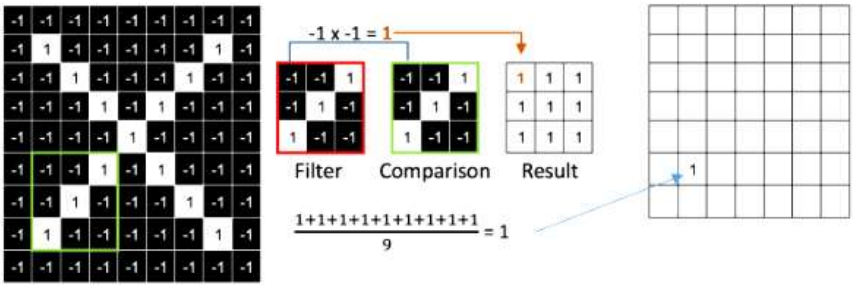
O = tinggi atau panjang *output*

w = tinggi atau panjang *input*

k = ukuran *filter*

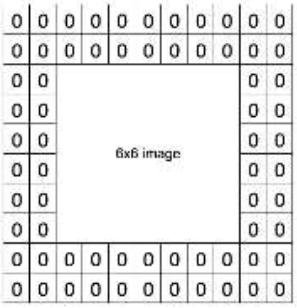
p = ukuran *padding*

s = nilai *stride*



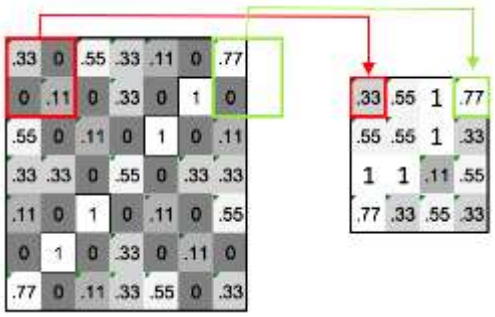
Gambar 2.2 Visualisasi membandingkan sebuah filter dengan sebuah blok dari gambar

Gambar 2.3 Contoh perpindahan *stride*=1 dan *stride*=3



Gambar 2.4 Contoh penambahan *padding* 2 pada gambar

Pada convolution layer terjadi proses konvolusi yang pada dasarnya hanya berupa dot product antara filter dengan sebuah bidang reseptif kecil pada (citra) masukan yang berukuran sama dengan ukuran filter (Suyanto, 2018:309). Pooling layer berfungsi menjaga ukuran data ketika konvolusi yaitu dengan melakukan prededuksian sampel atau downsampling sehingga didapatkan representasi data yang lebih kecil, mudah dikelola, dan mudah mengontrol overfitting. Proses pooling yang umum digunakan adalah max pooling yaitu memilih nilai maksimum dalam suatu area tertentu.



Gambar 2.5 Visualisasi proses *max pooling*

Normalization layer pada awalnya didesain untuk mengatasi adanya perbedaan rentang nilai yang signifikan pada citra masukan. Namun lapisan ini tidak banyak digunakan secara praktis karena dampaknya yang relatif kecil. *Rectified Linear Units* (Relu) *Layer* mengaplikasikan fungsi aktivasi f(x)=max(0,x) untuk meningkatkan sifat nonlinearitas fungsi keputusan dan jaringan secara keseluruhan tanpa memengaruhi bidang-bidang reseptif pada *convolution layer*. *Fully connected layer* setiap neuron memiliki koneksi penuh ke semua aktivasi dalam lapisan sebelumnya . Dan *Loss layer* menetukan bagaimana pelatihan memberikan penalti atas penyimpangan antara hasil prediksi dan label.

Berdasarkan penelitian yang dilakukan oleh Dhillon dan Verma (2019) telah terdapat beberapa model CNN yaitu LeNet, AlexNet, ZFNet, GoogleNet, VGGNet, ResNet, Inception model, ResNeXt, SeNet, MobileNet V1/V2, DenseNet, Xception model, NASNet/PNASNet/ENASNet, dan EfficientNet. LeNet adalah CNN pertama yang sukses diaplikasikan untuk membaca kode pos dan digit. Arsitektur ini dibangun pada tahun 1990-an oleh Yann LeCun. AlexNet dibangun oleh Alex Krizhevsky beserta dua rekannya pada tahun 2012. Arsitektur AlexNet mirip dengan LeNet namun lebih dalam, lebih besar, dan memiliki fitur baru yaitu sejumlah lapisan konvolusi yang ditumpuk di bagian atas. Sedangkan LeNet

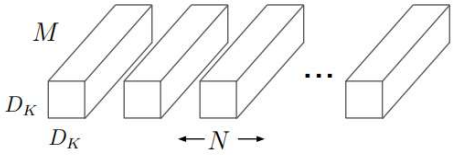
umumnya hanya menggunakan hanya satu lapisan konvolusi diikuti *pooling layer.* ZFNet dibangun oleh Matthew Zeiler dan Rob fergus. ZFNet memperbaiki kekurangan AlexNet dengan mengutak-atik *hyperparameters,* memperluas ukuran lapisan konvolusi bagian tengah, memperkecil *stride* dan ukuran filter pada lapisan pertama. Pada arsitektur GoogleNet dibangun *Inception Module* yang mereduksi jumlah parameter. GoogleNet banyak menggunakan *average pooling* di bagian atas untuk menghilangkan parameter-parameter yang tidak penting, tidak seperti arsitektur sebelumnya yang menggunakan *fully connected layers* di bagian atas. VGGNet menunjukkan semakin dalam jaringan CNN semakin tinggi akurasinya. Arsitektur ini memiliki parameter yang banyak sehingga membutuhkan banyak memori. Sebagaian besar parameter-parameter tersebut terdapat pada *fully connected layer* di bagian atas. ResNet atau *residual network* memiliki tiga ciri khusus yaitu *skip connections,* menggunakan *batch normalization,* dan menghilangkan *fully connected layers* di bagian akhir. Tujuan utama *Inception Model* adalah menemukan desain *sparse* lokal yang ideal dalam kerangka visi konvolusional dapat diamankan dengan komponen padat yang segera tersedia. Model ini menggunakan banyak filter untuk mengurangi dimeni dan aktivasi *rectified linear,* dan jumlah parameter yang digunakan lebih sedikit . ResNext merupakan perpaduan ResNet dan VGGs. Arsitektur ini menggunakan model *multi-branch* yang terdiri dari beberapa *hyperparameter,* dan terdapat dimensi baru bernama *cardinality.* SENet berfokus pada hubungan antar *channel* dan menggunakan sistem arsitektur *block squeeze and excitation* (SE). MobileNetV2 memanfaatkan konvolusi yang dapat dipisahkan secara mendalam sebagai blok,

memperkenalkan dua desain baru: (1) *bottleneck* linier di antara lapisan dan (2) jalan pintas di antara *bottlenecks*. Setiap lapisan pada arsitektur DenseNet digabungkan dengan cara *feed forward.* Model dengan lapisan L ini memiliki koneksi langsung L (L + 1) / 2, menggabungkan peta fitur output dengan peta fitur yang masuk; oleh karena itu, setiap lapisan memperoleh pengetahuan kolektif dari semua lapisan sebelumnya. Karya ini berguna dalam hal mengurangi masalah gradien hilang, meminimalkan jumlah parameter dan konsep penggunaan kembali fitur. Kelemahan model ini adalah menggunakan banyak memori. Xception adalah perkembangan dari Inception Model dengan jumlah koneksi lebih sedikit. NAS/PNAS/ENAS adalah algoritma untuk mencari model terbaik pada *neural network.*

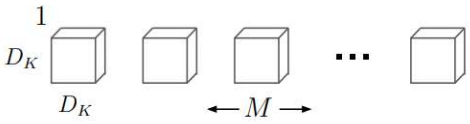
2.4 MobileNet

MobileNet terdapat beberapa versi. Arsitektur MobileNet versi pertama yang ditulis oleh Howard dkk (2017) terdiri dari *depthwise separable convolution* yaitu konvolusi yang memfaktorisasi konvolusi standar menjadi *depthwise convolution* dan konvolusi 1 × 1 yang disebut *pointwise convolution*. *Depthwise convolution* mengaplikasikan sebuah filter kepada setiap input chanel. *Pointwise convolution* kemudian mengaplikasikan konvolusi 1× 1 untuk mengkombinasikan output dari *depthwise convolution*. Sedangkan pada konvolusi standar, mengaplikasikan filter dan mengkombinasikan input menjadi output dilakukan dalam satu langkah. Dengan memfaktorisasi konvolusi, didapatkan komputasi dan ukuran model yang lebih kecil. Struktur MobileNet terdiri dari *depthwise separable convolution* dan layer awal berupa konvolusi standar. Semua layer diikuti dengan

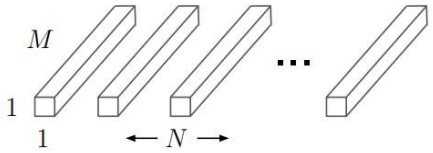
batchnorm dan ReLU kecuali layer terakhir berupa *full connected layer* yang diikuti dengan *layer softmax* untuk klasifikasi. Sembilan puluh lima persen komputasi MobileNet terjadi pada konvolusi 1 x 1 dan tujuh puluh lima persen parameter MobileNet juga terletak pada konvolusi 1 x 1.



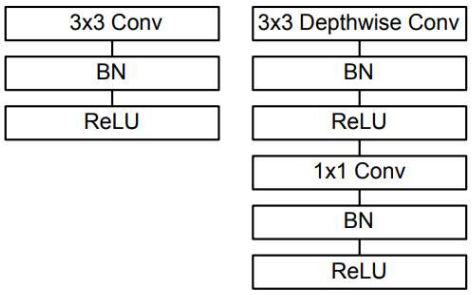
Gambar 2.6 Konvolusi standar



Gambar 2.7 Depthwise convolution



Gambar 2.8 Pointwise convolution



Gambar 2.9 Perbedaan konvolusi standar(kiri) dengan depthwise separable convolution(kanan)

MobileNetV2 memiliki arsitektur dan struktur yang mirip dengan MobileNet, hanya saja ditambahkan *linear bottleneck* dan *inverted residuals* pada blok konvolusinya. Pada *linear bottleneck*, jika *manifold of interest* tetap bervolume

tidak nol setelah transformasi ReLU, maka dilakukan transformasi linear. Transformasi linear dilakukan karena transformasi non-linear dapat menghilangkan beberapa informasi penting dari *manifold of interest.* ReLU dapat menyimpan informasi *manifold of interest* secara lengkap jika hanya terletak di *subspace input* berdimensi rendah (Sandler dkk. 2019). Pada *inverted residuals* dilakukan ekspansi fitur, ditambahkan fitur dari konvolusi di awal. Arsitektur MobileNetV2 terdiri dari layer konvolusi standar dengan 32 filter kemudaian diikuti dengan 19 *layer residual bottleneck*. *Layer residual bottleneck* dideskripsikan pada tabel 2.1. Arsitektur MobileNetV2 dideskripsikan pada tabel 2.2.

Tabel 2.1 Bottleneck residual block merubah k chanel menjadi k' chanel dengan stride s dan expansion factor t.

Input	*Operator*	*Output*
$h \times w \times k$	1×1 conv2d, ReLU6	$h \times w \times (tk)$
$h \times w \times tk$	3×3 dwise s=s, ReLu6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times (tk)$	linear 1×1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times k'$

Tabel 2.2 Struktur model MobileNetV2 oleh Sandler, dkk., 2019

No	Operator	Expansion rate	Channel	Stride
1.	Conv2d	-	32	2
2.	Bottleneck	1	16	1
3.	Bottleneck	6	24	2
4.	Bottleneck	6	24	1
5.	Bottleneck	6	32	2
6.	Bottleneck	6	32	1
7.	Bottleneck	6	32	1
8.	Bottleneck	6	64	1
9.	Bottleneck	6	64	2
10.	Bottleneck	6	64	1
11.	Bottleneck	6	64	1

Lanjutan

No	Operator	Expansion rate	Channel	Stride
12.	Bottleneck	6	96	1
13.	Bottleneck	6	96	1
14.	Bottleneck	6	96	1
15.	Bottleneck	6	160	1
16.	Bottleneck	6	160	2
17.	Bottleneck	6	160	1
18.	Bottleneck	6	320	2
19.	Conv2d	-	1280	1
20.	Avgpool 7 ×7			
21.	Fully connected layer			

2.5 *Confusion Matrix*

Untuk mengetahui peforma sistem yang dibangun digunakan *Confusion Matrix. Confusion Matrix* meringkas kinerja klasifikasi dari suatu *classifier* sehubungan dengan data uji (Sammut dan Webb, 2017:260) Berupa matriks dua dimensi dengan salah satu dimensinya adalah kelas sebenarnya dan sebuah objek dan dimensi lainnya adalah kelas yang ditetapkan oleh *classifier.* Pada umunya matriks yang digunakan terdiri dari dua kelas yaitu satu kelas positif dan satu kelas negatif sehingga berbentuk seperti tabel berikut.

Tabel 2.2 Hasil klasifikasi berupa kelas positif dan negatif

		Assigned class	
Actual class		*Positive*	*Negative*
	Positive	TP	FN
	Negative	FP	TN

Isi dari keempat sel tabel 1 yaitu *true positive* (TP)*, false positive* (FP)*, true negative* (TN) dan *false negative* (FN). TP adalah data positif yang diklasifikasikan

dengan benar oleh sistem. FP adalah data negatif yang diklasifikasikan dengan salah oleh sistem sebagai positif. TN adalah data negatif yang diklasifikasikan dengan benar oleh sistem. FN adalah jumlah data positif yang diklasifikasikan salah oleh sistem.

Pengukuran peforma klasifikasi dapat didefinisikan dengan empat istilah berikut:

$$\textit{Specificity} = \textit{true negative rate} = \frac{\mathrm{TN}}{\mathrm{TN+FP}} \tag{2.2}$$

$$\textit{Sensitivity} = \textit{true positive rate} = \mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP+FN}} \tag{2.3}$$

$$\textit{Positive predictive value} = \textit{Precision} = \frac{\mathrm{TP}}{\mathrm{TP+FP}} \tag{2.4}$$

$$\textit{Negative predictive value} = \frac{\mathrm{TN}}{\mathrm{TN+FN}} \tag{2.5}$$

Presisi adalah perbandingan TP dengan total jumlah data positif yang diprediksi. *Recall* adalah perbandingan TP dengan jumlah total data positif yang sebenarnya. Sensitivitas dan spesifisitas adalah dua ukuran yang digunakan bersama-sama dalam beberapa domain untuk mengukur kinerja prediktif model klasifikasi atau uji diagnosa (Sammut dan Webb, 2017:1152). Akurasi adalah ukuran derajat prediksi sebuah sistem sesuai dengan kenyataan, adalah perbandingan jumlah objek yang terklasifikasikan dengan benar dengan total jumlah objek. Akurasi berkaitan dengan *error rate* di mana hasil akurasi adalah 1.0-*error rate. Error rate* adalah ukuran tingkat prediksi yang salah dari sebuah sistem. Berikut adalah persamaan untuk mengukur akurasi:

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

BAB III
DESAIN DAN IMPLEMENTASI

3.1 Desain Sistem

Akan dibangun model yang dapat mengenali identitas dari input berupa citra wajah. Desain sistem dapat dilihat pada gambar 3.1.

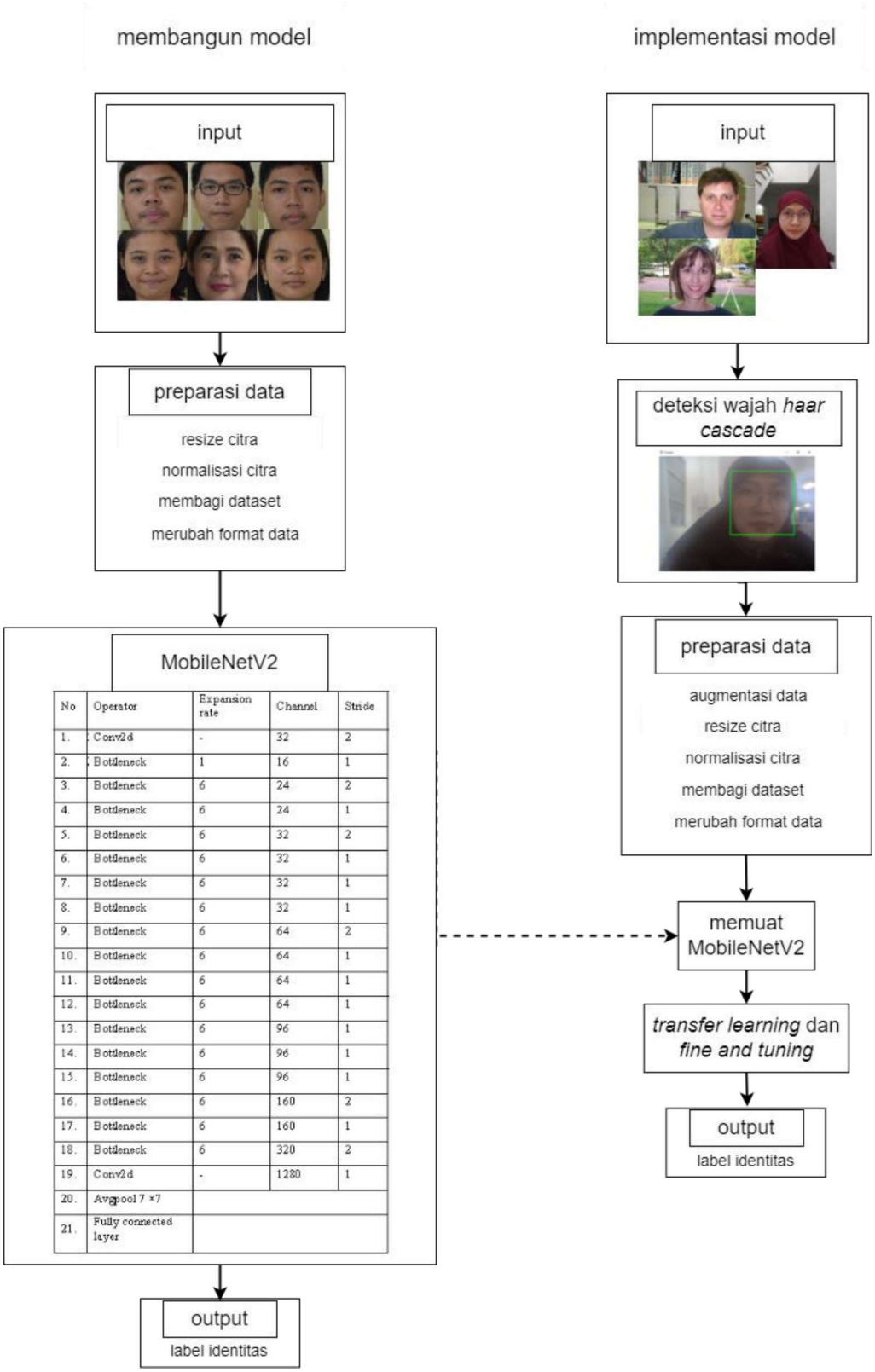
Secara garis besar, desain sistem terbagi menjadi 2 tahap yaitu:

1. Membangun model.

   Proses membangun model dideskripsikan pada gambar 3.1 pada bagan sebelah kiri. Dibangun model yang dapat menjalankan fungsi khusus yaitu dapat mengenali wajah. Model dibangun dengan data yang besar agar model dapat berfungsi dengan baik.

2. Implementasi model

   Proses implementasi model dideskripsikan pada gambar 3.1 pada bagan sebelah kanan. Model yang telah dibangun dengan dataset yang besar kemudian dimuat kembali dan dilakukan penyesuaian agar dapat digunakan pada pengguna pengenalan wajah.



Gambar 3.1 Desain sistem

3.1.1 Membangun Model

Proses membangun model terbagi menjadi beberapa bagian sebagai berikut.

1. Input

   Input berupa citra wajah manusia dengan beberapa identitas berbeda. Citra wajah yang digunakan adalah citra wajah dengan pencahayaan cukup, wajah tidak terhalangi objek lain, wajah menghadap ke depan dan citra berukuran 224 × 224 × 3.

2. Preparasi data

   Preparasi data dilakukan dengan mengatur ulang ukuran citra menjadi 224 × 224 × 3, menormalisasi citra dengan membaginya dengan nilai 255,0 kemudian membagi data menjadi beberapa dataset yaitu dataset pelatihan, evaluasi dan uji. Tahap terakhir dari pengolahan data adalah mengubah format dataset menjadi tensor dataset.

3. MobileNetV2

   Menyusun *layer-layer* model satu-persatu sesuai dengan rancangan struktur model. Rancangan struktur model yang akan dibangun mengikuti arsitektur MobileNetV2 dengan beberapa perubahan. Rancangan struktur model dideskripsikan pada tabel 3.1. Perbedaan rancangan struktur model dengan struktur model MobileNetV2 yang dibangun oleh Sandler, dkk. pada tahun 2019 adalah terdapat 4 *layer* dengan *channel* 32 pada rancangan struktur model sedangkan pada struktur model MobileNetV2 hanya terdapat 3 *layer* dengan *channel* 32. Perbedaan juga terjadi pada *layer* dengan *channel* 160, hanya terdapat 2 *layer channel* 160 pada rancangan struktur model sedangkan pada

struktur MobileNetV2 terdapat 3 *layer* dengan *channel* 160. Perbedaan jumlah *layer* pada *channel-channel* tersebut memengaruhi jumlah parameter yang dimiliki model. Rancangan struktur model terdiri dari dua puluh satu *layer* yaitu konvolusi standar sebagai *layer* pertama, tujuh belas belas *bottleneck layer*, sebuah konvolusi standar, *layer average pool* dan terakhir *fully connected layer*.

Tabel 3.1 Rancangan struktur

No	Operator	Expansion rate	Channel	Stride
1.	Conv2d	-	32	2
2.	Bottleneck	1	16	1
3.	Bottleneck	6	24	2
4.	Bottleneck	6	24	1
5.	Bottleneck	6	32	2
6.	Bottleneck	6	32	1
7.	Bottleneck	6	32	1
8.	Bottleneck	6	32	1
9.	Bottleneck	6	64	2
10.	Bottleneck	6	64	1
11.	Bottleneck	6	64	1
12.	Bottleneck	6	64	1
13.	Bottleneck	6	96	1
14.	Bottleneck	6	96	1
15.	Bottleneck	6	96	1
16.	Bottleneck	6	160	2
17.	Bottleneck	6	160	1
18.	Bottleneck	6	320	2

Tabel 3.2 Struktur model MobileNetV2 oleh Sandler, dkk., 2019

No	Operator	Expansion rate	Channel	Stride
1.	Conv2d	-	32	2
2.	Bottleneck	1	16	1
3.	Bottleneck	6	24	2
4.	Bottleneck	6	24	1
5.	Bottleneck	6	32	2
6.	Bottleneck	6	32	1
7.	Bottleneck	6	32	1
8.	Bottleneck	6	64	1
9.	Bottleneck	6	64	2
10.	Bottleneck	6	64	1
11.	Bottleneck	6	64	1
12.	Bottleneck	6	96	1
13.	Bottleneck	6	96	1
14.	Bottleneck	6	96	1
15.	Bottleneck	6	160	1
16.	Bottleneck	6	160	2
17.	Bottleneck	6	160	1
18.	Bottleneck	6	320	2

Lanjutan tabel

No	Operator	Expansion rate	Channel	Stride
19.	Conv2d	-	1280	1
20.	Avgpool 7 ×7			
21.	Fully connected layer			

No	Operator	Expansion rate	Channel	Stride
19.	Conv2d	-	1280	1
20.	Avgpool 7 ×7			
21.	Fully connected layer			

3.1.2 Implementasi Model

Setelah model dibangun, tahap selanjutnya adalah menggunakan model tersebut terhadap pengguna pengenalan wajah. Tahapan menggunakan model yang telah dibangun adalah sebagai berikut

1. Input

   Input berupa citra yang mengandung wajah manusia. Citra wajah yang digunakan adalah citra wajah dengan pencahayaan cukup, wajah tidak terhalangi objek lain dan wajah menghadap ke depan.

2. Deteksi Wajah menggunakan *Haar Cascade*

   Model yang dibangun memiliki input yang spesifik yaitu citra wajah manusia berukuran 224 × 224 × 3. Karena model digunakan pada pengenalan wajah, maka citra yang terkumpul nantinya adalah citra hasil tangkapan kamera yang belum diolah. Citra yang didapatkan adalah citra wajah yang belum dipisahkan dengan latar belakangnya. Agar model dapat berfungsi dengan baik maka perlu disamakan inputnya, perlu dilakukan deteksi wajah pada citra yang didapatkan menggunakan *haar cascade.* Penulis tidak membangun *haar cascade*

secara mandiri melainkan menggunakan *haar cascade* hasil penelitian Viola dan Jones yang dapat mendeteksi wajah *frontal*.

3. Preparasi data

Dibutuhkan waktu dan *resource* untuk terkumpul data berupa citra masing-masing identitas pengguna pengenalan wajah. Keterbatasan waktu dan *resource* dapat menyebabkan data atau citra yang terkumpul terbatas dan memiliki kualitas yang kurang baik, padahal kualitas data berpengaruh pada baik atau buruknya sebuah model. Untuk meningkatkan kualitas data maka perlu dilakukan augmentasi data pada citra yang telah terkumpul. Data yang telah diaugmentasi kemudian diatur ulang ukurannya menjadi 224 × 224 × 3, dinormalisasi dengan membagi nilai citra dengan nilai 255,0 dan kemudian membagi dataset menjadi dataset pelatihan, evaluasi dan uji. Tahap terakhir dari pengolahan data adalah mengubah format data menjadi tensor dataset sehingga data siap untuk dimasukkan ke model.

4. *Transfer learning* dan *fine and tuning*

Model yang telah dibangun yaitu model yang telah dilatih dengan data yang lebih besar kemudian digunakan kembali dengan melakukan beberapa penyesuaian. Penyesuaian model dapat dilakukan pada *layer* terakhir yaitu pada *fully connected layer* untuk menyesuaikan dengan jumlah identitas, atau pada beberapa layer namun dengan perubahan yang sangat kecil jika identitas pengguna pengenalan wajah memiliki perbedaan sangat mencolok dari identitas pada data yang digunakan untuk membangun model. Perubahan pada beberapa

layer namun dengan perubahan yang sangat kecil hanya dilakukan jika memiliki data yang cukup.

3.2 Implementasi Desain

Sistem dibangun menggunakan bahasa pemrograman *python* dan *library keras* dan *tensorflow*. Digunakan tools IDLE Python, Jupyter Notebook dan Google Colab.

3.2.1 Input

Input berupa citra wajah manusia dengan pencahayaan cukup, wajah menghadap ke depan dan wajah tidak terhalangi objek. Pengumpulan data dilakukan dengan mengambil data dari penelitian yang sudah ada dan pengambilan citra secara mandiri.

3.2.1.1 Membangun Model: Input

Data input yang digunakan untuk membangun model adalah data dari KomNet hasil penelitian Astawaa, dkk (2020). KomNet adalah dataset wajah untuk pengenalan wajah yang diambil dari tiga sumber media yaitu kamera *handphone*, kamera *digital* dan sosial media. Namun dataset yang digunakan pada penelitian ini adalah adalah dataset citra yang diambil menggunakan kamera *digital* dan kamera *handphone* saja tanpa citra yang diambil dari sosial media. Beberapa contoh citra dari penelitian yang dilakukan oleh Astawaa, dkk (2020) dapat dilihat pada gambar 3.2.

Gambar 3.2 Contoh data dari KomNet oleh Astawaa, dkk (2020)

3.2.1.2 Implementasi Model: Input

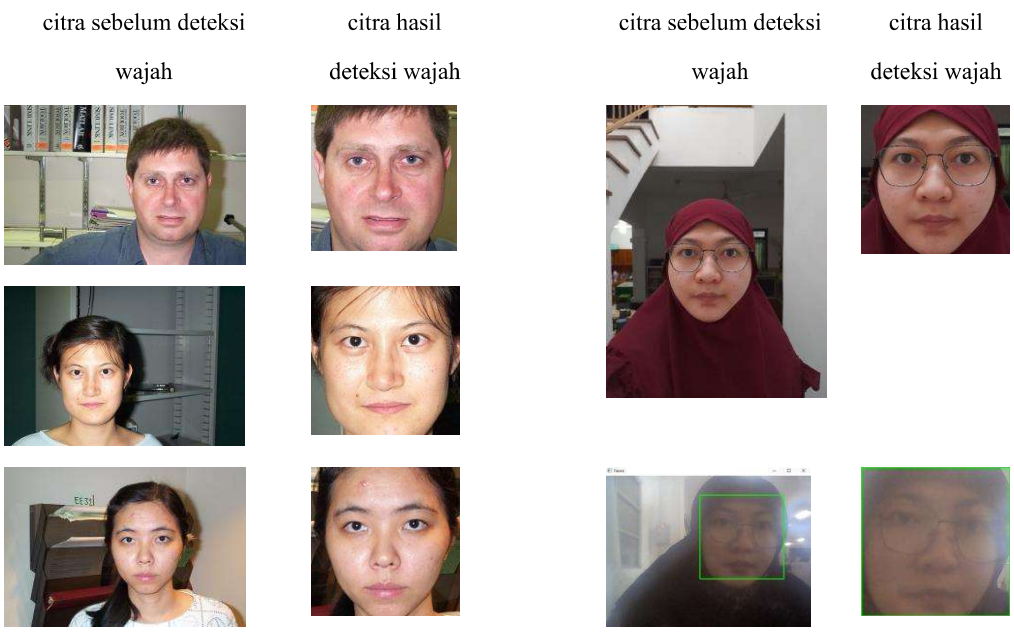
Dilakukan pengumpulan data berupa citra wajah peserta pengenalan wajah . Data didapatkan dari Caltech (didownload dari www.vision.caltech.edu/datasets) dan pengambilan citra secara mandiri sehingga terkumpul 200 citra dengan 20 identitas berbeda.

Gambar 3.3 Contoh data dari Caltech dan pengambilan citra secara mandiri

3.2.2 Implementasi Model: Deteksi Wajah *Haar Cascade*

Model yang dibangun memiliki input yang spesifik yaitu citra wajah manusia berukuran 224 × 224 × 3. Citra yang terkumpul adalah citra hasil tangkapan kamera yang belum diolah yaitu citra wajah yang belum dipisahkan dengan latar belakangnya. Agar model dapat berfungsi dengan baik maka perlu

disamakan inputnya, perlu dilakukan deteksi wajah pada citra yang didapatkan menggunakan *haar cascade.* Penulis tidak membangun *haar cascade* secara mandiri melainkan menggunakan *haar cascade* hasil penelitian Viola dan Jones yang dapat mendeteksi wajah *frontal*.



Gambar 3.4 Contoh deteksi wajah menggunakan *haar cascade*

```
import cv2

#cascade classifier
face_cascade=cv2.CascadeClassifier('E:\haarcascade_frontalface_default.xml')

#detect faces
def detect(img):
    faces=face_cascade.detectMultiScale(img, 1.1, 1)
    #crop image
    for obj in faces:
        (x, y, w, h) = obj
    crop_img=img[y:y+h, x:x+w]
    return crop_img
```

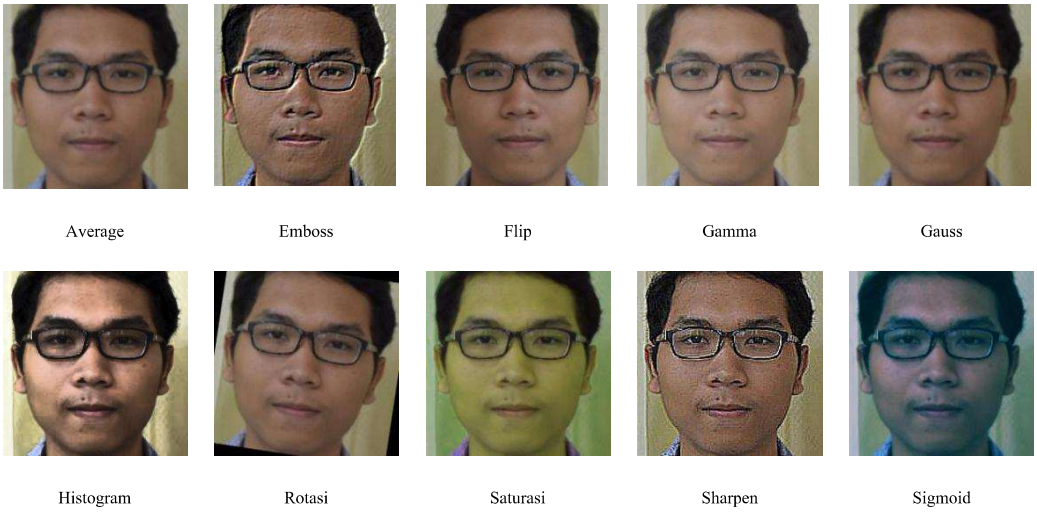
Gambar 3.5 Kode program deteksi wajah menggunakan *haar cascade*

3.2.3 Preparasi Data

Data yang telah terkumpul kemudian diolah, menyesuaikan dengan format input model. Dilakukan pengaturan ukuran citra, normalisasi citra dan pengubahan format data.

3.2.3.1 Membangun Model: Preparasi Data

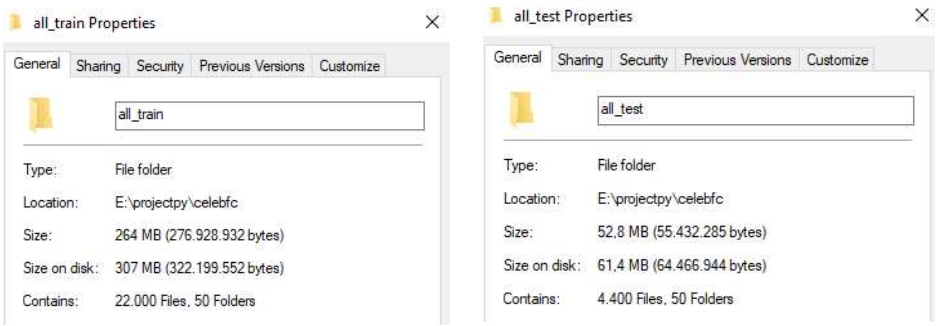
Data dari KomNet telah dipotong, dipisahkan dengan latar belakangnya dan memiliki ukuran 224 × 224 ×3. Telah dilakukan augmentasi data menggunakan *average blur*, *emboss*, *flip*, *gamma contrast*, *gaussian blur*, *histogram equalization*, rotasi, *hue* and saturasi, *sharpen*, *sigmoid contrast*. Contoh citra yang telah diaugmentasi dapat dilihat pada gambar 3.6.



Gambar 3.6 Augmentasi data pada KomNet oleh Astawaa, dkk (2020)

Data hasil augmentasi telah terbagi menjadi data pelatihan dan data uji. Terdapat 22.000 citra pada *folder training* dengan 50 identitas sehingga masing-masing identitas memiliki data sebanyak 440 citra wajah. Terdapat 4.400 citra pada

folder test dengan 50 identitas sehingga masing-masing identitas memiliki data sebanyak 88 citra wajah.



Gambar 3.7 Pembagian data pelatihan dan uji

Data berupa citra tersebut kemudian dirubah formatnya menjadi tensor dataset. Merubah format data menjadi tensor dataset diperlukan untuk menghemat memori karena pada tensor dataset, semua data tidak dimuat langsung ke memori melainkan terbagi dalam beberapa *batch.* Selain menghemat memori, mengubah format data menjadi tensor dataset diperlukan untuk menyesuaikan dengan format input model yaitu berupa tensor. Data berupa citra dimuat dari *file* dengan ukuran 224 × 224 dan *batch* 32.

```
import os
import tensorflow as tf
batch_size = 32
img_height = 224
img_width = 224
data_dir1 = (r"E:\all_train")
data_dir2 = (r"E:\all_test")
train_ds = tf.keras.utils.image_dataset_from_directory(data_dir1,
                                                       image_size=(img_height, img_width),
                                                       batch_size=batch_size)
```

```
Found 22000 files belonging to 50 classes.
Wall time: 1min 8s
```

Gambar 3.8 Kode program dan hasil merubah data pelatihan berupa citra pada *file disk* menjadi tensor dataset

```
test_ds = tf.keras.utils.image_dataset_from_directory(
data_dir2,
image_size=(img_height, img_width),
batch_size=batch_size
)
Found 4400 files belonging to 50 classes.
Wall time: 1.88 s
```

Gambar 3.9 Kode program dan hasil merubah data uji berupa citra pada *file disk* menjadi tensor dataset

Setelah merubah format data menjadi tensor dataset, langkah selanjutnya adalah normalisasi data yaitu membagi nilai citra dengan nilai *float* 255,0. Normalisasi baru dilakukan setelah data berbentuk tensor dataset karena tensor dataset sudah terbagi menjadi *batch* sehingga memori komputer dapat menangani operasi normalisasi.

```
normalization_layer = tf.keras.layers.Rescaling(1./255)

normalized_ds1 = train_ds.map(lambda x, y: (normalization_layer(x), y))
normalized_ds2 = test_ds.map(lambda x, y: (normalization_layer(x), y))

svdir="E:\dataribuan"
path_train = os.path.join(svdir, "tensorflow_traindataset")
path_test= os.path.join(svdir, "tensorflow_testdataset")

normalized_ds1.save(path_train)
normalized_ds2.save(path_test)
```

Gambar 3.10 Kode program menormalisasi data.

Diimpor *normalization_layer* dari keras untuk membagi nilai citra dengan 255. Digunakannya *library* untuk normalisasi data karena menyesuaikan dengan format data yang ada yaitu berupa tensor. Setelah dinormalisasi, tensor dataset yang terbentuk kemudian disimpan dalam *file.*

Tensor dataset di *file disk* kemudian diunggah ke *google drive* untuk proses selanjutnya di *google colab.* Tensor dataset dari *google drive* dimuat kemudian dataset evaluasi diambil secara acak dari 4400 data pelatihan dengan rasio 75% dari total data sehingga didapatkan 3.312 data evaluasi dan 1088 data uji. Semua data

pada tensor dataset pelatihan yaitu berjumlah 22.000 data digunakan sebagai dataset pelatihan.

```
import numpy as np
import tensorflow as tf
import os

from google.colab import drive
drive.mount('/gdrive', force_remount=True)
path1 = os.path.join("/gdrive/My Drive", "tensorflow_traindataset")
path2 = os.path.join("/gdrive/My Drive", "tensorflow_validatedataset")
with tf.device('CPU'):
    train_dataset = tf.data.Dataset.load(path1)
    test_dataset=tf.data.Dataset.load(path2)

ukuran=tf.data.experimental.cardinality(test_dataset).numpy()
test_size = int(ukuran * 0.25)
val_ds = test_dataset.skip(test_size)
test_ds = test_dataset.take(test_size)
```

Gambar 3.11 Kode program memuat tensor dataset dari google drive kemudian membagi data menjadi data pelatihan dan evaluasi.

3.2.3.2 Implementasi Model: Preparasi Data

Keterbatasan waktu dan *resource* dapat menyebabkan data atau citra yang terkumpul terbatas dan memiliki kualitas yang kurang baik, padahal kualitas data berpengaruh pada baik atau buruknya sebuah model. Untuk meningkatkan kualitas data maka perlu dilakukan augmentasi data pada citra yang telah terkumpul. Dilakukan augmentasi data dengan manipulasi saturasi, kontras, *hue, gama* dan *flip horizontal.*

```
pctpat=[]
aug=[]

for filel in os.listdir(fdirl):
  i=0
  files=os.path.join(fdirl, filel)
  for pct in os.listdir(files):
    nm=os.path.join(files, pct)
    pctpat.append(nm)
    x=PIL.Image.open(nm)
    flipped = tf.image.flip_left_right(x)
    huedl=tf.image.adjust_hue(x, -0.1, name=None)
    hued2=tf.image.adjust_hue(x, 0.1, name=None)
    saturated=tf.image.adjust_saturation(x, 0.8, name=None)
    contrsedl=tf.image.adjust_contrast(x, 1.5)
    contrsed2=tf.image.adjust_contrast(x, 0.7)
    gamaed=tf.image.adjust_gamma(x, gamma=1.5, gain=1)
    aug.append(x)
    aug.append(flipped)
    aug.append(huedl)
    aug.append(hued2)
    aug.append(saturated)
    aug.append(contrsedl)
    aug.append(contrsed2)
    aug.append(gamaed)
  for ag in aug:
    i+=1
    nms=filel+'_'+str(i)+'.jpg'
    svdir=os.path.join(fdir2, filel, nms)
    tf.keras.utils.save_img(svdir, ag)
  pcpat=[]
  aug=[]
```

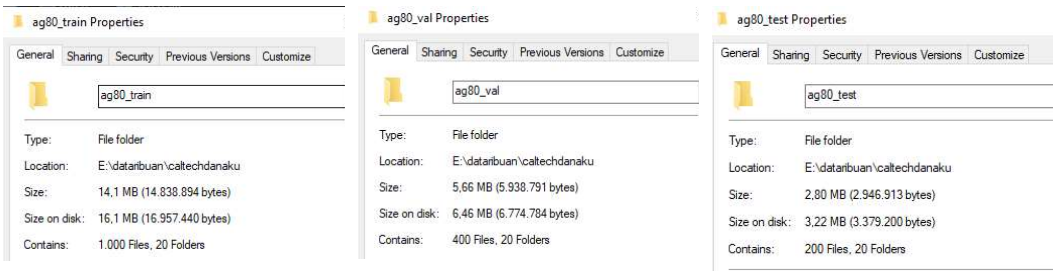
Gambar 3.12 Kode program augmentasi data

Original Flip Horizontal Hue

Saturasi Kontras Gama

Gambar 3.13 Contoh augmentasi data pada citra yang telah terkumpul

Terdapat total 1600 citra setelah dilakukan augmentasi data. Citra hasil augmentasi kemudian dibagi menjadi beberapa dataset sehingga terdapat 1000 citra pada dataset pelatihan, 400 citra pada dataset evaluasi dan 200 citra pada dataset uji. Dataset citra wajah kemudian dirubah menjadi tensor dataset dan dinormalisasi. Proses merubah menjadi tensor dataset dan normalisasi sama dengan sub bab 3.2.3.1.



Gambar 3.14 Pembagian dataset

3.2.4 MobileNetV2

Rancangan model mengikuti arsitektur MobileNetV2 namun memiliki struktur model yang berbeda yaitu terdapat perbedaan jumlah layer pada *channel-channel* tertentu sehingga perlu membangun *depthwise* dan *point wise convolutional layer* kemudian dibangun blok *bottleneck* dari kedua *layer* tersebut. Digunakan *library* dari tensorflow untuk membangun *layer-layer* tersebut.

```
import tensorflow as tf
from tensorflow.keras.layers import Conv2D, DepthwiseConv2D, ReLU, BatchNormalization,
                                    add, Softmax, AveragePooling2D, Dense, Input,
                                    GlobalAveragePooling2D
from tensorflow.keras.models import Model

def point_wisel(x, filters, t):
    dt=filters*t
    x = Conv2D(filters = dt, kernel_size = 1, strides = 1)(x)
    x = BatchNormalization()(x)
    x = ReLU(6)(x)
    return x
```

Gambar 3.15 Kode program *pointwise convolutional layer*

```
def depth_wise(x,strides):
    x = DepthwiseConv2D(kernel_size = 3, strides = strides, padding = 'same')(x)
    x = BatchNormalization()(x)
    x = ReLU(6)(x)
    return x

def point_wise2(x, out_channels):
    x = Conv2D(filters = out_channels, kernel_size = 1, strides = 1)(x)
    x = BatchNormalization()(x)
    x = ReLU(6)(x)
    return x

def Bottleneck(x, filters, strides, t, out_channels):
    y=point_wisel(x, filters, t)
    y=depth_wise(y, strides)
    y=point_wise2(y, out_channels)
    if y.shape[-1]==x.shape[-1]: y = add([x,y])
    return y
```

Gambar 3.16 Kode program *depthwise convolutional layer, pointwise convolutional layer dan blok bottleneck*

Setelah membangun fungsi *layer-layer* sesuai dengan arsitektur MobileNetV2, langkah selanjutnya adalah menyusun tiap-tiap layer model sesuai dengan rancangan struktur model yang dideskripsikan pada tabel 3.1. Pada penelitian ini, tidak digunakan *pre-trained model* MobileNetV2 dari keras. *Pre-trained model* MobileNetV2 dari keras telah memiliki bias dan bobot dari ImageNet yang telah dilatih terhadap berbagai macam citra seperti buah-buahan, berbagai macam bunga, berbagai macam binatang, dll. yang jumlahnya mencapai 1,4 juta citra dengan seribu kelas. Pada penelitian ini dilakukan pelatihan dari awal, model yang disusun belum memiliki bias dan bobot dan model yang dibangun hanya dilatih untuk menjalankan fungsi khusus yaitu mengenali wajah manusia. Model yang dibangun memiliki input spesifik yaitu citra wajah berukuran 224 × 224 × 3.

```
def MyMobileNetV2(input_image = (224,224,3), n_classes=50):
    input = Input(input_image)
    x = Conv2D(32,kernel_size=3,strides=(2,2),padding = 'same', use_bias=False)(input)
    x = BatchNormalization()(x)
    x = ReLU(6)(x)
    x = depth_wise(x,strides=1)
    x = point_wise2(x, out_channels=16)
    x = Bottleneck(x, t = 6, filters = x.shape[-1], out_channels = 24, strides = 2)
    x = Bottleneck(x, t = 6, filters = x.shape[-1], out_channels = 24, strides = 1)
    x = Bottleneck(x, t = 6, filters = x.shape[-1], out_channels = 32, strides = 2)
    x = Bottleneck(x, t = 6, filters = x.shape[-1], out_channels = 32, strides = 1)
    x = Bottleneck(x, t = 6, filters = x.shape[-1], out_channels = 32, strides = 1)
    x = Bottleneck(x, t = 6, filters = x.shape[-1], out_channels = 32, strides = 1)
    x = Bottleneck(x, t = 6, filters = x.shape[-1], out_channels = 64, strides = 2)
    x = Bottleneck(x, t = 6, filters = x.shape[-1], out_channels = 64, strides = 1)
    x = Bottleneck(x, t = 6, filters = x.shape[-1], out_channels = 64, strides = 1)
    x = Bottleneck(x, t = 6, filters = x.shape[-1], out_channels = 64, strides = 1)
    x = Bottleneck(x, t = 6, filters = x.shape[-1], out_channels = 96, strides = 1)
    x = Bottleneck(x, t = 6, filters = x.shape[-1], out_channels = 96, strides = 1)
    x = Bottleneck(x, t = 6, filters = x.shape[-1], out_channels = 96, strides = 1)
    x = Bottleneck(x, t = 6, filters = x.shape[-1], out_channels = 160, strides = 2)
    x = Bottleneck(x, t = 6, filters = x.shape[-1], out_channels = 160, strides = 1)
    x = Bottleneck(x, t = 6, filters = x.shape[-1], out_channels = 320, strides = 1)
    #1*1 conv
    x = Conv2D(filters = 1280,kernel_size = 1,padding='same',use_bias=False)(x)
    x = BatchNormalization()(x)
    x = ReLU(6)(x)
    #AvgPool 7*7
    x = GlobalAveragePooling2D()(x)
    output = Dense(n_classes,activation='softmax')(x)
    model = Model(input, output)
    return model
```

Gambar 3.17 Kode program MobileNetV2 sesuai rancangan struktur model

3.2.5 *Transfer Learning* dan *Fine and Tuning*

Jumlah dan identitas pengguna pengenalan wajah dapat berbeda-beda atau berubah-ubah pada tiap kasusnya oleh karena itu perlu dilakukan penyesuaian model. Penyesuaian model dapat dilakukan pada *layer* terakhir yaitu pada *fully connected layer* atau pada keseluruhan layer namun dengan perubahan yang sangat kecil. Jumlah identitas pada dataset dari KomNet adalah 50 sedangkan jumlah identitas pada dataset Caltech dan pengambilan citra secara mandiri adalah 20 sehingga *layer* terakhir pada model yang berupa *fully connected layer* dengan output 50 tersebut diambil dan kemudian diganti dengan *fully connected layer* dengan output 20.

```
pth= os.path.join("/gdrive/My Drive", "mymodel.h5")
my_model = tf.keras.models.load_model(pth)
my_model.trainable=False
model2 = Model(model.input, my_model.layers[-2].output)
modelfix = models.Sequential()
modelfix.add(layers.Dense(20, activation='softmax'))
model_fix=tf.keras.Sequential([model2, modelfix])
```

Gambar 3.18 Kode program memuat model kemudian mengambil *fully connected layer* dari model yang telah dibangun dan menggantinya dengan *fully connected* layer output 20

Penyesuaian model juga dapat dilakukan pada beberapa *layer* terakhir dengan *learning rate* yang lebih kecil daripada *learning rate* membangun model.

```
for idx in range(len(model.layers)):
  if (idx < startlayer) or ('batch_normalization' in model.get_layer(index=idx).name):
    model.get_layer(index=idx).trainable=False
```

Gambar 3.19 Kode program memuat penyesuaian model pada beberapa layer terakhir

Kode program di atas hanya memperbolehkan perubahan dimulai pada *layer* tertentu dan *layer batchnorm* mutlak tidak boleh diubah. Setelah model dimuat dan dilakukan perubahan pada *layer,* langkah berikutnya adalah mengompile kembali model tersebut dan melatihnya terhadap dataset pelatihan yang terdiri dari citra wajah pengguna pengenalan wajah.

BAB IV
UJI COBA DAN PEMBAHASAN

4.1 Skenario Uji Coba

Pada penelitian ini digunakan 4 skenario pengujian dengan *learning rate* dan *epoch* berbeda yang dijelaskan pada tabel 4.1.

Tabel 4.1 Skenario uji coba

Skenario	*Learning Rate*	*Epoch*
1	0,001	100
2	0,002	100
3	0,01 dan 0,001	5 dan 8
4	0,01 dan 0,001	5 dan 30

Adapun langkah-langkah pengujian yang dilakukan adalah sebagai berikut.

1. Pelatihan yaitu model dihadapkan dengan data yang telah disiapkan. Diharapkan model dapat mempelajari data sehingga model yang terbentuk dapat menjalankan fungsi tertentu dengan baik. Model yang telah dilatih kemudian disimpan untuk tahap selanjutnya.
2. Pengujian yaitu tahap menguji model terhadap data baru yang belum pernah dilihat oleh model. Model diujikan dengan dataset uji hingga didapatkan label prediksi. Label hasil prediksi model kemudian dibandingan dengan label sebenarnya kemudian dihitung peforma model menggunakan *confusion matrix.*

3. *Transfer learning* model yaitu model yang telah dilatih dimuat kembali dan dibekukan. Dilakukan perubahan hanya pada *layer* terakhir dan model kembali dilatih terhadap dataset pengguna pengenalan wajah.
4. Pengujian model hasil *transfer learning* yaitu model hasil *transfer learning* dihadapkan kepada dataset uji pengguna pengenalan wajah hingga didapatkan label prediksi. Label hasil prediksi model kemudian dibandingan dengan label sebenarnya kemudian dihitung peforma model menggunakan *confusion matrix.*
5. *Fine and tuning* model yaitu model hasil *transfer learning* dimuat kembali dan dibekukan pada beberapa *layer* awal. Dilakukan perubahan hanya pada beberapa *layer* terakhir dan model kembali dilatih terhadap dataset pengguna pengenalan wajah.
6. Pengujian model hasil *fine and tuning* yaitu model hasil *fine and tuning* dihadapkan kepada dataset uji pengguna pengenalan wajah hingga didapatkan label prediksi. Label hasil prediksi model kemudian dibandingan dengan label sebenarnya kemudian dihitung peforma model menggunakan *confusion matrix.*

4.2 Hasil Uji Coba

Dilakukan uji coba sesuai skenario uji coba sehingga didapatkan model dengan peforma beragam. Model dikompile menggunakan *loss function* s*parse categorical crossentropy* dan fungsi optimasi *stohastic gradient descent.* Pelatihan model dilakukan dengan *learning rate* dan *epoch* yang berbeda-beda hingga didapatkan hasil sebagai berikut.

1. Skenario 1 model dilatih dengan *learning rate* 0,001 dan *epoch* 100.
   a. Pelatihan menghasilkan model dengan akurasi 88,10% dan nilai *loss* 0,4522 terhadap datastet validasi. Melatih 25.000 data membutuhkan memori besar sehingga pelatihan dilakukan secara bertahap. Tahap pertama dilakukan dengan *learning rate* 0,001 dan *epoch* 50, tahap kedua dilakukan dengan *learning rate* 0,001 dan *epoch* 50. Berikut adalah rincian 5 *epoch* terakhir beserta total lama waktu yang dibutuhkan pada pelatihan pelatihan tahap pertama dan kedua.

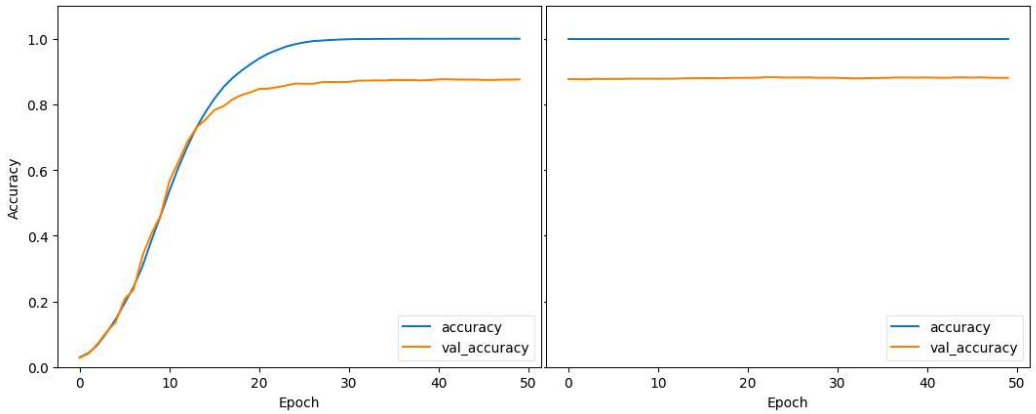
```
Epoch 46/50
688/688 [==============================] - 263s 381ms/step - loss: 0.0247 - accuracy: 1.0000 - val_loss: 0.4639 - val_accuracy: 0.8747
Epoch 47/50
688/688 [==============================] - 212s 308ms/step - loss: 0.0235 - accuracy: 1.0000 - val_loss: 0.4637 - val_accuracy: 0.8747
Epoch 48/50
688/688 [==============================] - 195s 281ms/step - loss: 0.0224 - accuracy: 1.0000 - val_loss: 0.4635 - val_accuracy: 0.8753
Epoch 49/50
688/688 [==============================] - 208s 299ms/step - loss: 0.0214 - accuracy: 1.0000 - val_loss: 0.4635 - val_accuracy: 0.8756
Epoch 50/50
688/688 [==============================] - 203s 294ms/step - loss: 0.0205 - accuracy: 1.0000 - val_loss: 0.4632 - val_accuracy: 0.8759
CPU times: user 1h 56s, sys: 12min 16s, total: 1h 13min 13s
Wall time: 3h 24min 55s
```

Gambar 4.1 Lima epoch terakhir pelatihan tahap pertama pada skenario 1

```
Epoch 46/50
688/688 [==============================] - 228s 329ms/step - loss: 0.0065 - accuracy: 1.0000 - val_loss: 0.4524 - val_accuracy: 0.8816
Epoch 47/50
688/688 [==============================] - 115s 166ms/step - loss: 0.0064 - accuracy: 1.0000 - val_loss: 0.4523 - val_accuracy: 0.8822
Epoch 48/50
688/688 [==============================] - 116s 168ms/step - loss: 0.0063 - accuracy: 1.0000 - val_loss: 0.4523 - val_accuracy: 0.8813
Epoch 49/50
688/688 [==============================] - 217s 312ms/step - loss: 0.0062 - accuracy: 1.0000 - val_loss: 0.4524 - val_accuracy: 0.8810
Epoch 50/50
688/688 [==============================] - 215s 312ms/step - loss: 0.0061 - accuracy: 1.0000 - val_loss: 0.4522 - val_accuracy: 0.8810
CPU times: user 1h 28s, sys: 12min 20s, total: 1h 12min 48s
Wall time: 2h 47min 11s
```

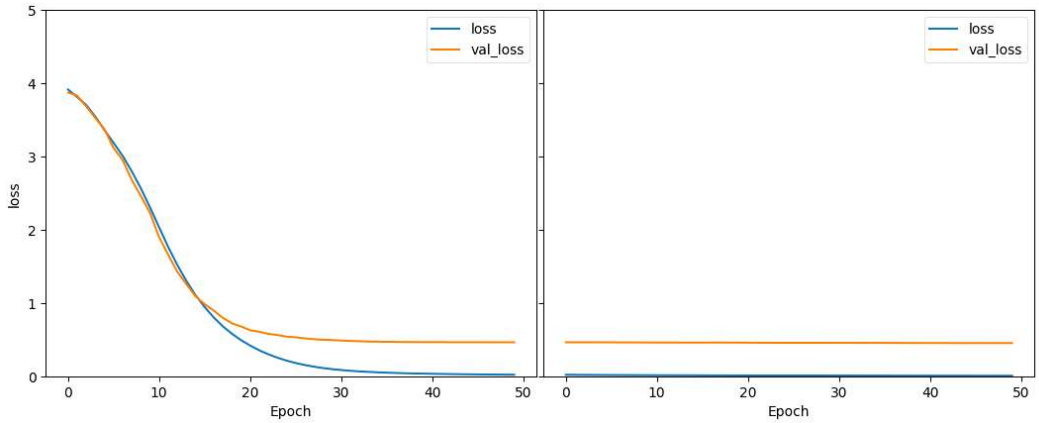
Gambar 4.2 Lima epoch terakhir pelatihan tahap kedua pada skenario 1

Grafik akurasi pelatihan tahap pertama dan kedua dapat digabungkan menjadi gambar 4.3.



Gambar 4.3 Grafik akurasi pelatihan dengan *learning rate* 0,001 *epoch* 100

Grafik *loss* pelatihan tahap pertama dan kedua dapat digabungkan menjadi gambar 4.4.



Gambar 4.4 Grafik *loss* pelatihan dengan *learning rate* 0,001 *epoch* 100

b. Model hasil pelatihan dihadapkan dengan dataset uji yang terdiri dari 1088 citra dengan 50 identitas berbeda. Jumlah citra tiap identitas berbeda-beda. Hasil perhitungan peforma model secara rinci dapat dilihat pada lampiran, sedangkan rangkuman peforma model menggunakan *confusion matrix* dapat dilihat pada tabel 4.2.

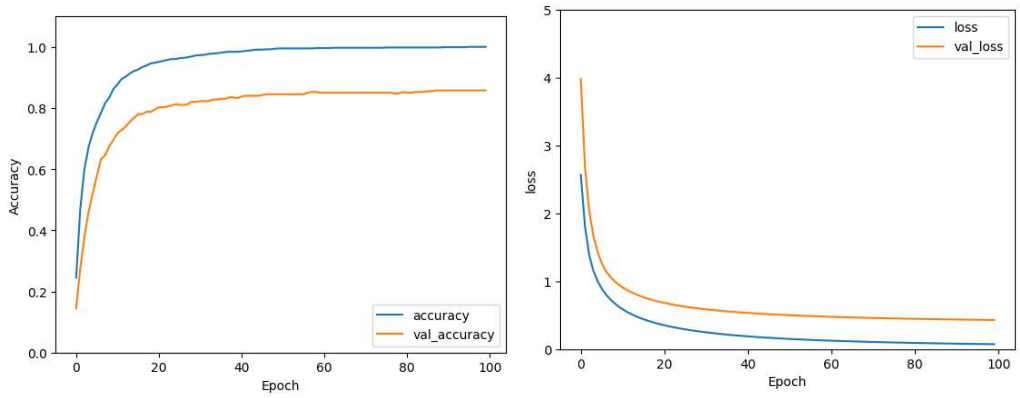
Tabel 4.2 Peforma model hasil skenario 1

Skenario	macro avg			weighted avg			akurasi
	precision	recall	f1-score	precision	recall	f1-score	
1	0,88	0,88	0,88	0,89	0,88	0,88	0,88

c. Dilakukan *transfer learning* pada model yang telah dilatih, dikompile kembali dengan *learning rate* 0,08 dan dilatih dengan *epoch* 100. *Transfer learning* model skenario 1 menghasilkan akurasi 85,75% dan nilai *loss* 0,4297 terhadap dataset validasi.

```
Epoch 96/100
32/32 [==============================] - 2s 74ms/step - loss: 0.0754 - accuracy: 1.0000 - val_loss: 0.4326 - val_accuracy: 0.8575
Epoch 97/100
32/32 [==============================] - 2s 75ms/step - loss: 0.0745 - accuracy: 1.0000 - val_loss: 0.4318 - val_accuracy: 0.8575
Epoch 98/100
32/32 [==============================] - 3s 97ms/step - loss: 0.0737 - accuracy: 1.0000 - val_loss: 0.4311 - val_accuracy: 0.8575
Epoch 99/100
32/32 [==============================] - 3s 82ms/step - loss: 0.0729 - accuracy: 1.0000 - val_loss: 0.4304 - val_accuracy: 0.8575
Epoch 100/100
32/32 [==============================] - 2s 74ms/step - loss: 0.0721 - accuracy: 1.0000 - val_loss: 0.4297 - val_accuracy: 0.8575
CPU times: user 2min 35s, sys: 1min 4s, total: 3min 40s
Wall time: 5min 6s
```

Gambar 4.5 Lima epoch terakhir *transfer learning* skenario 1



Gambar 4.6 Grafik akurasi dan *loss* pada pelatihan *transfer learning* model skenario 1

d. Model hasil *transfer learning* dihadapkan dengan dataset uji yang terdiri dari 200 citra dengan 20 identitas berbeda. Hasil perhitungan peforma model secara rinci dapat dilihat pada lampiran, sedangkan rangkuman peforma model menggunakan *confusion matrix* dapat dilihat pada tabel 4.3.

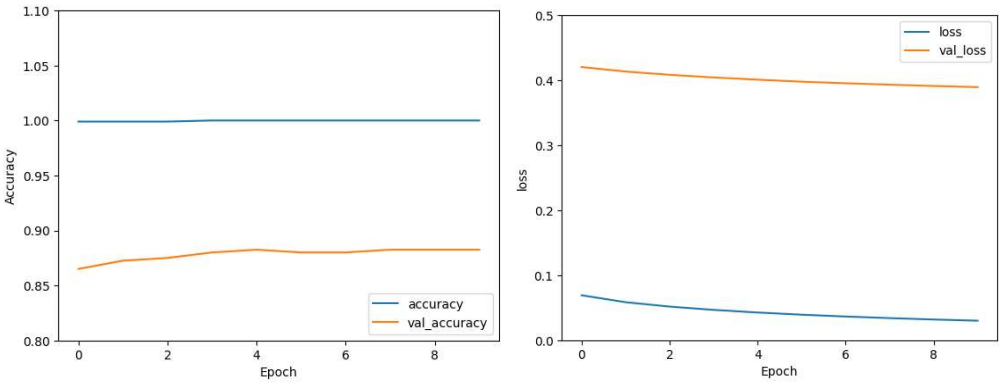
Tabel 4.3 Peforma model hasil *transfer learning* model skenario 1

Skenario	macro avg			weighted avg			akurasi
	precision	recall	f1-score	precision	recall	f1-score	
1	0,88	0,87	0,87	0,88	0,87	0,87	0,87

e. Dilakukan *fine and tuning* pada model hasil *transfer learning,* dikompile kembali dengan *learning rate* lebih kecil yaitu *base learning rate* dibagi dengan 10 dan dilatih dengan *epoch* 10 sehingga didapatkan nilai *learning rate* sebesar 0,0001. Dilakukan perubahan hanya pada beberapa *layer* terakhir. Terdapat total 167 *layer* pada model dan perubahan hanya dilakukan pada *layer* ke-132 hingga *layer* terakhir. Semua *layer batchnorm* pada model tetap dibekukan. *Fine and tuning* model skenario 1 menghasilkan akurasi 88,25% dan nilai *loss* 0,3895 terhadap dataset validasi.

```
Epoch 1/10
32/32 [==============================] - 32s 506ms/step - loss: 0.0691 - accuracy: 0.9990 - val_loss: 0.4203 - val_accuracy: 0.8650
Epoch 2/10
32/32 [==============================] - 3s 83ms/step - loss: 0.0584 - accuracy: 0.9990 - val_loss: 0.4134 - val_accuracy: 0.8725
Epoch 3/10
32/32 [==============================] - 3s 107ms/step - loss: 0.0516 - accuracy: 0.9990 - val_loss: 0.4084 - val_accuracy: 0.8750
Epoch 4/10
32/32 [==============================] - 3s 83ms/step - loss: 0.0466 - accuracy: 1.0000 - val_loss: 0.4043 - val_accuracy: 0.8800
Epoch 5/10
32/32 [==============================] - 3s 84ms/step - loss: 0.0426 - accuracy: 1.0000 - val_loss: 0.4009 - val_accuracy: 0.8825
Epoch 6/10
32/32 [==============================] - 3s 84ms/step - loss: 0.0392 - accuracy: 1.0000 - val_loss: 0.3978 - val_accuracy: 0.8800
Epoch 7/10
32/32 [==============================] - 4s 112ms/step - loss: 0.0364 - accuracy: 1.0000 - val_loss: 0.3954 - val_accuracy: 0.8800
Epoch 8/10
32/32 [==============================] - 3s 83ms/step - loss: 0.0339 - accuracy: 1.0000 - val_loss: 0.3932 - val_accuracy: 0.8825
Epoch 9/10
32/32 [==============================] - 3s 82ms/step - loss: 0.0318 - accuracy: 1.0000 - val_loss: 0.3912 - val_accuracy: 0.8825
Epoch 10/10
32/32 [==============================] - 4s 113ms/step - loss: 0.0300 - accuracy: 1.0000 - val_loss: 0.3895 - val_accuracy: 0.8825
CPU times: user 26.2 s, sys: 7.94 s, total: 34.2 s
Wall time: 1min 19s
```

Gambar 4.7 Rincian *epoch fine and tuning* skenario 1



Gambar 4.8 Grafik akurasi dan *loss* pada pelatihan *fine and tuning* model skenario 1

f. Model hasil *fine and tuning* dihadapkan dengan dataset uji yang terdiri dari 200 citra dengan 20 identitas berbeda. Hasil perhitungan peforma model hasil *fine and tuning* secara rinci dapat dilihat pada lampiran, sedangkan rangkuman peforma model menggunakan *confusion matrix* dapat dilihat pada tabel 4.4.

Tabel 4.4 Rangkuman peforma model hasil *fine and tuning* pada skenario 1

Skenario	macro avg			weighted avg			akurasi
	precision	recall	f1-score	precision	recall	f1-score	
1	0,88	0,87	0,87	0,88	0,87	0,87	0,87

2. Skenario 2 model dilatih dengan *learning rate* 0,002 dan *epoch* 100.

a. Pelatihan menghasilkan model dengan akurasi 89,52% dan nilai *loss* 0,3890 terhadap datastet validasi. Melatih 25.000 data membutuhkan memori besar sehingga pelatihan dilakukan secara bertahap. Tahap pertama dilakukan dengan *learning rate* 0,002 dan *epoch* 50, tahap kedua dilakukan dengan *learning rate* 0,002 dan *epoch* 50. Berikut adalah rincian 5 *epoch* terakhir beserta total lama waktu yang dibutuhkan pada pelatihan pelatihan tahap pertama dan kedua.

```
Epoch 46/50
688/688 [==============================] - 118s 171ms/step - loss: 0.0076 - accuracy: 1.0000 - val_loss: 0.3942 - val_accuracy: 0.8913
Epoch 47/50
688/688 [==============================] - 233s 337ms/step - loss: 0.0074 - accuracy: 1.0000 - val_loss: 0.3942 - val_accuracy: 0.8907
Epoch 48/50
688/688 [==============================] - 219s 318ms/step - loss: 0.0071 - accuracy: 1.0000 - val_loss: 0.3940 - val_accuracy: 0.8904
Epoch 49/50
688/688 [==============================] - 230s 332ms/step - loss: 0.0069 - accuracy: 1.0000 - val_loss: 0.3936 - val_accuracy: 0.8919
Epoch 50/50
688/688 [==============================] - 216s 314ms/step - loss: 0.0067 - accuracy: 1.0000 - val_loss: 0.3930 - val_accuracy: 0.8919
CPU times: user 1h 2min 18s, sys: 12min 55s, total: 1h 15min 13s
Wall time: 2h 50min 10s
```

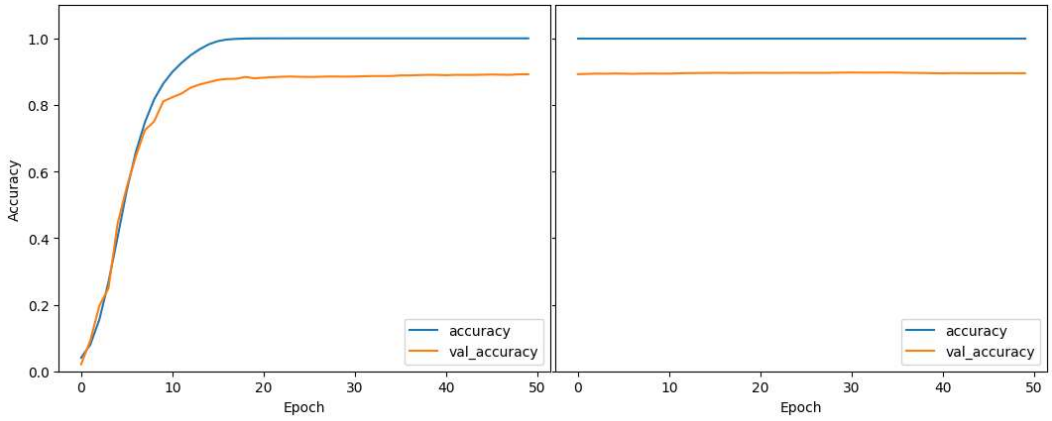
Gambar 4.9 Lima *epoch* terakhir pelatihan tahap pertama pada skenario 2

Berikut adalah rincian 5 *epoch* terakhir beserta total lama waktu yang dibutuhkan pada pelatihan tahap kedua.

```
Epoch 46/50
688/688 [==============================] - 201s 291ms/step - loss: 0.0026 - accuracy: 1.0000 - val_loss: 0.3890 - val_accuracy: 0.8952
Epoch 47/50
688/688 [==============================] - 210s 297ms/step - loss: 0.0026 - accuracy: 1.0000 - val_loss: 0.3890 - val_accuracy: 0.8952
Epoch 48/50
688/688 [==============================] - 205s 298ms/step - loss: 0.0026 - accuracy: 1.0000 - val_loss: 0.3890 - val_accuracy: 0.8955
Epoch 49/50
688/688 [==============================] - 233s 338ms/step - loss: 0.0025 - accuracy: 1.0000 - val_loss: 0.3890 - val_accuracy: 0.8952
Epoch 50/50
688/688 [==============================] - 207s 294ms/step - loss: 0.0025 - accuracy: 1.0000 - val_loss: 0.3890 - val_accuracy: 0.8952
CPU times: user 58min 23s, sys: 12min 14s, total: 1h 10min 37s
Wall time: 2h 56min 17s
```

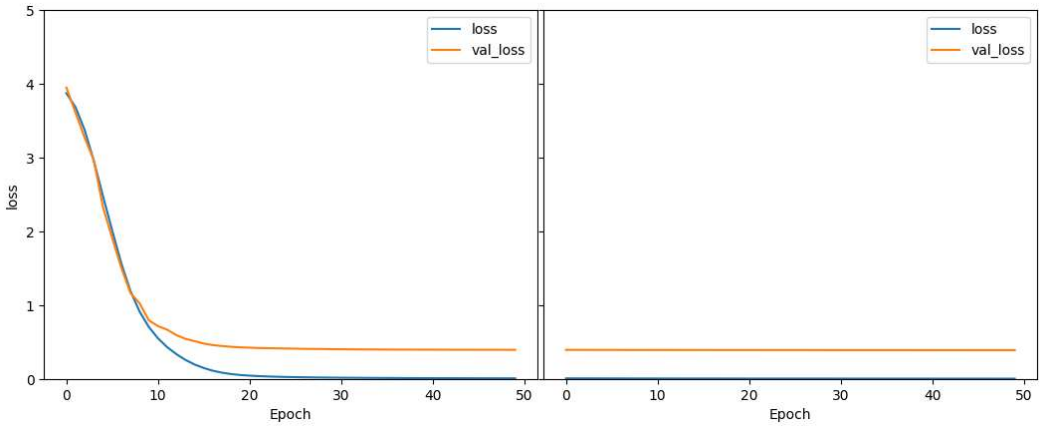
Gambar 4.10 Lima *epoch* terakhir pelatihan tahap kedua pada skenario 2

Grafik akurasi pelatihan tahap pertama dan kedua dapat digabungkan menjadi gambar 4.11.



Gambar 4.11 Grafik akurasi pelatihan dengan *learning rate* 0,002 *epoch* 100

Grafik *loss* pelatihan tahap pertama dan kedua dapat digabungkan menjadi gambar 4.12.



Gambar 4.12 Grafik *loss* pelatihan dengan *learning rate* 0,002 *epoch* 100

b. Model hasil pelatihan dihadapkan dengan dataset uji yang terdiri dari 1088 citra dengan 50 identitas berbeda. Jumlah citra tiap identitas berbeda-beda. Hasil perhitungan peforma model secara rinci dapat dilihat pada lampiran, sedangkan rangkuman peforma model menggunakan *confusion matrix* dapat dilihat pada tabel 4.5.

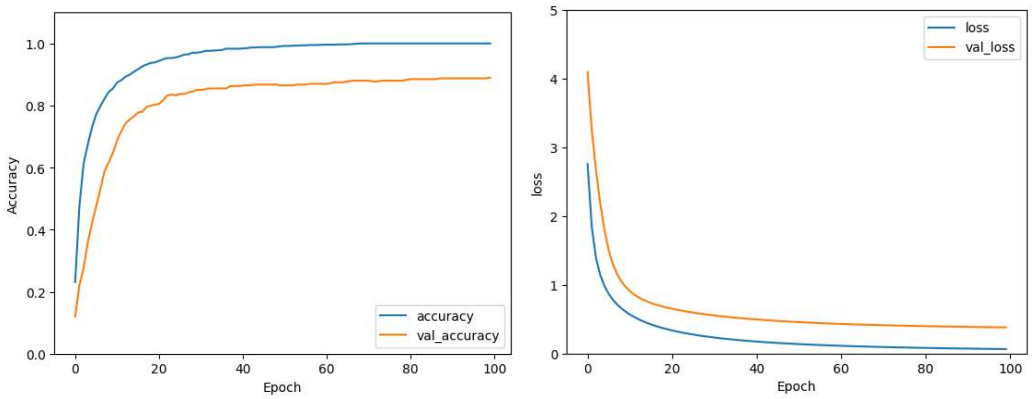
Tabel 4.5 Peforma model hasil skenario 2

Skenario	macro avg			weighted avg			akurasi
	precision	recall	f1-score	precision	recall	f1-score	
2	0,90	0,90	0,89	0,90	0,90	0,90	0,90

c. Dilakukan *transfer learning* pada model yang telah dilatih, dikompile kembali dengan *learning rate* 0,08 dan dilatih dengan *epoch* 100 sehingga didapatkan hasil sebagai berikut. *Transfer learning* model skenario 2 menghasilkan akurasi 89% dan nilai *loss* 0,3799 terhadap dataset validasi. Berikut adalah rincian 5 *epoch* terakhir beserta total lama waktu yang dibutuhkan untuk pelatihan.

```
Epoch 96/100
32/32 [==============================] - 3s 82ms/step - loss: 0.0672 - accuracy: 1.0000 - val_loss: 0.3831 - val_accuracy: 0.8875
Epoch 97/100
32/32 [==============================] - 3s 87ms/step - loss: 0.0664 - accuracy: 1.0000 - val_loss: 0.3823 - val_accuracy: 0.8875
Epoch 98/100
32/32 [==============================] - 2s 74ms/step - loss: 0.0657 - accuracy: 1.0000 - val_loss: 0.3815 - val_accuracy: 0.8875
Epoch 99/100
32/32 [==============================] - 2s 74ms/step - loss: 0.0649 - accuracy: 1.0000 - val_loss: 0.3807 - val_accuracy: 0.8875
Epoch 100/100
32/32 [==============================] - 2s 73ms/step - loss: 0.0642 - accuracy: 1.0000 - val_loss: 0.3799 - val_accuracy: 0.8900
CPU times: user 2min 34s, sys: 1min 7s, total: 3min 41s
Wall time: 5min 22s
```

Gambar 4.13 Lima epoch terakhir *transfer learning* skenario 2



Gambar 4.14 Grafik akurasi dan *loss* pada pelatihan *transfer learning* model skenario 2

d. Model hasil *transfer learning* dihadapkan dengan dataset uji yang terdiri dari 200 citra dengan 20 identitas berbeda. Hasil perhitungan peforma model secara rinci dapat dilihat pada lampiran, sedangkan rangkuman peforma model menggunakan *confusion matrix* dapat dilihat pada tabel 4.6.

Tabel 4.6 Peforma model hasil *transfer learning* model skenario 2

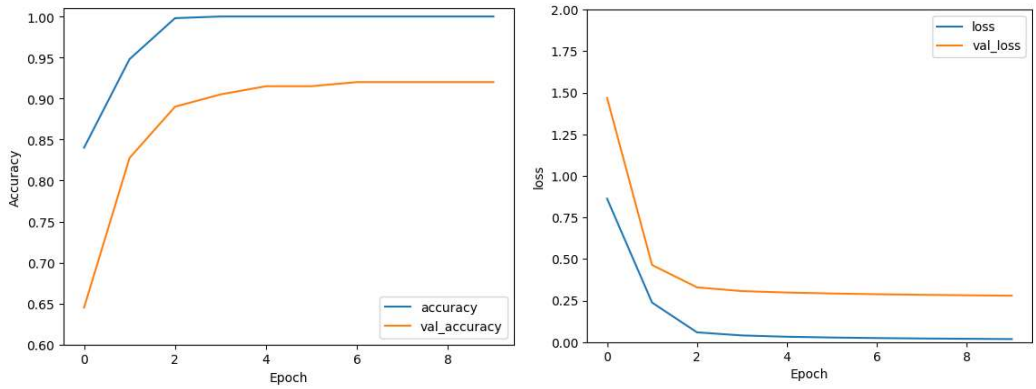
Skenario	macro avg			weighted avg			akurasi
	precision	recall	f1-score	precision	recall	f1-score	
2	0,89	0,88	0,88	0,89	0,88	0,88	0,88

e. Dilakukan *fine and tuning* pada model hasil *transfer learning,* dikompile kembali dengan *learning rate* lebih kecil yaitu *base learning rate* dibagi dengan 10 dan dilatih dengan *epoch* 10 sehingga didapatkan nilai *learning rate* sebesar 0,0002. Dilakukan perubahan hanya pada beberapa *layer* terakhir.

Terdapat total 167 *layer* pada model dan perubahan hanya dilakukan pada *layer* ke-132 hingga *layer* terakhir. Semua *layer batchnorm* pada model tetap dibekukan. *Fine and tuning* model skenario 2 menghasilkan akurasi 92% dan nilai *loss* 0,2791 terhadap dataset validasi. Berikut adalah rincian tiap *epoch* beserta total lama waktu yang dibutuhkan untuk pelatihan.

```
Epoch 1/10
32/32 [==============================] - 14s 278ms/step - loss: 0.8627 - accuracy: 0.8400 - val_loss: 1.4690 - val_accuracy: 0.6450
Epoch 2/10
32/32 [==============================] - 3s 100ms/step - loss: 0.2376 - accuracy: 0.9480 - val_loss: 0.4637 - val_accuracy: 0.8275
Epoch 3/10
32/32 [==============================] - 3s 89ms/step - loss: 0.0585 - accuracy: 0.9980 - val_loss: 0.3292 - val_accuracy: 0.8900
Epoch 4/10
32/32 [==============================] - 3s 84ms/step - loss: 0.0399 - accuracy: 1.0000 - val_loss: 0.3066 - val_accuracy: 0.9050
Epoch 5/10
32/32 [==============================] - 3s 101ms/step - loss: 0.0320 - accuracy: 1.0000 - val_loss: 0.2980 - val_accuracy: 0.9150
Epoch 6/10
32/32 [==============================] - 3s 90ms/step - loss: 0.0273 - accuracy: 1.0000 - val_loss: 0.2923 - val_accuracy: 0.9150
Epoch 7/10
32/32 [==============================] - 3s 84ms/step - loss: 0.0240 - accuracy: 1.0000 - val_loss: 0.2878 - val_accuracy: 0.9200
Epoch 8/10
32/32 [==============================] - 3s 83ms/step - loss: 0.0216 - accuracy: 1.0000 - val_loss: 0.2844 - val_accuracy: 0.9200
Epoch 9/10
32/32 [==============================] - 3s 103ms/step - loss: 0.0196 - accuracy: 1.0000 - val_loss: 0.2815 - val_accuracy: 0.9200
Epoch 10/10
32/32 [==============================] - 3s 84ms/step - loss: 0.0180 - accuracy: 1.0000 - val_loss: 0.2791 - val_accuracy: 0.9200
CPU times: user 25.4 s, sys: 6.99 s, total: 32.4 s
Wall time: 49.7 s
```

Gambar 4.15 Rincian *epoch fine and tuning* skenario 2



Gambar 4.16 Grafik akurasi dan *loss* pada pelatihan *fine and tuning* model skenario 2

f. Model hasil *fine and tuning* dihadapkan dengan dataset uji yang terdiri dari 200 citra dengan 20 identitas berbeda. Hasil perhitungan peforma model hasil *fine and tuning* secara rinci dapat dilihat pada lampiran, sedangkan rangkuman peforma model menggunakan *confusion matrix* dapat dilihat pada tabel 4.7.

Tabel 4.7 Rangkuman peforma model hasil *fine and tuning* pada skenario 2

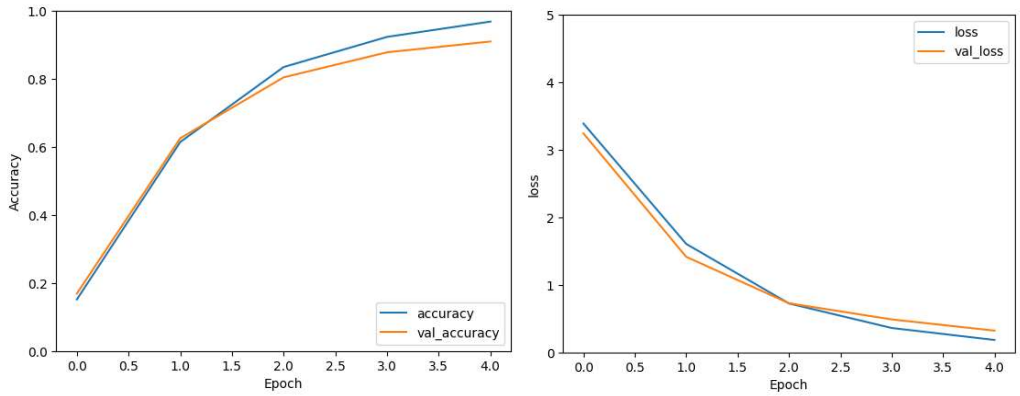
Skenario	macro avg			weighted avg			akurasi
	precision	recall	f1-score	precision	recall	f1-score	
2	0,91	0,90	0,90	0,91	0,91	0,90	0,91

3. Skenario 3 model dilatih dengan *learning rate* 0,01 dan 0,001; *epoch* 5 dan 8.

   a. Pelatihan model dilakukan dengan dua tahap. Pelatihan tahap pertama dilakukan dengan *learning rate* 0,01 dan *epoch 5*. Model hasil tahap pertama disimpan kemudian dilanjutkan ke tahap kedua dilakukan pelatihan dengan *learning rate* berbeda yaitu 0,001 dan *epoch* 8. Pelatihan model tahap pertama menghasilkan model dengan tingkat akurasi 91% dan *loss* sebesar 0,3217 terhadap dataset validasi.

```
Epoch 1/5
688/688 [==============================] - 214s 267ms/step - loss: 3.3892 - accuracy: 0.1517 - val_loss: 3.2428 - val_accuracy: 0.1688
Epoch 2/5
688/688 [==============================] - 219s 317ms/step - loss: 1.6065 - accuracy: 0.6145 - val_loss: 1.4154 - val_accuracy: 0.6256
Epoch 3/5
688/688 [==============================] - 260s 376ms/step - loss: 0.7242 - accuracy: 0.8350 - val_loss: 0.7271 - val_accuracy: 0.8046
Epoch 4/5
688/688 [==============================] - 274s 389ms/step - loss: 0.3607 - accuracy: 0.9236 - val_loss: 0.4877 - val_accuracy: 0.8783
Epoch 5/5
688/688 [==============================] - 285s 414ms/step - loss: 0.1838 - accuracy: 0.9686 - val_loss: 0.3217 - val_accuracy: 0.9100
CPU times: user 6min 32s, sys: 1min 5s, total: 7min 38s
Wall time: 23min 10s
```

Gambar 4.17 Rincian *epoch* pelatihan tahap pertama pada skenario 3

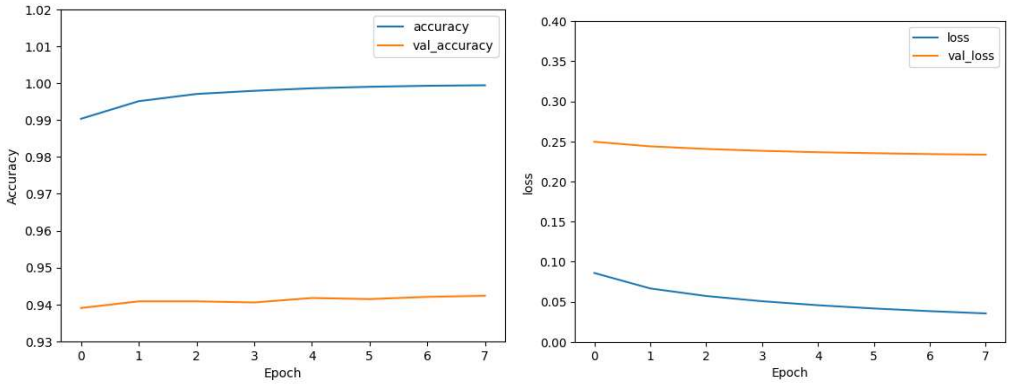


Gambar 4.18 Grafik akurasi dan *loss* pada pelatihan tahap pertama dengan *learning rate* 0,01 dan *epoch* 5

Pelatihan model pada tahap kedua menghasilkan model dengan tingkat akurasi 94,23% dan *loss* sebesar 0,2334 pada dataset validasi.

```
Epoch 1/8
688/688 [==============================] - 219s 291ms/step - loss: 0.0858 - accuracy: 0.9904 - val_loss: 0.2495 - val_accuracy: 0.9390
Epoch 2/8
688/688 [==============================] - 252s 358ms/step - loss: 0.0665 - accuracy: 0.9951 - val_loss: 0.2438 - val_accuracy: 0.9408
Epoch 3/8
688/688 [==============================] - 237s 341ms/step - loss: 0.0571 - accuracy: 0.9971 - val_loss: 0.2405 - val_accuracy: 0.9408
Epoch 4/8
688/688 [==============================] - 238s 344ms/step - loss: 0.0506 - accuracy: 0.9980 - val_loss: 0.2382 - val_accuracy: 0.9405
Epoch 5/8
688/688 [==============================] - 247s 358ms/step - loss: 0.0456 - accuracy: 0.9986 - val_loss: 0.2365 - val_accuracy: 0.9417
Epoch 6/8
688/688 [==============================] - 223s 322ms/step - loss: 0.0415 - accuracy: 0.9990 - val_loss: 0.2352 - val_accuracy: 0.9414
Epoch 7/8
688/688 [==============================] - 123s 178ms/step - loss: 0.0382 - accuracy: 0.9993 - val_loss: 0.2341 - val_accuracy: 0.9420
Epoch 8/8
688/688 [==============================] - 250s 363ms/step - loss: 0.0354 - accuracy: 0.9995 - val_loss: 0.2334 - val_accuracy: 0.9423
CPU times: user 10min 21s, sys: 2min 6s, total: 12min 28s
Wall time: 31min 9s
```

Gambar 4.19 Rincian *epoch* pelatihan tahap kedua pada skenario 3



Gambar 4.20 Grafik akurasi dan *loss* pada pelatihan tahap kedua dengan *learning rate* 0,001 dan *epoch* 8

b. Model hasil pelatihan dihadapkan dengan dataset uji yang terdiri dari 1088 citra dengan 50 identitas berbeda. Jumlah citra tiap identitas berbeda-beda. Hasil perhitungan peforma model secara rinci dapat dilihat pada lampiran, sedangkan rangkuman peforma model menggunakan *confusion matrix* dapat dilihat pada tabel 4.8.

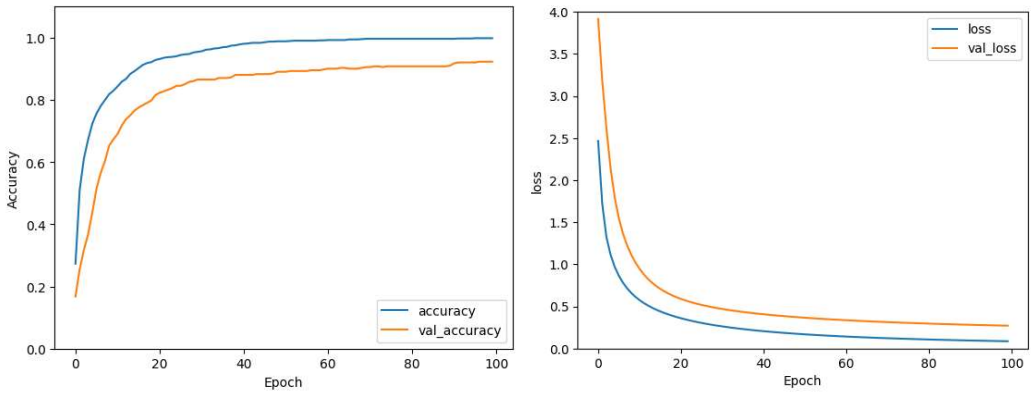
Tabel 4.8 Peforma model hasil skenario 3

Skenario	macro avg			weighted avg			akurasi
	precision	recall	f1-score	precision	recall	f1-score	
3	0,94	0,94	0,94	0,95	0,94	0,94	0,94

c. Dilakukan *transfer learning* pada model yang telah dilatih, dikompile kembali dengan *learning rate* 0,08 dan dilatih dengan *epoch* 100 sehingga didapatkan hasil sebagai berikut. *Transfer learning* model skenario 3 menghasilkan akurasi 92,25% dan nilai *loss* 0,2706 terhadap dataset validasi. Berikut adalah rincian 5 *epoch* terakhir beserta total lama waktu yang dibutuhkan untuk pelatihan.

```
Epoch 96/100
32/32 [==============================] - 3s 89ms/step - loss: 0.0895 - accuracy: 0.9980 - val_loss: 0.2751 - val_accuracy: 0.9200
Epoch 97/100
32/32 [==============================] - 3s 77ms/step - loss: 0.0886 - accuracy: 0.9980 - val_loss: 0.2740 - val_accuracy: 0.9225
Epoch 98/100
32/32 [==============================] - 3s 78ms/step - loss: 0.0876 - accuracy: 0.9980 - val_loss: 0.2728 - val_accuracy: 0.9225
Epoch 99/100
32/32 [==============================] - 3s 77ms/step - loss: 0.0867 - accuracy: 0.9980 - val_loss: 0.2717 - val_accuracy: 0.9225
Epoch 100/100
32/32 [==============================] - 3s 99ms/step - loss: 0.0858 - accuracy: 0.9980 - val_loss: 0.2706 - val_accuracy: 0.9225
CPU times: user 2min 41s, sys: 1min 7s, total: 3min 49s
Wall time: 5min 43s
```

Gambar 4.21 Lima *epoch* terakhir *transfer learning* skenario 3



Gambar 4.22 Grafik akurasi dan *loss* pada pelatihan *transfer learning* model skenario 3

d. Model hasil *transfer learning* dihadapkan dengan dataset uji yang terdiri dari 200 citra dengan 20 identitas berbeda. Hasil perhitungan peforma model secara rinci dapat dilihat pada lampiran, sedangkan rangkuman peforma model menggunakan *confusion matrix* dapat dilihat pada tabel 4.9.

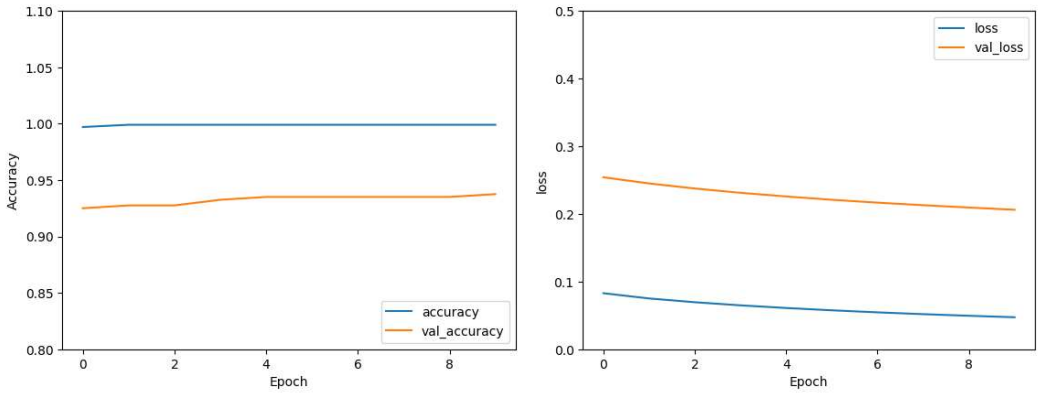
Tabel 4.9 Peforma model hasil *transfer learning* model skenario 3

Skenario	macro avg			weighted avg			akurasi
	precision	recall	f1-score	precision	recall	f1-score	
3	0,94	0,93	0,94	0,94	0,94	0,94	0,94

e. Dilakukan *fine and tuning* pada model hasil *transfer learning,* dikompile kembali dengan *learning rate* lebih kecil yaitu *base learning rate* dibagi dengan 10 dan dilatih dengan *epoch* 10 sehingga didapatkan nilai *learning rate* sebesar 0,0001. Dilakukan perubahan hanya pada beberapa *layer* terakhir. Terdapat total 167 *layer* pada model dan perubahan hanya dilakukan pada *layer* ke-132 hingga *layer* terakhir. Semua *layer batchnorm* pada model tetap dibekukan. *Fine and tuning* model hasil skenario 3 memiliki akurasi 93,75% dan nilai *loss* 0,2061 terhadap dataset validasi. Berikut adalah rincian tiap *epoch* beserta total lama waktu yang dibutuhkan untuk pelatihan.

```
Epoch 1/10
32/32 [==============================] - 8s 130ms/step - loss: 0.0828 - accuracy: 0.9970 - val_loss: 0.2541 - val_accuracy: 0.9250
Epoch 2/10
32/32 [==============================] - 3s 107ms/step - loss: 0.0750 - accuracy: 0.9990 - val_loss: 0.2450 - val_accuracy: 0.9275
Epoch 3/10
32/32 [==============================] - 3s 83ms/step - loss: 0.0694 - accuracy: 0.9990 - val_loss: 0.2375 - val_accuracy: 0.9275
Epoch 4/10
32/32 [==============================] - 3s 83ms/step - loss: 0.0648 - accuracy: 0.9990 - val_loss: 0.2311 - val_accuracy: 0.9325
Epoch 5/10
32/32 [==============================] - 3s 91ms/step - loss: 0.0609 - accuracy: 0.9990 - val_loss: 0.2257 - val_accuracy: 0.9350
Epoch 6/10
32/32 [==============================] - 3s 84ms/step - loss: 0.0575 - accuracy: 0.9990 - val_loss: 0.2208 - val_accuracy: 0.9350
Epoch 7/10
32/32 [==============================] - 3s 84ms/step - loss: 0.0545 - accuracy: 0.9990 - val_loss: 0.2166 - val_accuracy: 0.9350
Epoch 8/10
32/32 [==============================] - 3s 83ms/step - loss: 0.0518 - accuracy: 0.9990 - val_loss: 0.2128 - val_accuracy: 0.9350
Epoch 9/10
32/32 [==============================] - 3s 83ms/step - loss: 0.0494 - accuracy: 0.9990 - val_loss: 0.2093 - val_accuracy: 0.9350
Epoch 10/10
32/32 [==============================] - 3s 82ms/step - loss: 0.0473 - accuracy: 0.9990 - val_loss: 0.2061 - val_accuracy: 0.9375
CPU times: user 22.5 s, sys: 6.85 s, total: 29.3 s
Wall time: 43.1 s
```

Gambar 4.23 Rincian *epoch fine and tuning* skenario 3



Gambar 4.24 Grafik akurasi dan *loss* pada pelatihan *fine and tuning* model skenario 3

f. Model hasil *fine and tuning* dihadapkan dengan dataset uji yang terdiri dari 200 citra dengan 20 identitas berbeda. Hasil perhitungan peforma model hasil *fine and tuning* secara rinci dapat dilihat pada lampiran, sedangkan rangkuman peforma model menggunakan *confusion matrix* dapat dilihat pada tabel 4.10.

Tabel 4.10 Rangkuman peforma model hasil *fine and tuning* pada skenario 3

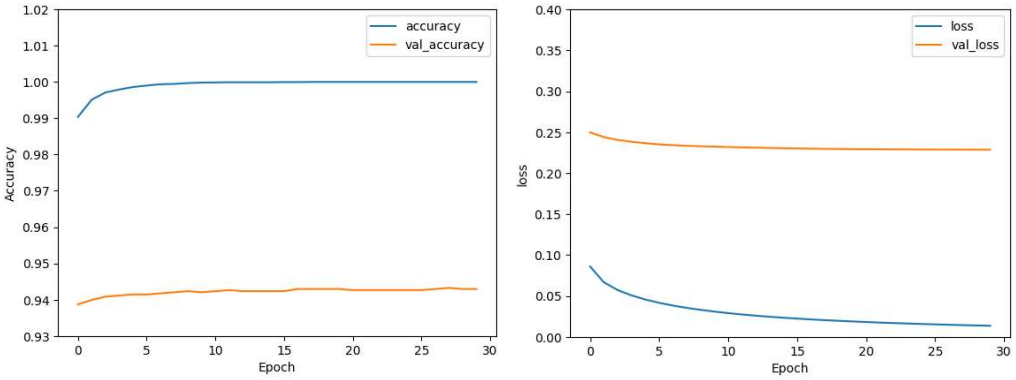
Skenario	macro avg			weighted avg			akurasi
	precision	recall	f1-score	precision	recall	f1-score	
3	0,95	0,95	0,95	0,95	0,95	0,95	0,95

4. Skenario 4 model dilatih dengan *learning rate* 0,01 dan 0,001; *epoch* 5 dan 30.

a. Pelatihan model dilakukan dengan dua tahap. Tahap pertama dilakukan pelatihan dengan *learning rate* 0,01 dan *epoch 5*. Pelatihan model tahap pertama telah dilakukan sebelumnya pada skenario 3 sehingga cukup melanjutkan ke tahap kedua. Pelatihan tahap kedua dilakukan dengan *learning rate* 0,001 dan *epoch* 30. Model yang dihasilkan memiliki tingkat akurasi 94,29% dan *loss* sebesar 0,2286 terhadap dataset validasi.

```
Epoch 26/30
688/688 [==============================] - 210s 304ms/step - loss: 0.0152 - accuracy: 1.0000 - val_loss: 0.2288 - val_accuracy: 0.9426
Epoch 27/30
688/688 [==============================] - 215s 312ms/step - loss: 0.0147 - accuracy: 1.0000 - val_loss: 0.2287 - val_accuracy: 0.9429
Epoch 28/30
688/688 [==============================] - 211s 306ms/step - loss: 0.0143 - accuracy: 1.0000 - val_loss: 0.2287 - val_accuracy: 0.9432
Epoch 29/30
688/688 [==============================] - 220s 318ms/step - loss: 0.0139 - accuracy: 1.0000 - val_loss: 0.2286 - val_accuracy: 0.9429
Epoch 30/30
688/688 [==============================] - 224s 324ms/step - loss: 0.0135 - accuracy: 1.0000 - val_loss: 0.2286 - val_accuracy: 0.9429
CPU times: user 35min 17s, sys: 7min 10s, total: 42min 28s
Wall time: 1h 47min 29s
```

Gambar 4.25 Lima *epoch* terakhir pelatihan tahap kedua pada skenario 4



Gambar 4.26 Grafik akurasi dan *loss* pada pelatihan tahap kedua dengan *learning rate* 0,001 dan *epoch* 30

b. Model hasil pelatihan dihadapkan dengan dataset uji yang terdiri dari 1088 citra dengan 50 identitas berbeda. Jumlah citra tiap identitas berbeda-beda. Hasil perhitungan peforma model secara rinci dapat dilihat pada lampiran, sedangkan rangkuman peforma model menggunakan *confusion matrix* dapat dilihat pada tabel 4.11.

Tabel 4.11 Peforma model hasil skenario 4

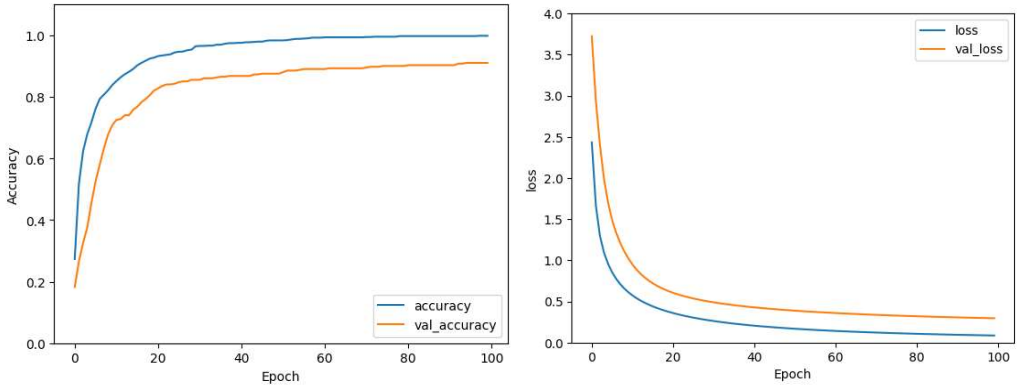
Skenario	macro avg			weighted avg			akurasi
	precision	recall	f1-score	precision	recall	f1-score	
4	0,94	0,94	0,94	0,94	0,94	0,94	0,94

c. Dilakukan *transfer learning* pada model yang telah dilatih, dikompile kembali dengan *learning rate* 0,08 dan dilatih dengan *epoch* 100 sehingga didapatkan hasil sebagai berikut. *Transfer learning* model skenario 4 menghasilkan

akurasi 91% dan nilai *loss* 0.2957 terhadap dataset validasi. Berikut adalah rincian 5 *epoch* terakhir beserta total lama waktu yang dibutuhkan untuk pelatihan.

```
Epoch 96/100
32/32 [==============================] - 2s 72ms/step - loss: 0.0895 - accuracy: 0.9970 - val_loss: 0.3000 - val_accuracy: 0.9100
Epoch 97/100
32/32 [==============================] - 3s 93ms/step - loss: 0.0885 - accuracy: 0.9970 - val_loss: 0.2989 - val_accuracy: 0.9100
Epoch 98/100
32/32 [==============================] - 3s 83ms/step - loss: 0.0876 - accuracy: 0.9980 - val_loss: 0.2978 - val_accuracy: 0.9100
Epoch 99/100
32/32 [==============================] - 2s 74ms/step - loss: 0.0867 - accuracy: 0.9980 - val_loss: 0.2967 - val_accuracy: 0.9100
Epoch 100/100
32/32 [==============================] - 2s 73ms/step - loss: 0.0858 - accuracy: 0.9980 - val_loss: 0.2957 - val_accuracy: 0.9100
CPU times: user 2min 27s, sys: 1min 7s, total: 3min 35s
Wall time: 5min 15s
```

Gambar 4.27 Lima *epoch* terakhir *transfer learning* pada skenario 4



Gambar 4.28 Grafik akurasi dan *loss* pada pelatihan *transfer learning* model skenario 4

d. Model hasil *transfer learning* dihadapkan dengan dataset uji yang terdiri dari 200 citra dengan 20 identitas berbeda. Hasil perhitungan peforma model secara rinci dapat dilihat pada lampiran, sedangkan rangkuman peforma model menggunakan *confusion matrix* dapat dilihat pada tabel 4.12.

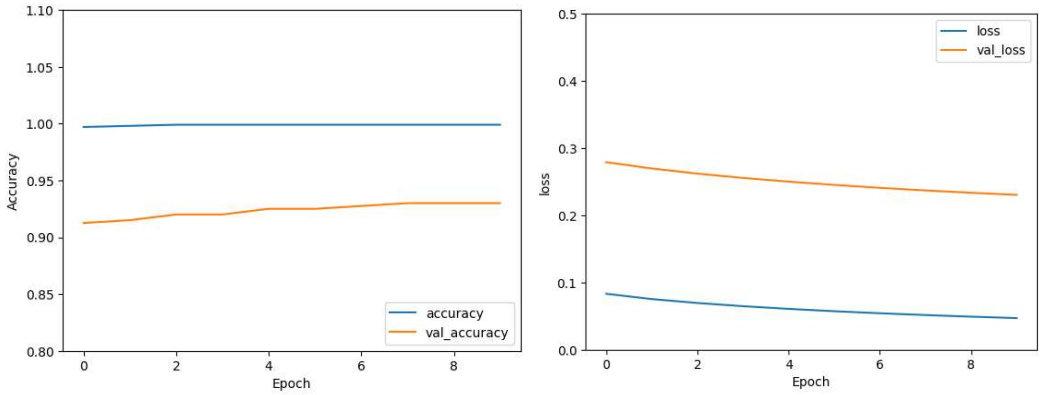
Tabel 4.12 Peforma model hasil *transfer learning* model skenario 4

Skenario	macro avg			weighted avg			akurasi
	precision	recall	f1-score	precision	recall	f1-score	
4	0,94	0,93	0,93	0,94	0,93	0,93	0,93

e. Dilakukan *fine and tuning* pada model hasil *transfer learning,* dikompile kembali dengan *learning rate* lebih kecil yaitu *base learning rate* dibagi dengan 10 dan dilatih dengan *epoch* 10 sehingga didapatkan nilai *learning rate* sebesar 0,0001. Dilakukan perubahan hanya pada beberapa *layer* terakhir. Terdapat total 167 *layer* pada model dan perubahan hanya dilakukan pada *layer* ke-132 hingga *layer* terakhir. Semua *layer batchnorm* pada model tetap dibekukan. *Fine and tuning* model hasil skenario 4 memiliki akurasi 93% dan nilai *loss* 0,2303 terhadap dataset validasi. Berikut adalah rincian tiap *epoch* beserta total lama waktu yang dibutuhkan untuk pelatihan.

```
Epoch 1/10
32/32 [==============================] - 39s 741ms/step - loss: 0.0831 - accuracy: 0.9970 - val_loss: 0.2790 - val_accuracy: 0.9125
Epoch 2/10
32/32 [==============================] - 3s 85ms/step - loss: 0.0752 - accuracy: 0.9980 - val_loss: 0.2696 - val_accuracy: 0.9150
Epoch 3/10
32/32 [==============================] - 5s 137ms/step - loss: 0.0694 - accuracy: 0.9990 - val_loss: 0.2619 - val_accuracy: 0.9200
Epoch 4/10
32/32 [==============================] - 3s 83ms/step - loss: 0.0646 - accuracy: 0.9990 - val_loss: 0.2555 - val_accuracy: 0.9200
Epoch 5/10
32/32 [==============================] - 3s 83ms/step - loss: 0.0606 - accuracy: 0.9990 - val_loss: 0.2500 - val_accuracy: 0.9250
Epoch 6/10
32/32 [==============================] - 3s 92ms/step - loss: 0.0571 - accuracy: 0.9990 - val_loss: 0.2452 - val_accuracy: 0.9250
Epoch 7/10
32/32 [==============================] - 3s 85ms/step - loss: 0.0541 - accuracy: 0.9990 - val_loss: 0.2408 - val_accuracy: 0.9275
Epoch 8/10
32/32 [==============================] - 3s 83ms/step - loss: 0.0514 - accuracy: 0.9990 - val_loss: 0.2369 - val_accuracy: 0.9300
Epoch 9/10
32/32 [==============================] - 3s 81ms/step - loss: 0.0490 - accuracy: 0.9990 - val_loss: 0.2335 - val_accuracy: 0.9300
Epoch 10/10
32/32 [==============================] - 3s 84ms/step - loss: 0.0469 - accuracy: 0.9990 - val_loss: 0.2303 - val_accuracy: 0.9300
CPU times: user 26.2 s, sys: 7.4 s, total: 33.6 s
Wall time: 1min 18s
```

Gambar 4.29 Rincian *epoch fine and tuning* pada skenario 4



Gambar 4.30 Grafik akurasi dan *loss* pada pelatihan *fine and tuning* model skenario 4

f. Model hasil *fine and tuning* dihadapkan dengan dataset uji yang terdiri dari 200 citra dengan 20 identitas berbeda. Hasil perhitungan peforma model hasil *fine and tuning* secara rinci dapat dilihat pada lampiran, sedangkan rangkuman peforma model menggunakan *confusion matrix* dapat dilihat pada tabel 4.13.

Tabel 4.13 Rangkuman peforma model hasil *fine and tuning* pada skenario 4

Skenario	macro avg			weighted avg			akurasi
	precision	recall	f1-score	precision	recall	f1-score	
4	0,95	0,95	0,95	0,95	0,95	0,95	0,95

Berdasarkan hasil uji coba pelatihan dengan *learning rate* dan *epoch* sesuai skenario uji coba, didapatkan nilai akurasi, *loss* dan lama waktu pelatihan yang berbeda-beda. Untuk memudahkan dalam membaca, hasil pelatihan tersebut dirangkum dalam tabel 4.14.

Tabel 4.14 Rangkuman hasil pelatihan model sesuai skenario

Skenario	*Learning rate*	*Epoch*	Total waktu pelatihan	Akurasi	Nilai *loss*
1	0,001	100	6 jam 12 menit	88,10%	0,4522
2	0,002	100	5 jam 46 menit	89,52%	0,3890
3	0,01 dan 0,001	5 dan 8	54 menit	94,23%	0,2334
4	0,01 dan 0,001	5 dan 30	2 jam 11 menit	94,29%	0,2286

Untuk memudahkan dalam memahami *transfer learning* dan *fine and tuning* yang dilakukan pada model, maka ditunjukkan visualisasi model dalam bentuk tabel. Visualisasi *transfer learning* model dapat dilihat pada tabel 4.15. Terdapat total 1,9 juta parameter dan hanya terdapat 25 ribu parameter yang dapat dilatih yaitu pada *layer sequential* yang memiliki *output* sejumlah 20. Visualisasi *fine and tuning* model dapat dilihat pada tabel 4.16. Terdapat total 1,9 juta parameter dan 1,4 juta parameter di antaranya dapat dilatih.

Tabel 4.15 Visualisasi *transfer learning* model

Layer (type)	Output Shape	Param#
Model (Functional)	(None, 1280)	1963584
Sequential (Sequential)	(None, 20)	25629
Total params: 1,989,204 Trainable params: 25,620 Non-trainable params: 1,963,584		

Tabel 3.16 Visualisasi *fine and tuning* model

Layer (type)	Output Shape	Param#
Model (Functional)	(None, 1280)	1963584
Sequential (Sequential)	(None, 20)	25629
Total params: 1,989,204 Trainable params: 1,434,164 Non-trainable params: 555,040		

Rangkuman pengukuran peforma model pada seluruh skenario dapat dilihat pada tabel 4.17.

Tabel 4.17 Rangkuman peforma model pada seluruh skenario

Keterangan	macro avg			weighted avg			akurasi
	precision	recall	f1-score	precision	recall	f1-score	
Skenario 1	0,88	0,88	0,88	0,89	0,88	0,88	0,88
Transfer learning skenario 1	0,88	0,87	0,87	0,88	0,87	0,87	0,87
Fine and tuning skenario 1	0,88	0,87	0,87	0,88	0,87	0,87	0,87
Skenario 2	0,90	0,90	0,89	0,90	0,90	0,90	0,90
Transfer learning skenario 2	0,89	0,88	0,88	0,89	0,88	0,88	0,88
Fine and tuning skenario 2	0,91	0,90	0,90	0,91	0,91	0,90	0,91

Lanjutan

Keterangan	macro avg			weighted avg			akurasi
	precision	recall	f1-score	precision	recall	f1-score	
Skenario 3	0,94	0,94	0,94	0,95	0,94	0,94	0,94
Transfer learning skenario 3	0,94	0,93	0,94	0,94	0,94	0,94	0,94
Fine and tuning skenario 3	0,95	0,95	0,95	0,95	0,95	0,95	0,95
Skenario 4	0,94	0,94	0,94	0,94	0,94	0,94	0,94
Transfer learning skenario 4	0,94	0,93	0,93	0,94	0,93	0,93	0,93
Fine and tuning skenario 4	0,95	0,95	0,95	0,95	0,95	0,95	0,95

4.2 Pembahasan

Pada skenario 1 model dilatih dengan *learning rate* 0,001 dan *epoch* 100 dan dibutuhkan waktu 6 jam 12 menit untuk pelatihan. Model hasil skenario 1 memiliki nilai *weighted average precision, recall, f1-score* dan akurasi sebesar 0,88. Ketika dilakukan *transfer learning*, model memiliki nilai *precision* sebesar 0,88 dan nilai *recall, f1-score* serta akurasi sebesar 0,87. *Fine and tuning* pada model tidak memberikan perubahan yang signifikan, model tetap memiliki nilai *precision, recall, f1-score* dan akurasi yang sama pada hasil *transfer learning*.

Pada skenario 2 model dilatih dengan *learning rate* 0,002 dan *epoch* 100, dan dibutuhkan waktu 5 jam 46 menit untuk pelatihan. Model hasil skenario 2 memiliki nilai *weighted average precision, recall, f1-score* dan akurasi sebesar 0,9. *Transfer learning* model menghasilkan nilai *precision* sebesar 0,89 dan nilai *recall, f1-score* serta akurasi sebesar 0,88. *Fine and tuning* pada model mampu memberikan perubahan yang cukup signifikan, model mengalami peningkatan nilai *precision, recall* dan akurasi menjadi 0,91 dan nilai *f1-score* menjadi 0.90.

Pada skenario 3 model melalui dua tahap pelatihan dengan *learning rate* berbeda. Pelatihan tahap pertama dilakukan dengan *learning rate* 0,01 dan *epoch* 5. Pelatihan tahap kedua dilakukan dengan *learning rate* 0,001 dan *epoch* 8. Lama waktu pelatihan model adalah 54 menit. Model memiliki nilai *weighted average precision* sebesar 0,95 dan nilai *weighted average recall, f1-score* serta akurasi sebesar 0,94. *Transfer learning* pada model menghasilkan nilai *precision, recall, f1-score* dan akurasi sebesar 0,94. *Fine and tuning* pada model mampu meningkatkan peforma model, model mengalami peningkatan nilai *precision, recall, f1-score* dan akurasi menjadi 0,95.

Pada skenario 4 model melalui dua tahap pelatihan dengan *learning rate* berbeda. Pelatihan tahap pertama dilakukan dengan *learning rate* 0,01 dan *epoch* 5. Pelatihan tahap kedua dilakukan dengan *learning rate* 0,001 dan *epoch* 30. Dibutuhkan total waktu 2 jam 11 menit untuk pelatihan. Model memiliki nilai *weighted average precision, recall, f1-score* dan akurasi sebesar 0,94. *Transfer learning* pada model menghasilkan nilai *precision* sebesar 0,94 dan nilai *recall, f1-score* serta akurasi sebesar 0,93. *Fine and tuning* pada model mampu meningkatkan peforma model, model mengalami peningkatan nilai *precision, recall, f1-score* dan akurasi menjadi 0,95.

Berdasarkan hasil pengukuran peforma model dapat diketahui bahwa model hasil skenario 3 memiliki peforma terbaik. Model hasil skenario 3 memiliki skor *confusion matrix* paling tinggi pada tahap membangun model dan implementasi model, selain itu model hasil skenario 3 memiliki waktu pelatihan paling singkat yaitu 54 menit, sedangkan model hasil skenario lainnya memiliki skor *confusion*

matrix yang yang lebih rendah atau sama namun membutuhkan waktu pelatihan lebih lama yaitu dibutuhkan waktu dua hingga enam jam.

Hasil penelitian menunjukkan bahwa pelatihan yang dilakukan secara bertahap dengan *learning rate* berbeda yaitu dari *learning rate* besar ke *learning rate* lebih kecil mampu menghasilkan model dengan peforma lebih baik dalam waktu yang lebih singkat. Pelatihan yang dilakukan dengan *learning rate* kecil dan *epoch* besar menghasilkan model dengan peforma lebih buruk dan sangat memakan banyak waktu dan memori. Model yang dilatih dengan *learning rate* kecil dan *epoch* besar memiliki tendensi untuk terjadi *overfitting. Overfitting* mungkin terjadi dikarenakan model terlalu sering melihat data sehingga yang terjadi adalah model mengingat atau menghafal data bukannya mempelajari data. Pelatihan secara bertahap dengan *learning rate* berbeda yaitu dari *learning rate* besar ke *learning rate* lebih kecil dapat dilakukan sebagai alternatif pilihan untuk mengatasi resiko *overfitting* dan untuk menghemat memori dan lama waktu pelatihan.

Identitas dari dataset yang digunakan untuk membangun model adalah ras *asian* khususnya berasal dari negara Indonesia. Sedangkan dataset yang digunakan pada *transfer learning* dan *fine and tuning* didapatkan dari caltech adalah mayoritas dari ras *caucasian* khususnya berasal dari negara Inggris. Ras *asian* dan *caucasian* memiliki beberapa karakteristik berbeda yang mencolok seperti warna kulit, bentuk hidung dan mata namun model yang dihasilkan dapat berfungsi dengan baik terhadap kedua dataset tersebut. Selain ras, jumlah identitas pada dataset yang digunakan untuk membangun dan menguji model juga berbeda. Ini dapat menunjukkan bahwa model dengan arsitektur MobileNetV2 dapat digunakan

sebagai pengenalan wajah yang fleksibel terhadap perbedaan mencolok atau perubahan pada data. Dapat dilakukan *fine and tuning* pada model yang sudah dibangun untuk mengatasi masalah-masalah tersebut.

Setiap manusia memiliki karakteristik yang berbeda antara satu dengan lainnya sehingga mudah untuk mengenali tiap-tiap orang, hal ini sesuai dengan Alqur'an surat Al-Hujurat ayat 13.

يَٰٓأَيُّهَا ٱلنَّاسُ إِنَّا خَلَقْنَٰكُم مِّن ذَكَرٍ وَأُنثَىٰ وَجَعَلْنَٰكُمْ شُعُوبًا وَقَبَآئِلَ لِتَعَارَفُوٓاْ ۚ إِنَّ أَكْرَمَكُمْ عِندَ ٱللَّهِ
أَتْقَىٰكُمْ ۚ إِنَّ ٱللَّهَ عَلِيمٌ خَبِيرٌ

"Hai manusia, sesungguhnya Kami menciptakan kamu dari seorang laki-laki dan seorang perempuan dan menjadikan kamu berbangsa-bangsa dan bersuku-suku supaya kamu saling kenal-mengenal. Sesungguhnya orang yang paling mulia di antara kamu di sisi Allah ialah orang yang paling takwa di antara kamu. Sesungguhnya Allah Maha Mengetahui lagi Maha Mengenal."(QS. Al Hujurat:13)

Tafsir Al-Hujurat ayat 13 menurut Al-Muyassar atau Kementerian Agama Saudi Arabia adalah sebagai berikut.

"Wahai manusia, sesungguhnya Kami menciptakan kalian dari satu bapak, yaitu Adam dan satu ibu yaitu Hawa. Maka janganlah merasa lebih utama di antara sebagian kalian atas sebagian yang lain dari sisi nasab. Kami menjadikan kalian berbangsa-bangsa dan bersuku-suku melalui proses berketurunan, agar sebagian dari kalian mengenal sebagian yang lain. Sesungguhnya orang yang paling mulia di antara kalian di sisi Allah adalah yang paling bertakwa kepada Allah. Sesungguhnya Allah Maha Mengetahui orang-orang yang bertakwa dan Maha teliti terhadap mereka."

Setiap ras atau suku bangsa memiliki karakteristik yang berbeda satu dengan lainnya seperti misalnya kulit putih pada ras *caucasian,* kelopak mata tidak

memiliki lipatan pada orang-orang asia timur, dan sebagainya. Karakteristik-karekteristik tersebut diturunkan secara genetik dari ayah dan ibu. Tujuan diciptakannya perbedaan karakteristik pada tiap manusia bukan untuk menjadi penyebab pertikaian, melainkan untuk manusia saling mengenali satu dengan lainnya. Karekteristik unik pada tiap manusia tidak hanya dapat dipelajari oleh manusia, namun juga dapat dipelajari oleh model MobileNetV2. Model MobilenetV2 dapat mempelajari karakteristik unik tiap manusia kemudian mengingatnya, menyimpannya dalam bentuk parameter.

Baik atau buruk sebuah model bergantung pada proses pelatihan dimana model dihadapkan banyak data untuk dipelajari. Ketika model baru dibuat, parameter masih bernilai *default* kemudian dengan mempelajari berbagai data, perlahan-lahan nilai parameter tersebut diperbarui hingga model dapat menjalankan fungsi tertentu. Proses belajar pada model mirip dengan proses belajar manusia. Ketika manusia baru dilahirkan, manusia belum bisa berbicara, dengan terus-menerus mendengarkan orang-orang berbicara, perlahan-lahan dapat memahami topik yang dibicarakan kemudian mencoba untuk berbicara. Hal ini sesuai dengan Alqur'an surat An-Nahl ayat 78.

وَاللَّهُ أَخْرَجَكُمْ مِنْ بُطُونِ أُمَّهَاتِكُمْ لَا تَعْلَمُونَ شَيْئًا وَجَعَلَ لَكُمُ السَّمْعَ وَالْأَبْصَارَ وَالْأَفْئِدَةَ ۙ لَعَلَّكُمْ تَشْكُرُونَ

"Dan Allah mengeluarkan kamu dari perut ibumu dalam keadaan tidak mengetahui sesuatupun, dan Dia memberi kamu pendengaran, penglihatan dan hati, agar kamu bersyukur." (QS. An nahl 16:78)

Tafsir An-Nahl ayat 78 menurut Fathul Qadir adalah sebagai berikut.

وَاللَّهُ أَخْرَجَكُمْ مِنْ بُطُونِ أُمَّهَاتِكُمْ لَا تَعْلَمُونَ شَيْئًا

"Dan Allah mengeluarkan kamu dari perut ibumu dalam keadaan tidak mengetahui sesuatupun" (QS. An nahl 16:78)

Maksudnya, mengeluarkanmu dari perut ibumu sebagai bayi yang tidak mengetahui apapun.

وَجَعَلَ لَكُمُ ٱلسَّمْعَ وَٱلْأَبْصَٰرَ وَٱلْأَفْـِٔدَةَ ۙ

"dan Dia memberi kamu pendengaran, penglihatan dan hati" (QS. An nahl 16:78)

Maksudnya adalah Allah memberikan hal-hal tersebut kepadamu.

لَعَلَّكُم تَشْكُرُونَ

"agar kamu bersyukur" (QS. An nahl 16:78)

Maksudnya adalah agar kamu menggunakan semua alat tubuh itu sesuai dengan untuk apa itu diciptakan. Jadi, saat itulah kamu mengetahui kadar apa-apa yang Allah anugerahkan kepadamu sehingga kamu mensyukurinya.

An-Nahl ayat 78 mengajarkan betapa pentingnya belajar mendapatkan ilmu. Sarana belajar juga bermacam-macam seperti pendengaran, pengelihatan dan hati. Demikian juga pada model , data yang digunakan bisa bermacam-macam seperti citra, teks, suara, dan lain-lain. Dengan belajar, manusia dapat mengetahui nilai kenikmatan yang Allah. Dengan mempelajari perbedaan-perbedaan karakteristik pada manusia, manusia dapat mengetahui keagungan Allah dalam menciptakan makhluknya hingga dari milyaran manusia, tidak ada seorang manusiapun yang memiliki karakteristik yang sama persis.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil pengukuran peforma model menggunakan *confusion matrix*, diperoleh kesimpulan bahwa MobileNetV2 memiliki peforma yang baik untuk pengenalan wajah.

1) Pada tahap membangun model, didapati model MobileNetV2 terbaik yaitu model hasil skenario 3 yang dibangun melalui dua tahap dengan *learning rate* berbeda yaitu dari *learning rate* yang lebih besar ke *learning rate* lebih kecil. Pelatihan tahap pertama dilakukan dengan *learning rate* 0,01 dan pelatihan tahap kedua dilakukan dengan *learning rate* 0,001. Model memiliki nilai *weighted average precision* sebesar 0,95 dan nilai *weighted average recall, f1-score* serta akurasi sebesar 0,94.
2) Pada tahap implementasi model *Transfer learning,* didapati model MobileNetV2 hasil skenario 3 memiliki nilai *precision, recall, f1-score* dan akurasi sebesar 0,94. *Fine and tuning* pada model mampu meningkatkan peforma model, model mengalami peningkatan nilai *precision, recall, f1-score* dan akurasi menjadi 0,95.

5.2 Saran

Dalam melatih MobileNetV2 untuk pengenalan wajah perlu diperhatikan beberapa hal yaitu:

1) Pelatihan dengan *learning rate* terlalu kecil dan *epoch* terlalu banyak sebaiknya dihindari karena dapat terjadi overfitting yaitu model mengingat dataset, bukannya mempelajari dataset. Selain itu pelatihan terjadi sangat lambat menghabiskan banyak waktu dan memori.
2) Untuk dapat membangun model dengan peforma baik dalam waktu lebih singkat, dapat digunakan pelatihan secara bertahap dengan *learning rate* berbeda yaitu dari *learning rate lebih* besar *learning rate* lebih kecil.
3) Pada penelitian ini digunakan *confussion matrix* untuk mengukur peforma model dan dihasilkan peforma model yang baik. Namun terdapat kriteria-kriteria lain yang tidak terliput pada *confussion matrix* seperti nilai *loss,* lama pelatihan dan beban komputasi.

DAFTAR PUSTAKA

Astawaa, I dkk. 2020. *KomNET: Face Image Dataset from Various Media for Face Recognition.* Data in Brief 31.

Asy-Syaukani, Imam. Tafsir Fathul Qadir Tahqiq dan Takhrij: Sayyid Ibrahim.

Bhattacharya, Shubhobrata et al. 2018. *Smart Attendance Monitoring System (SAMS): A Face Recognition based Attendance System for Classroom Environment*. IEEE 18th International Conference on Advanced Learning Technologies.

Damale, Radhika C. dan Pathak, Bageshree.V. 2018. *Face Recognition Based Attendance System Using Machine Learning Algorithms*. Proceedings of the Second International Conference on Intelligent Computing and Control Systems.

Delbiaggio, Nicolas. 2017. *A comparison of facial recognition's algorithms*. Haaga Helia University of Applied Sciences.

Dhillon, Anamika dan Verma, Gyanendra K. 2019. *Convolutional neural network: a review of models, methodologies and applications to object detection*. Springer: Progress in Artificial Intelligence

Howard, A dkk. 2017. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.* arXiv:1704.04861

Jayant, Nazare Kanchan dan Borra, Surekha. 2016 *Attendance Management System Using Hybrid Face Recognition Techniques.* Conference on Advances in Signal Processing.

Kurniawan, Vincentius dkk. 2017. *The Implementation of Eigenface Algorithm for Face Recognition in Attendance System*. International Conference on New Media Studies.

Li, Stan Z. dan Jain, Anil K. 2011. Handbook of Face RecognitionSecond Edition. Springer.

Lukas, Samuel dkk. 2016. S*tudent Attendance System in Classroom Using Face Recognition Technique*. International Conference on Information and Communication Technology Convergence.

Sammut, Claude dan Webb, Geoffrey I. 2017. *Encyclopediaof Machine Learningand Data MiningSecond Edition.* Sydney: Springer.

Sandler, Mark dkk. 2019. *MobileNetV2: Inverted Residuals and Linear Bottlenecks.* arXiv:1801.04381

Sanli, Onur dan Ilgen, Bahar. 2018. *Face Detection and Recognition for Automatic Attendance System*. Springer.

Suyanto. 2018. *Machine Learning Tingkat Dasar dan Lanjut*. Bandung: INFORMATIKA.

Viola, Paul dan Jones, Michael. 2001. *Robust Real-time Object Detection*. Vancouver: Second International Workshop On Statistical And Computational Theories Of Vision – Modeling, Learning, Computing, And Sampling

LAMPIRAN

Tabel peforma model skenario 1 menggunakan *confusion matrix*

label	precision	recall	f1-score	support
0	1.00	1.00	1.00	20
1	0.89	1.00	0.94	17
2	1.00	0.94	0.97	18
3	0.84	1.00	0.91	16
4	0.95	1.00	0.98	20
5	0.87	1.00	0.93	20
6	0.93	0.93	0.93	41
7	0.92	0.88	0.90	26
8	0.93	1.00	0.96	27
9	0.96	0.96	0.96	26
10	0.95	0.95	0.95	21
11	0.92	0.92	0.92	24
12	0.77	0.87	0.82	23
13	0.95	0.91	0.93	23
14	0.86	1.00	0.93	19
15	0.83	1.00	0.91	20
16	0.90	0.93	0.92	30
17	1.00	0.95	0.98	22
18	1.00	0.88	0.94	17
19	1.00	1.00	1.00	21
20	0.88	0.75	0.81	20
21	1.00	1.00	1.00	15
22	0.92	0.96	0.94	23
23	1.00	1.00	1.00	16
24	1.00	0.84	0.91	25
25	0.84	0.78	0.81	27
26	0.57	0.38	0.46	21

Lanjutan tabel peforma model skenario 1 menggunakan *confusion matrix*

label	precision	recall	f1-score	support
27	0.80	0.92	0.86	26
28	0.88	0.84	0.86	25
29	0.40	0.62	0.48	13
30	0.62	0.67	0.65	15
31	0.78	0.74	0.76	19
32	0.88	0.83	0.86	18
33	0.78	0.82	0.80	22
34	0.88	0.54	0.67	26
35	0.74	0.82	0.78	17
36	0.78	1.00	0.88	25
37	0.83	0.79	0.81	24
38	0.94	0.94	0.94	17
39	0.83	0.70	0.76	27
40	0.89	0.89	0.89	19
41	0.89	0.81	0.85	21
42	0.96	1.00	0.98	25
43	0.95	0.95	0.95	21
44	0.92	0.96	0.94	23
45	0.92	0.65	0.76	17
46	0.96	0.88	0.92	26
47	1.00	1.00	1.00	16
48	0.83	0.86	0.84	22
49	0.96	1.00	0.98	26
accuracy			0.88	1088
macro avg	0.88	0.88	0.88	1088
weighted avg	0.89	0.88	0.88	1088

Tabel peforma model skenario 2 menggunakan *confusion matrix*

label	precision	recall	f1-score	support
0	1.00	1.00	1.00	20
1	1.00	1.00	1.00	17
2	0.86	1.00	0.92	18
3	0.80	1.00	0.89	16
4	0.95	0.90	0.92	20
5	0.87	1.00	0.93	20
6	1.00	0.98	0.99	41
7	0.89	0.92	0.91	26
8	0.96	1.00	0.98	27
9	0.93	1.00	0.96	26
10	0.95	1.00	0.98	21
11	0.96	0.96	0.96	24
12	0.85	0.96	0.90	23
13	0.79	1.00	0.88	23
14	0.86	1.00	0.93	19
15	0.95	1.00	0.98	20
16	1.00	0.83	0.91	30
17	0.87	0.91	0.89	22
18	0.82	0.82	0.82	17
19	0.91	0.95	0.93	21
20	1.00	0.90	0.95	20
21	0.88	1.00	0.94	15
22	0.96	0.96	0.96	23
23	1.00	1.00	1.00	16
24	1.00	0.72	0.84	25
25	0.89	0.93	0.91	27
26	0.67	0.48	0.56	21
27	1.00	0.92	0.96	26
28	0.83	0.76	0.79	25
29	0.71	0.77	0.74	13

Lanjutan tabel peforma model skenario 2 menggunakan *confusion matrix*

label	precision	recall	f1-score	support
30	0.57	0.53	0.55	15
31	0.94	0.79	0.86	19
32	0.89	0.94	0.92	18
33	0.94	0.77	0.85	22
34	0.77	0.77	0.77	26
35	0.78	0.82	0.80	17
36	0.85	0.92	0.88	25
37	0.86	0.79	0.83	24
38	0.89	1.00	0.94	17
39	0.90	0.96	0.93	27
40	0.95	0.95	0.95	19
41	0.83	0.95	0.89	21
42	1.00	1.00	1.00	25
43	0.89	0.76	0.82	21
44	0.96	0.96	0.96	23
45	0.86	0.71	0.77	17
46	1.00	0.77	0.87	26
47	1.00	1.00	1.00	16
48	0.87	0.91	0.89	22
49	0.86	0.92	0.89	26
accuracy			0.90	1088
macro avg	0.90	0.90	0.89	1088
weighted avg	0.90	0.90	0.90	1088

Tabel peforma model skenario 3 menggunakan *confusion matrix*

label	precision	recall	f1-score	support
0	0.95	1.00	0.98	20
1	1.00	1.00	1.00	17
2	1.00	1.00	1.00	18
3	1.00	1.00	1.00	16
4	1.00	1.00	1.00	20
5	0.95	1.00	0.98	20
6	1.00	1.00	1.00	41
7	0.90	1.00	0.95	26
8	0.96	1.00	0.98	27
9	1.00	0.96	0.98	26
10	1.00	0.90	0.95	21
11	1.00	0.92	0.96	24
12	1.00	0.96	0.98	23
13	1.00	1.00	1.00	23
14	0.95	1.00	0.97	19
15	1.00	1.00	1.00	20
16	0.94	0.97	0.95	30
17	1.00	1.00	1.00	22
18	0.94	0.94	0.94	17
19	1.00	1.00	1.00	21
20	0.95	0.90	0.92	20
21	1.00	1.00	1.00	15
22	0.96	1.00	0.98	23
23	1.00	1.00	1.00	16
24	1.00	0.80	0.89	25
25	0.89	0.89	0.89	27
26	0.56	0.48	0.51	21
27	0.93	0.96	0.94	26
28	1.00	0.88	0.94	25
29	0.61	0.85	0.71	13

Lanjutan tabel peforma model skenario 3 menggunakan *confusion matrix*

label	precision	recall	f1-score	support
30	0.87	0.87	0.87	15
31	0.64	0.84	0.73	19
32	1.00	0.94	0.97	18
33	1.00	1.00	1.00	22
34	0.91	0.81	0.86	26
35	0.94	0.94	0.94	17
36	0.89	0.96	0.92	25
37	0.96	0.96	0.96	24
38	0.85	1.00	0.92	17
39	0.89	0.89	0.89	27
40	0.90	0.95	0.92	19
41	1.00	0.95	0.98	21
42	0.96	1.00	0.98	25
43	0.95	0.90	0.93	21
44	0.96	0.96	0.96	23
45	1.00	0.76	0.87	17
46	0.96	1.00	0.98	26
47	1.00	1.00	1.00	16
48	1.00	1.00	1.00	22
49	0.96	0.92	0.94	26
accuracy			0.94	1088
macro avg	0.94	0.94	0.94	1088
weighted avg	0.95	0.94	0.94	1088

Tabel peforma model skenario 4 menggunakan *confusion matrix*

label	precision	recall	f1-score	support
0	1.00	1.00	1.00	20
1	1.00	1.00	1.00	17
2	1.00	1.00	1.00	18
3	1.00	1.00	1.00	16
4	1.00	1.00	1.00	20
5	0.95	1.00	0.98	20
6	1.00	1.00	1.00	41
7	0.90	1.00	0.95	26
8	1.00	1.00	1.00	27
9	1.00	1.00	1.00	26
10	1.00	0.90	0.95	21
11	1.00	0.96	0.98	24
12	1.00	0.96	0.98	23
13	1.00	1.00	1.00	23
14	0.95	1.00	0.97	19
15	1.00	1.00	1.00	20
16	0.94	0.97	0.95	30
17	1.00	1.00	1.00	22
18	1.00	0.94	0.97	17
19	1.00	0.95	0.98	21
20	0.95	0.90	0.92	20
21	1.00	1.00	1.00	15
22	0.96	1.00	0.98	23
23	0.94	1.00	0.97	16
24	0.95	0.80	0.87	25
25	0.86	0.89	0.87	27
26	0.56	0.48	0.51	21
27	0.93	0.96	0.94	26
28	0.96	0.88	0.92	25
29	0.65	0.85	0.73	13

Lanjutan tabel peforma model skenario 4 menggunakan *confusion matrix*

label	precision	recall	f1-score	support
30	0.86	0.80	0.83	15
31	0.64	0.84	0.73	19
32	1.00	0.94	0.97	18
33	1.00	1.00	1.00	22
34	0.91	0.81	0.86	26
35	0.94	0.94	0.94	17
36	0.89	0.96	0.92	25
37	0.96	0.96	0.96	24
38	0.85	1.00	0.92	17
39	0.89	0.89	0.89	27
40	0.86	0.95	0.90	19
41	1.00	0.95	0.98	21
42	0.96	1.00	0.98	25
43	0.95	0.90	0.93	21
44	0.96	0.96	0.96	23
45	1.00	0.71	0.83	17
46	0.96	1.00	0.98	26
47	1.00	1.00	1.00	16
48	1.00	1.00	1.00	22
49	0.96	0.92	0.94	26
accuracy			0.94	1088
macro avg	0.94	0.94	0.94	1088
weighted avg	0.94	0.94	0.94	1088

Tabel peforma *transfer learning* model hasil skenario 1 menggunakan *confusion matrix*

label	precision	recall	f1-score	support
0	1.00	0.90	0.95	10
1	1.00	0.90	0.95	10
2	0.90	0.90	0.90	10
3	0.83	1.00	0.91	10
4	0.86	0.60	0.71	10
5	0.91	1.00	0.95	10
6	0.70	0.70	0.70	10
7	0.75	0.90	0.82	10
8	0.91	1.00	0.95	10
9	0.91	1.00	0.95	10
10	0.80	0.80	0.80	10
11	0.67	0.60	0.63	10
12	1.00	1.00	1.00	10
13	0.82	0.90	0.86	10
14	1.00	0.80	0.89	10
15	0.88	0.70	0.78	10
16	1.00	0.70	0.82	10
17	1.00	1.00	1.00	10
18	0.77	1.00	0.87	10
19	0.83	1.00	0.91	10
accuracy			0.87	200
macro avg	0.88	0.87	0.87	200
weighted avg	0.88	0.87	0.87	200

Tabel peforma *transfer learning* model hasil skenario 2 menggunakan *confusion matrix*

label	precision	recall	f1-score	support
0	1.00	0.90	0.95	10
1	0.75	0.90	0.82	10
2	1.00	1.00	1.00	10
3	0.82	0.90	0.86	10
4	0.86	0.60	0.71	10
5	0.88	0.70	0.78	10
6	1.00	1.00	1.00	10
7	0.89	0.80	0.84	10
8	0.71	1.00	0.83	10
9	1.00	0.90	0.95	10
10	1.00	0.50	0.67	10
11	0.90	0.90	0.90	10
12	1.00	1.00	1.00	10
13	0.89	0.80	0.84	10
14	0.83	1.00	0.91	10
15	0.90	0.90	0.90	10
16	0.83	1.00	0.91	10
17	0.89	0.80	0.84	10
18	0.91	1.00	0.95	10
19	0.77	1.00	0.87	10
accuracy			0.88	200
macro avg	0.89	0.88	0.88	200
weighted avg	0.89	0.88	0.88	200

Tabel peforma *transfer learning* model hasil skenario 3 menggunakan *confusion matrix*

label	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	0.91	1.00	0.95	10
2	0.67	0.80	0.73	10
3	0.90	0.90	0.90	10
4	1.00	1.00	1.00	10
5	0.91	1.00	0.95	10
6	1.00	1.00	1.00	10
7	0.91	1.00	0.95	10
8	1.00	1.00	1.00	10
9	1.00	0.90	0.95	10
10	0.89	0.80	0.84	10
11	0.82	0.90	0.86	10
12	1.00	1.00	1.00	10
13	1.00	0.80	0.89	10
14	1.00	1.00	1.00	10
15	1.00	0.80	0.89	10
16	1.00	0.90	0.95	10
17	0.91	1.00	0.95	10
18	1.00	1.00	1.00	10
19	0.90	0.90	0.90	10
accuracy			0.94	200
macro avg	0.94	0.93	0.94	200
weighted avg	0.94	0.94	0.94	200

Tabel peforma *transfer learning* model hasil skenario 4 menggunakan *confusion matrix*

label	precision	recall	f1-score	support
0	1.00	0.90	0.95	10
1	0.91	1.00	0.95	10
2	0.73	0.80	0.76	10
3	0.90	0.90	0.90	10
4	1.00	1.00	1.00	10
5	0.83	1.00	0.91	10
6	1.00	0.90	0.95	10
7	0.91	1.00	0.95	10
8	1.00	1.00	1.00	10
9	0.90	0.90	0.90	10
10	0.80	0.80	0.80	10
11	0.82	0.90	0.86	10
12	1.00	1.00	1.00	10
13	1.00	0.80	0.89	10
14	1.00	1.00	1.00	10
15	1.00	0.80	0.89	10
16	1.00	1.00	1.00	10
17	0.91	1.00	0.95	10
18	1.00	1.00	1.00	10
19	1.00	0.90	0.95	10
accuracy			0.93	200
macro avg	0.94	0.93	0.93	200
weighted avg	0.94	0.93	0.93	200

Tabel peforma *fine and tuning* model hasil skenario 1 menggunakan *confusion matrix*

label	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	0.90	0.95	10
2	0.82	0.90	0.86	10
3	0.91	1.00	0.95	10
4	0.70	0.70	0.70	10
5	0.90	0.90	0.90	10
6	0.88	0.70	0.78	10
7	0.75	0.90	0.82	10
8	0.91	1.00	0.95	10
9	0.91	1.00	0.95	10
10	0.91	1.00	0.95	10
11	0.64	0.70	0.67	10
12	1.00	0.90	0.95	10
13	0.75	0.60	0.67	10
14	1.00	0.80	0.89	10
15	0.88	0.70	0.78	10
16	1.00	0.70	0.82	10
17	1.00	1.00	1.00	10
18	0.71	1.00	0.83	10
19	0.91	1.00	0.95	10
accuracy			0.87	200
macro avg	0.88	0.87	0.87	200
weighted avg	0.88	0.87	0.87	200

Tabel peforma *fine and tuning* model hasil skenario 2 menggunakan *confusion matrix*

label	precision	recall	f1-score	support
0	0.83	1.00	0.91	10
1	0.80	0.80	0.80	10
2	1.00	1.00	1.00	10
3	0.82	0.90	0.86	10
4	1.00	0.90	0.95	10
5	0.89	0.80	0.84	10
6	1.00	1.00	1.00	10
7	0.90	0.90	0.90	10
8	0.77	1.00	0.87	10
9	1.00	0.90	0.95	10
10	1.00	0.50	0.67	10
11	0.91	1.00	0.95	10
12	1.00	0.90	0.95	10
13	1.00	0.70	0.82	10
14	0.91	1.00	0.95	10
15	0.90	0.90	0.90	10
16	0.91	1.00	0.95	10
17	0.82	0.90	0.86	10
18	0.91	1.00	0.95	10
19	0.91	1.00	0.95	10
accuracy			0.91	200
macro avg	0.91	0.90	0.90	200
weighted avg	0.91	0.91	0.90	200

Tabel peforma *fine and tuning* model hasil skenario 3 menggunakan *confusion matrix*

label	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	0.91	1.00	0.95	10
2	0.73	0.80	0.76	10
3	0.90	0.90	0.90	10
4	1.00	1.00	1.00	10
5	1.00	1.00	1.00	10
6	1.00	1.00	1.00	10
7	0.91	1.00	0.95	10
8	1.00	1.00	1.00	10
9	1.00	0.90	0.95	10
10	0.89	0.80	0.84	10
11	0.82	0.90	0.86	10
12	1.00	1.00	1.00	10
13	1.00	0.80	0.89	10
14	1.00	1.00	1.00	10
15	1.00	0.90	0.95	10
16	1.00	1.00	1.00	10
17	1.00	1.00	1.00	10
18	1.00	1.00	1.00	10
19	0.91	1.00	0.95	10
accuracy			0.95	200
macro avg	0.95	0.95	0.95	200
weighted avg	0.95	0.95	0.95	200

Tabel peforma *fine and tuning* model hasil skenario 4 menggunakan *confusion matrix*

label	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	0.91	1.00	0.95	10
2	0.73	0.80	0.76	10
3	0.90	0.90	0.90	10
4	1.00	1.00	1.00	10
5	1.00	1.00	1.00	10
6	1.00	1.00	1.00	10
7	0.91	1.00	0.95	10
8	1.00	1.00	1.00	10
9	1.00	1.00	1.00	10
10	1.00	0.80	0.89	10
11	0.82	0.90	0.86	10
12	1.00	1.00	1.00	10
13	1.00	0.80	0.89	10
14	1.00	1.00	1.00	10
15	1.00	0.90	0.95	10
16	1.00	1.00	1.00	10
17	0.91	1.00	0.95	10
18	1.00	1.00	1.00	10
19	0.90	0.90	0.90	10
accuracy			0.95	200
macro avg	0.95	0.95	0.95	200
weighted avg	0.95	0.95	0.95	200

ABSTRACT

Shabrina, Idzni. 2023. **Face Recognition Using MobileNetV2.** Thesis. Informatics Engineering Department, Faculty of Science and Technology, Universitas Islam Negeri Maulana Malik Ibrahim Malang. Advisors: (I) Dr. M. Amin Hariyadi (II) Dr. Cahyo Crysdian.

Keywords: *Face recognition, MobileNetV2.*

Face recognition can facilitate human work because its use is practical and flexible. Nowadays, there are many methods of face recognition. However, the researcher did this research to develop existing methods and to find new methods continuously in order to produce better face recognition. In this research, the MobileNetV2 architecture will be used for face recognition. Several trial scenarios were carried out, and a confusion matrix was used to measure the performance of the model with the MobileNetV2 architecture when used as face recognition. The result shows the best performance with a weighted average precision value of 0.95, a weighted average recall value, an f1-score, and an accuracy of 0.94. Transfer learning produces precision, recall, f1-score, and accuracy values 0.94. Fine and tuning can improve the performance of the model. The model has increased the precision, recall, f1-score, and accuracy value to 0.95.

Translator	Date	Director of Language Center
Norma Noviana	10 – 7 – 2023	Dr. H. M. Abdul Hamid, MA. NIP. 19730701 1998031007

مستخلص البحث

صبرنا، إذني. ٢٠٢٣. التعرف على الوجوه باستخدام بنية MobileNetV2. البحث الجامعي. قسم الهندسة المعلوماتية، كلية العلوم والتكنولوجيا بجامعة مولانا مالك إبراهيم الإسلامية الحكومية مالانج. المشرف الأول: د. أمين هريادي. المشرف الثاني: د. جهيو كريسديان.

الكلمات الرئيسية: التعرف على الوجه، MobileNetV2.

يمكن التعرف على الوجه تسهيل العمل البشري لأن استخدامه عملي ومرن. حاليا، هناك العديد من طرق التعرف على الوجه. ومع ذلك، يستمر الباحثون في إجراء الأبحاث لتطوير الطرق الحالية وإيجاد طريقة جديدة لإنتاج التعرف على الوجه بشكل أفضل. في هذا البحث، سيتم استخدام بنية MobileNetV2 للتعرف على الوجه. تم تنفيذ العديد من سيناريوهات الاختبار وتم استخدام مصفوفة الارتباك لقياس أداء النموذج باستخدام بنية MobileNetV2 عند استخدامها كالتعرف على الوجه. وجدت نتائج قياس أداء النموذج أفضل أداء من عدة سيناريوهات الاختبار بمتوسط قيمة الضبط يبلغ ٠.٩٥ ومتوسط قيمة الاستدعاء ودرجة ف١ وقيمة الدقة ٠.٩٤. أنتج نقل التعلم قيمة الضبط والاستدعاء ودرجة ف١ وقيمة الدقة تبلغ ٠.٩٤. يمكن أن يؤدي الصقل والضبط إلى تحسين أداء النموذج، وقد زاد النموذج [illegible] قيمة الضبط والاستدعاء ودرجة ف١ والدقة إلى ٠.٩٥.

Penerjemah,	Tanggal 11-7-2023	Validasi Kepala [illegible],
M.Mubasysyir Munir, MA NIDT:19860513201802011215		Prof. Dr. H. M. Abdul Hamid, M.A NIP: 19730201 199803 1002

KEMENTERIAN AGAMA REPUBLIK INDONESIA