TESIS

EVALUASI PENGARUH FINE-TUNING LAYER TERHADAP AKURASI PENGENALAN WAJAH BERBASIS CONVOLUTIONAL NEURAL NETWORK (CNN)

disusun oleh

TOMMY SARAWAN
23.55.1375
Konsentrasi : Business Intelligence

FAKULTAS ILMU KOMPUTER
UNIVERSITAS AMIKOM YOGYAKARTA
YOGYAKARTA
2025

TESIS

EVALUASI PENGARUH FINE-TUNING LAYER TERHADAP AKURASI PENGENALAN WAJAH BERBASIS CONVOLUTIONAL NEURAL NETWORK (CNN)

EVALUATION OF THE EFFECT OF FINE-TUNING LAYER ON THE ACCURACY OF FACE RECOGNITION BASED ONCONVOLUTIONAL NEURAL NETWORK (CNN)

Diajukan untuk memenuhi salah satu syarat mencapai derajat Pascasarjana
Program Studi S2 PJJ Informatika

disusun oleh
TOMMY SARAWAN
23.55.1375
Konsentrasi : Business Intelligence

FAKULTAS ILMU KOMPUTER
UNIVERSITAS AMIKOM YOGYAKARTA
YOGYAKARTA
2025

HALAMAN PERSETUJUAN

EVALUASI PENGARUH FINE-TUNING LAYER TERHADAP AKURASI PENGENALAN WAJAH BERBASIS CONVOLUTIONAL NEURAL NETWORK (CNN)

EVALUATION OF THE EFFECT OF FINE-TUNING LAYER ON THE ACCURACY OF FACE RECOGNITION BASED ON CONVOLUTIONAL NEURAL NETWORK (CNN)

yang disusun dan diajukan oleh

TOMMY SARAWAN
23.55.1375

telah disetujui oleh Dosen Pembimbing Tesis
pada tanggal 3 September 2025

Dosen Pembimbing,

Prof. Dr. Kusrini,M.Kom
NIK. 190302100

HALAMAN PENGESAHAN

EVALUASI PENGARUH FINE-TUNING LAYER TERHADAP AKURASI PENGENALAN WAJAH BERBASIS CONVOLUTIONAL NEURAL NETWORK (CNN)

EVALUATION OF THE EFFECT OF FINE-TUNING LAYER ON THE ACCURACY OF FACE RECOGNITION BASED ONCONVOLUTIONAL NEURAL NETWORK (CNN)

yang disusun dan diajukan oleh

Tommy Sarawan
23.55.1375

Telah dipertahankan di depan Dewan Penguji
pada tanggal 3 September 2025

Susunan Dewan Penguji

Nama Penguji	Tanda Tangan
Dr Andi Sunyoto, M.Kom	
NIK. 190302052	
Dr. Kumara Ari Yuana,S.T., M.T.	
NIK. 190302575	
Prof. Dr. Kusrini.M.Kom	
NIK. 190302106	

Tesis ini telah diterima sebagai salah satu persyaratan
untuk memperoleh gelar Magister Komputer
Tanggal 3 September 2025

DEKAN FAKULTAS ILMU KOMPUTER

Prof. Dr. Kusrini, M.Kom.
NIK. 190302106

PERNYATAAN KEASLIAN

Yang bertandatangan di bawah ini,

Nama mahasiswa : Tommy Sarawan
NIM : 23.55.1375
Konsentrasi : Business Intelligence

Menyatakan bahwa Tesis dengan judul berikut:
Evaluasi Pengaruh Fine-Tuning Layer Terhadap Akurasi Pengenalan Wajah Berbasis Convolutional Neural Network (CNN)

Dosen Pembimbing Utama : Prof. Dr. Kusrini,M.Kom
Dosen Pembimbing Pendamping : Prof. Dr. Kusrini,M.Kom

1. Karya tulis ini adalah benar-benar ASLI dan BELUM PERNAH diajukan untuk mendapatkan gelar akademik, baik di Universitas AMIKOM Yogyakarta maupun di Perguruan Tinggi lainnya
2. Karya tulis ini merupakan gagasan, rumusan dan penelitian SAYA sendiri, tanpa bantuan pihak lain kecuali arahan dari Tim Dosen Pembimbing
3. Dalam karya tulis ini tidak terdapat karya atau pendapat orang lain, kecuali secara tertulis dengan jelas dicantumkan sebagai acuan dalam naskah dengan disebutkan nama pengarang dan disebutkan dalam Daftar Pustaka pada karya tulis ini
4. Perangkat lunak yang digunakan dalam penelitian ini sepenuhnya menjadi tanggung jawab SAYA, bukan tanggung jawab Universitas AMIKOM Yogyakarta
5. Pernyataan ini SAYA buat dengan sesungguhnya, apabila di kemudian hari terdapat penyimpangan dan ketidakbenaran dalam pernyataan ini, maka SAYA bersedia menerima SANKSI AKADEMIK dengan pencabutan gelar yang sudah diperoleh, serta sanksi lainnya sesuai dengan norma yang berlaku di Perguruan Tinggi

Yogyakarta, 3 September 2025
Yang Menyatakan,

Tommy Sarawan

HALAMAN PERSEMBAHAN

Puji dan Syukur saya panjatkan kepada Tuhan Yang Maha Esa atas segala berkat dan anugerah yang diberikan, saya masih bisa melanjutin perkuliahan saya hingga S2. Tidak lupa saya mengucap syukur dan berterima kasih juga kepada Kedua Orang Tua saya yang juga masih memberikan izin untuk saya tetap melanjutkan perkuliahan saya hingga jenjang S2. Dan semua teman kuliah angkatan 23 PJJ Magister Teknik Informatika Universitas Amikom Yogyakarta, saya mengucapkan banyak terima kasih atas dorongan dan bantuan teman-teman semua sehingga saya dapat menyelesaikan perkuliahan dengan baik. Tidak lupa saya mengucapkan terima kasih banyak untuk Universitas Amikom Yogyakarta yang sudah banyak memberikan ilmu dan pelajaran yang disampaikan selama masa perkuliahan saya.

KATA PENGANTAR

Puji syukur penulis panjtakan kehadirat Tuhan Yang Maha Esa atas segala berkat dan hikmatnya, penulisi masih diberi kesempatan dan kemudahan untuk menyelesaikan Tesis ini. Tesis ini disusun dalam rangka memenuhi salah satu syarat kelulusan perguruan tinggi Program Studi Strata-2 Magister Teknik Informatika di universitas Amikom Yogyakarta dan Meraih gelar Magister Teknik Komputer (M.Kom). Selain itu tesis ini juga bertujuan untuk mengevaluasi pengaruh variasi *fine-tuning layer* terhadap akurasi model CNN dalam pengenalan wajah, terutama yang menggunakan model pretrained populer arsitektur *MobileNetV2* dan *ResNetV2*. Penulis juga mengucapkan terimakasih yang sebesar-besarnya kepada:

1. Bapak Prof. Dr. M. Suyanto, MM, selaku Rektor dan Ketua Amikom Yogyakarta.
2. Ibu Prof. Dr. Kusrini, M. Kom dan Bapak Emha Taufiq selaku dosen pembimbing saya yang selalu memberikan masukan, saran, bantuan, dan bimbingan dalam menyelesaikan naskah Tesis ini.
3. Ibu Prof. Dr. Kusrini, M. Kom selaku Direktur Program Pascasarjana Universitas Amikom Yogyakarta.
4. Kedua Orang tua yang tak pernah lelah dalam memberikan support dan doanya.
5. Semua teman teman dilingkungan kerja di Nabire yang telah memberi masukan dan dorongan.
6. Semua teman-teman seperjuangan magister teknik Informatika Universitas Amikom Yogyakarta tahun akademik 2023 Kelas PJJ dan pihak-pihak yang penulis tidak dapat sebut satu persatu
7. Dosen Amikom Yogyakarta yang telah memberikan banyak ilmu dan pengalaman.

Semua pihak yang telah membantu penyelesaian tesis ini yang tentunya sangat berharga dan tidak bisa disebutkan satu persatu. Penulis juga meminta maaf apabila dalam penyusunan tesis ini masih banyak kekurangan dan masih jauh untuk memberikan kata sempurna. Penulis juga dengan senang hati menerima kritik dan saran.

Semoga tesis ini dapat menambah pengetahuan dan memberikan manfaat bagi para pembacanya maupun diri penulis sendiri serta dapat digunakan sebagai salah referensi untuk penelitian yang lain.

Yogyakarta, 3 September 2025

Penulis

DAFTAR ISI

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

INTISARI

Pesatnya kemajuan kecerdasan buatan dan *deep learning*, sehingga pengenalan wajah berbasis *Convolutional Neural Network* (CNN) menghadapi tantangan nyata ketika berhadapan dengan data berukuran terbatas dan sangat bervariasi. dalam konteks ini, penelitian terdahulu menyatakan *transfer learning* dengan *fine-tuning* mampu meningkatkan akurasi karena menyesuaikan *fitur pralatih* dengan karakteristik domain target. Penelitian ini berfokus pada peningkatan akurasi pengenalan wajah melalui pembandingan strategi *fine-tuning* pada dua arsitektur CNN yaitu *MobileNetV2* dan *ResNetV2*. Dataset yang digunakan terdiri atas 16 aktor dengan masing-masing 100 citra. Seluruh citra diproses melalui tahapan *cropping* otomatis berbasis *Haar Cascade*, pembersihan data, pembagian data, dan augmentasi untuk memperkuat generalisasi model. Penelitian ini merancang empat skenario pelatihan dimana dua pada *MobileNetV2* dan dua pada *ResNetV2* dengan variasi jumlah layer pralatih yang dibuka untuk di-*fine-tune*. Hasil pelatihan akan dievaluasi menggunakan *confusion matrix*. Hasil menunjukkan *MobileNetV2* dengan *fine-tuning* pada 50 layer terakhir memberikan performa terbaik dengan akurasi mencapai 91%, presisi 92%, recall 91% melampaui *ResNetV2* pada konfigurasi terbaiknya mencapai akurasi 89%, presisi 89%, recall 90%. Temuan ini menegaskan bahwa membuka sebagian besar layer akhir untuk menyesuaikan representasi fitur terhadap variasi wajah dapat meningkatkan akurasi model.

Kata kunci: Pengenalan wajah, CNN, *transfer learning*, *fine-tuning*

ABSTRACT

The rapid advances in artificial intelligence and deep learning mean that face recognition based on Convolutional Neural Networks (CNN) faces real challenges when dealing with limited and highly variable data. In this context, previous studies state that transfer learning with fine-tuning can increase accuracy because it adapts pretrained features to the characteristics of the target domain. This study focuses on improving face recognition accuracy by comparing fine-tuning strategies on two CNN architectures, namely MobileNetV2 and ResNetV2. The dataset used consists of 16 actors with 100 images each. All images were processed through stages of automated Haar Cascade–based cropping, data cleaning, data splitting, and augmentation to strengthen model generalization. This study designed four training scenarios, two on MobileNetV2 and two on ResNetV2, with variations in the number of pretrained layers that were unfrozen for fine-tuning. The training results will be evaluated using a confusion matrix. The results show that MobileNetV2 with fine-tuning on the last 50 layers provides the best performance with accuracy reaching 91%, precision 92%, recall 91%, surpassing ResNetV2 in its best configuration, which achieved 89% accuracy, 89% precision, and 90% recall. These findings affirm that unfreezing most of the final layers to adapt feature representations to facial variation can improve model accuracy.

Keyword: face recognition, CNN, transfer learning, fine-tuning

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Perkembangan teknologi *Artificial Intelligence* (AI) mengalami lonjakan signifikan dalam beberapa dekade terakhir. *Artificial Intelligence*, sebagai bagian dari ilmu komputer yang dimanfaatkan untuk membuat sistem yang dapat melakukan fungsi yang menyerupai kecerdasan seseorang, seperti pengenalan terhadap pola, pengambilan keputusan, dan pembelajaran data (Tjahyanti, Saputra, and Gitakarma 2022). *Deep learning*, cabang AI yang paling terkenal, menggunakan jaringan saraf tiruan untuk mengolah data yang kompleks seperti teks, gambar, dan suara (Ahmed et al. 2023).

Pengembangan teknologi pengenalan wajah (*face recognition*) telah mengalami kemajuan pesat dalam beberapa tahun terakhir, terutama dengan kemunculan algoritma *deep learning* yang memiliki kemampuan ekstraksi fitur citra secara otomatis dan efektif. Salah satu algoritma *deep learning* yang banyak digunakan adalah *Convolutional Neural Network* (CNN), yang terbukti sangat unggul dalam berbagai tugas *computer vision*, termasuk pengenalan wajah. Namun, dalam penerapan CNN pada pengenalan wajah, tantangan utama ialah bagaimana meningkatkan akurasi model, terutama ketika dataset yang tersedia tidak besar atau sangat bervariasi (Anggraini et al. 2024)

Salah satu teknik yang populer digunakan untuk meningkatkan performa CNN adalah *transfer learning* dengan metode *fine-tuning*. *Transfer learning*

memanfaatkan model CNN yang telah dilatih pada dataset besar (*pretrained model*) dan kemudian melakukan pelatihan ulang (*retraining*) pada dataset target dengan jumlah data yang lebih kecil atau berbeda domain (Ramadhani, Rahardiantoro, and Masjkur 2024). *Fine-tuning* sendiri merupakan proses membuka kembali dan melatih ulang sebagian atau seluruh layer dari model pretrained tersebut untuk menyesuaikan model dengan karakteristik dataset baru, sehingga model dapat mengadaptasi fitur yang lebih spesifik dan relevan terhadap permasalahan yang dihadapi (Sasongko and Amrullah 2023). Beberapa penelitian menunjukkan bahwa penerapan *fine-tuning* pada model CNN dapat meningkatkan akurasi misalnya, penelitian yang dilakukan oleh (Mikolaj et al. 2024) menemukan bahwa model yang sudah di-*fine-tuning* mampu mencapai akurasi mencapai 6% lebih tinggi dibandingkan model tanpa *fine-tuning*.

Tantangan yang dihadapi dalam pengenalan wajah adalah variasi kondisi seperti pencahayaan, pose wajah, ekspresi, dan kualitas gambar yang tidak konsisten. CNN yang dioptimalkan dengan *fine-tuning* harus mampu mengenali fitur wajah meskipun dihadapkan dengan variasi tersebut. Penelitian pada dataset kecil telah menunjukkan bahwa model CNN yang di-*fine-tune* dengan tepat dapat mengatasi sebagian variasi ini dan menghasilkan akurasi yang tinggi, bahkan mencapai lebih dari 90% (Anggraini et al. 2024).

Dalam konteks implementasi praktis, evaluasi terhadap *pengaruh fine-tuning* pada *layer-layer* CNN sangat diperlukan untuk memahami secara lebih mendalam bagaimana konfigurasi dan teknik pelatihan dapat mempengaruhi hasil akhir pengenalan wajah. Hal ini berkaitan dengan kemampuan model dalam

menghadapi data baru. Oleh karena itu, penelitian yang fokus pada evaluasi dampak *fine-tuning* layer terhadap akurasi pengenalan wajah berbasis CNN sangat relevan dan memberikan kontribusi penting bagi pengembangan sistem pengenalan wajah yang lebih akurat.

Permasalahan utama dalam penelitian ini berdasarkan pada penelitian sebelumnya yang menunjukkan keterbatasan model CNN dalam pengenalan wajah, terutama pada kondisi dataset terbatas dan bervariasi. (Chitrapu, Morampudi, and Kalluri 2025) menegaskan bahwa performa CNN sangat dipengaruhi oleh keterbatasan dataset, sehingga diperlukan pendekatan *fine-tuning* untuk meningkatkan akurasi. Meskipun CNN mampu mengekstraksi fitur wajah secara efektif, perbedaan pencahayaan, ekspresi, dan kualitas gambar masih menjadi tantangan yang signifikan dalam identifikasi. Oleh karena itu, penelitian mengenai optimasi *fine-tuning layer* pada arsitektur CNN untuk meningkatkan akurasi pengenalan wajah masi sangat relevan.

Dengan latar belakang tersebut, penelitian ini bertujuan untuk mengevaluasi pengaruh variasi *fine-tuning layer* terhadap akurasi model CNN dalam pengenalan wajah, terutama yang menggunakan model pretrained populer arsitektur *MobileNetV2* dan *ResNetV2*. Penelitian ini diharapkan dapat memberikan pemahaman yang lebih sistematis mengenai strategi *fine-tuning* yang optimal, sehingga dapat menjadi dasar pengembangan sistem pengenalan wajah yang handal di masa depan.

1.2. Rumusan Masalah

Berdasarkan pemaparan latar belakang di atas, dapat diperoleh rumusan masalah sebagai berikut:

a. Bagaimana membangun model pengenalan wajah dengan algoritma *Convolutional Neural Network* (CNN) menggunakan *transfer learning* dari arsitektur *MobileNetV2* dan *ResNetV2*?
b. Arsitektur mana yang memberikan performa terbaik untuk tugas pengenalan wajah?
c. Faktor apa yang mempengaruhi peningkatan akurasi model pengenalan wajah dari arsitektur *Convolutional Neural Network* (CNN)?

1.3. Batasan Masalah

Agar penelitian tetap fokus pada rumusan masalah perlu untuk menentukan batas masalah. Batasan dari penelitian ini adalah sebagai berikut:

a. Penelitian ini menggunakan algoritma *Convolutional Neural Network* (CNN) dengan teknik transfer learning dari arsitektur *MobileNetV2* dan *ResNetV2*.
b. Untuk menjaga privasi, keamanan data, efek psikologis dan etika penelitian ini menggunakan data public berupa gambar aktor *Hollywood* yang diperolah dari platform online.
c. Penelitian ini menggunakan empat skenario pengujian dengan memanfaatkan teknik *fine-tuning* yang berbeda pada kedua arsitektur yang telah ditentukan.
d. Proses evaluasi menggunakan *confusion matrix*.

1.4. Tujuan Penelitian

Berdasarkan rumusan masalah, penelitian ini bertujuan untuk:

a. Membangun model pengenalan wajah dengan algoritma *Convolutional Neural Network* (CNN) menggunakan *transfer learning* dari arsitektur *MobileNetV2* dan *ResNetV2*.
b. Mengetahui arsitektur mana yang memberikan performa terbaik untuk tugas pengenalan wajah.
c. Mengetahui faktor apa yang mempengaruhi peningkatan akurasi model pengenalan wajah dari arsitektur *Convolutional Neural Network* (CNN).

1.5. Manfaat Penelitian

Manfaat dari penelitian ini adalah:

a. Bagi peneliti, mengetahui arsitektur yang tepat dan tingkat akurasi tertinggi antara model klasifikasi *Convolutional Neural Network* (CNN)
b. Memberikan pengetahuan baru mengenai model yang dapat melakukan pengenalan wajah menggunakan *Convolutional Neural Network* (CNN)

BAB II

TINJAUAN PUSTAKA

2.1. Tinjauan Pustaka

Penelitian yang dilakukan oleh (Kumar and Kumar 2025) dengan judul *Facial Recognition and Object Detection using Machine learning.* Penelitian ini bertujuan mengembangkan sistem pengenalan wajah berbasis *deep learning* menggunakan arsitektur VGG16 dengan data *augmentation* untuk meningkatkan generalisasi dan akurasi model. Penelitian ini menyimpulkan bahwa *transfer learning* dengan VGG16 dan data *augmentation* efektif untuk pengenalan wajah dengan akurasi validasi mencapai 76,39% dengan menggunakan lima *epoch*. Penelitian ini menggunakan dataset yang sama dengan penelitian yang akan dilakukan yaitu *celebrity-face-recognition-dataset* tetapi hanya menggunakan arsitektur VGG16 sedangkan penelitian yang akan dilakukan menggunakan akritektur MobileNetV2 dan ResNetV2.

Penelitian yang dilakukan oleh (Mikolaj et al. 2024) dengan judul *Improving classification accuracy of fine-tuned CNN models: Impact of hyperparameter optimization* penelitian ini bertujuan untuk meningkatkan akurasi klasifikasi model CNN pretrained dengan mengoptimalkan strategi *fine-tuning* pada arsitektur *Xception*. Penelitian ini menyimpulkan bahwa melatih ulang sebagian layer terakhir pada model CNN pretrained sekaligus membekukan (*freeze*) layer awal secara signifikan dapat meningkatkan akurasi klasifikasi hingga 6% dibandingkan dengan pelatihan tanpa *fine-tuning*. Penelitian ini hanya

menggunakan arsitektur *Xception* untuk menguji beberapa teknik untuk meningkatkan akurasi. Sedangkan penelitian yang akan dilakukan menggunakan akritektur MobileNetV2 dan ResNetV2.

Penelitian yang dilakukan oleh (Astawa et al. 2021) dengan judul *Face Images Classification using VGG-CNN* dengan tujuan meningkatkan akurasi klasifikasi gambar wajah menggunakan VGG-CNN dengan *transfer learning*. Penelitian ini menggunakan dataset besar yang terdiri dari 36.600 gambar wajah dengan ukuran 224x224 piksel yang dikumpulkan dari berbagai sumber seperti ponsel, kamera digital, dan media sosial. Penelitian ini menyimpulkan bahwa akurasi validasi mencapai 99.84% untuk gambar dari kamera digital, dengan loss 0.69%. Penelitian ini juga mengidentifikasi bahwa kualitas gambar sangat memengaruhi hasil, di mana gambar dari media sosial memiliki akurasi lebih rendah (92.75%) dibandingkan sumber lainnya. Penelitian ini relevan dengan penelitian yang akan dilakukan karena sama menggunakan *transfer learning*, tetapi penelitian yang akan dilakuka akan menguji beberapa arsitektur sedangkan penelitian ini hanya menggunakan arsitektur VGG.

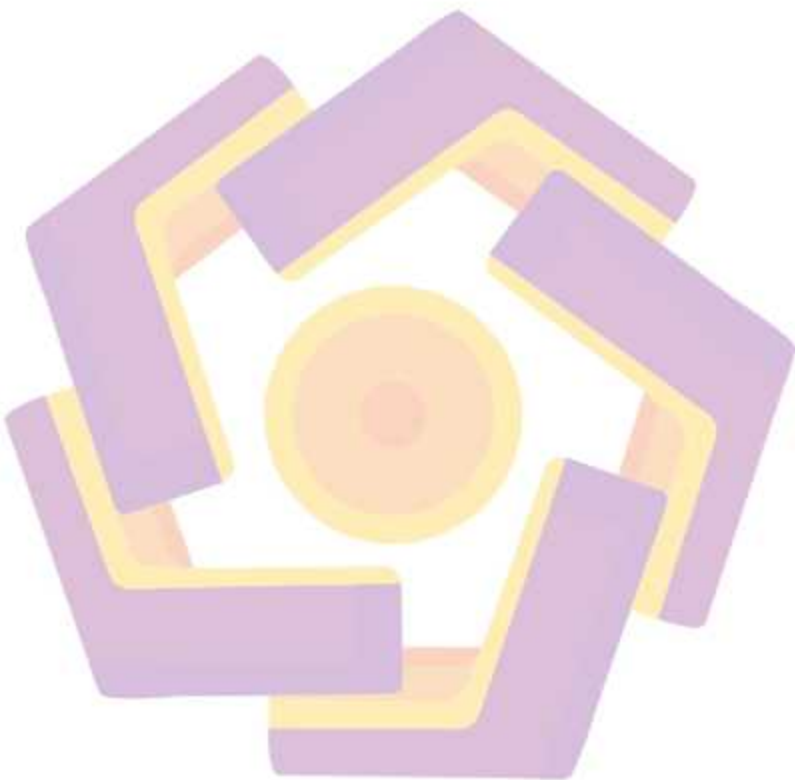
Penelitian yang dilakukan oleh (Shukla and Tiwari 2023) dengan judul *Masked Face Recognition Using MobileNet V2 with Transfer Learning* penelitian ini bertujuan untuk mengembangkan model pengenalan wajah tertutup masker untuk mengurangi penyebaran COVID-19 dengan akurasi tinggi. Penelitian ini menyimpulkan bahwa model dengan arsitektru MobileNetV2 mencapai akurasi 99,82%, lebih unggul dari arsitektur VGG16, VGG19, ResNet50, dan ResNet101. Penelitian ini relevan dengan penelitian yang akan dilakukan karena sama-sama

menggunakan transfer learning dan arsitektur CNN, tetapi berbeda dalam fokusnya yaitu pengenalan wajah tertutup masker sedangkan penelitian yang akan dilakukan adalah membangun model untuk pengenalan wajah.

Penelitian yang dilakukan oleh (Ai et al. 2022) dengan judul *Real-Time Facemask Detection for Preventing COVID-19 Spread Using Transfer Learning Based Deep Neural Network* yang bertujuan untuk mengembangkan sistem deteksi masker wajah berbasis *deep learning* untuk mencegah penyebaran COVID-19 secara *real-time*. Penelitian ini menyimpulkan bahwa arsitektur ResNetV2 dan MobileNetV2 menghasilkan akurasi yang cukup baik dalam melakukan deteksi wajah dengan akurasi ResNetV2 90,49% dan MobileNetV2 89,18% dibandingkan dengan arsitektur lain seperti AlexNet, VGG16, dan InceptionV2. Penelitian ini relevan dengan penelitian yang akan dilakukan karena sama-sama menggunakan *transfer learning* dan arsitektur CNN, tetapi berbeda dalam fokusnya, yaitu pengenalan wajah tertutup masker sedangkan penelitian yang akan dilakukan adalah membangun model untuk pengenalan wajah.

Penelitian yang dilakukan oleh (Anarki, Auliasari, and Orisa 2021) dengan judul Penerapan Metode Haar Cascade pada Aplikasi Deteksi Masker yang bertujuan untuk mengembangkan aplikasi pendeteksi penggunaan masker untuk meminimalisir penyebaran COVID-19 dengan fitur peringatan audio dan pemotretan. Hasil penelitian menunjukkan metode *Haar Cascade* mencapai akurasi tertinggi 88.7% pada faktor skala 1.2, tetapi menurun drastis menjadi 44.9% pada skala 1.3, menunjukkan sensitivitas terhadap parameter. Penelitian ini relevan dengan konteks penelitian yang akan dilakukan dalam hal penggunaan *Haar*

Cascade untuk deteksi wajah, tetapi berbeda tujuan antara deteksi masker dan pengenalan wajah.



2.2. Keaslian Penelitian

Tabel 2.1 Matriks literatur review dan posisi penelitian evaluasi pengaruh *fine-tuning layer* terhadap akurasi pengenalan wajah berbasis *Convolutional Neural Network* (CNN)

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
1	*Facial Recognition and Object Detection using Machine learning*	Abhishek Kumar, Kapil Kumar, *Journal of Recent Innovations in Computer Science and Technology*, 2025	Penelitian ini bertujuan mengembangkan sistem pengenalan wajah berbasis deep learning menggunakan arsitektur VGG16 dengan data augmentation untuk meningkatkan generalisasi dan akurasi model	Penelitian ini menyimpulkan bahwa transfer learning dengan VGG16 dan data augmentation efektif untuk pengenalan wajah dengan akurasi validasi mencapai 76,39% dengan menggunakan lima epoch.	Peningkatan akurasi dengan augmentasi data dan eksplorasi arsitektur CNN lebih dalam.	Penelitian ini menggunakan dataset yang sama dengan penelitian yang akan dilakukan yaitu *celebrity-face-recognition-dataset*. Selain itu, penelitian ini hanya menggunakan arsitektur VGG16 sedangkan penelitian yang akan dilakukan menggunakan akritektur MobileNetV2 dan ResNetV2
2	*Improving classification accuracy of fine-tuned CNN models: Impact of hyperparameter optimization*	Wojciuk Mikolaj, Swiderska-Chadaj Zaneta, Siwek Krzysztof, Gertych Arkadiusz, *Heliyon*, 2024	Untuk meningkatkan akurasi klasifikasi model CNN pretrained dengan mengoptimalkan strategi *fine-tuning* pada arsitektur *Xception*.	Melatih ulang sebagian layer terakhir pada model CNN pretrained sekaligus membekukan (*freeze*) layer awal secara signifikan dapat meningkatkan akurasi klasifikasi hingga 6% dibandingkan dengan	Penelitian ini hanya focus pada satu arsitektur yaitu *Xception*.	Penelitian ini hanya menggunakan arsitektur *Xception* untuk menguji beberapa teknik untuk meningkatkan akurasi. Sedangkan penelitian yang akan dilakukan menggunakan akritektur MobileNetV2 dan ResNetV2.

Lanjutan

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
				pelatihan tanpa *fine-tuning*		
3	*Face Images Classification using VGG-CNN*	I Nyoman Gede Arya Astawa, Made Leo Radhitya, I Wayan Raka Ardana, Felix Andika Dwiyanto, *Knowledge Engineering and Data Science* (KEDS), 2021	Meningkatkan akurasi klasifikasi gambar wajah menggunakan VGG-CNN dengan *transfer learning*. Penelitian ini menggunakan dataset besar yang terdiri dari 36.600 gambar wajah dengan ukuran 224x224 piksel, yang dikumpulkan dari berbagai sumber seperti ponsel, kamera digital, dan media sosial.	Penelitian ini menyimpulkan bahwa akurasi validasi mencapai 99.84% untuk gambar dari kamera digital, dengan loss 0.69%. Penelitian ini juga mengidentifikasi bahwa kualitas gambar sangat memengaruhi hasil, di mana gambar dari media sosial memiliki akurasi lebih rendah (92.75%) dibandingkan sumber lainnya.	Kelemahan utama penelitian ini adalah ketergantungan pada kualitas gambar input dan kebutuhan untuk optimasi lebih lanjut agar dapat beradaptasi dengan berbagai kondisi pencahayaan dan resolusi	Penelitian ini sama-sama menggunakan *transfer learning*, tetapi penelitian yang akan dilakuka akan menguji beberapa arsitektur sedangkan penelitian ini hanya menggunakan arsitektur VGG.
4	*Masked Face Recognition Using MobileNet V2 with Transfer Learning*	Ratnesh Kumar Shukla, Arvind Kumar Tiwari, *Computer Systems Science & Engineering*, 2023	Mengembangkan model pengenalan wajah tertutup masker untuk mengurangi penyebaran COVID-19 dengan akurasi tinggi.	Penelitian ini menyimpulkan bahwa model dengan arsitektru MobileNetV2 mencapai akurasi 99,82%, lebih unggul dari arsitektur VGG16, VGG19,	Penelitian ini memiliki keterbatasan dalam hal kualitas gambar sebagai dataset yang digunakan.	Penelitian ini relevan dengan penelitian yang akan dilakukan karena sama-sama menggunakan transfer learning dan arsitektur CNN, tetapi berbeda dalam fokusnya, yaitu pengenalan wajah tertutup masker

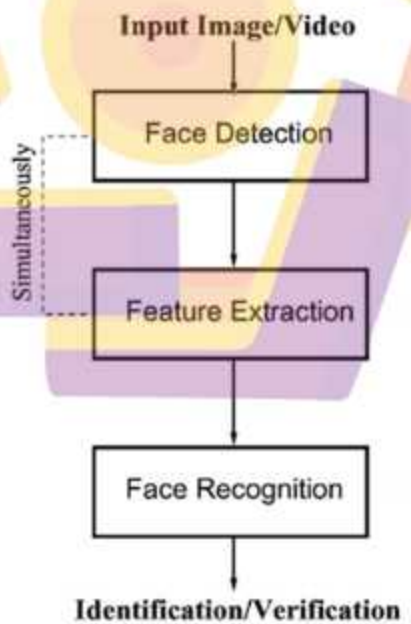
Lanjutan

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
				ResNet50, dan ResNet101		sedangkan penelitian yang akan dilakukan adalah membangun model untuk pengenalan wajah.
5	*Real-Time Facemask Detection for Preventing COVID-19 Spread Using Transfer Learning Based Deep Neural Network*	Mona A. S. Ai, Anitha Shanmugam, Suresh Muthusamy, Chandrasekaran Viswanathan, Hitesh Panchal, Mahendran Krishnamoorthy, Diaa Salama Abd Elminaam, Rasha Orban, *Electronics*, 2022	Mengembangkan sistem deteksi masker wajah berbasis deep learning untuk mencegah penyebaran COVID-19 secara real-time	Penelitian ini menyimpulkan bahwa arsitektur ResNetV2 dan MobileNetV2 menghasilkan akurasi yang cukup baik dalam melakukan deteksi wajah dengan akurasi ResNetV2 90,49% dan MobileNetV2 89,18% dibandingkan dengan arsitektur lain seperti AlexNet, VGG16, dan InceptionV2.	Model yang dibangun sebaiknya diuji pada wajah tertutup parsial atau masker yang tidak sempurna sehingga diharapkan lebih relevan saat diimplementasikan.	Penelitian ini relevan dengan penelitian yang akan dilakukan karena sama-sama menggunakan *transfer learning* dan arsitektur CNN, tetapi berbeda dalam fokusnya, yaitu pengenalan wajah tertutup masker sedangkan penelitian yang akan dilakukan adalah membangun model untuk pengenalan wajah.
6	Penerapan Metode Haar Cascade pada Aplikasi Deteksi Masker	Galang Aprilian Anarki, Karina Auliasari, Mira Orisa, JATI (Jurnal Mahasiswa Teknik Informatika), 2021	Mengembangkan aplikasi pendeteksi penggunaan masker untuk meminimalisir penyebaran COVID-19 dengan fitur peringatan audio dan pemotretan	Hasil penelitian menunjukkan metode *Haar Cascade* mencapai akurasi tertinggi 88.7% pada faktor skala 1.2, tetapi menurun drastis menjadi 44.9% pada skala 1.3, menunjukkan sensitivitas terhadap parameter.	Akurasi model bisa lebih ditingkatkan dengan berbagai cara seperti penambahan datasat, *fine-tuning* atau *pre-processing* yang tepat.	Penelitian ini relevan dengan konteks penelitian yang akan dilakukan dalam hal penggunaan *Haar Cascade* untuk deteksi wajah, tetapi berbeda tujuan antara deteksi masker dan pengenalan wajah.

2.3. Landasan Teori

2.3.1. Konsep Dasar Pengenalan Wajah

Pengenalan wajah (*face recognition*) merupakan salah satu cabang penting dalam bidang *computer vision* dan biometrik yang bertujuan untuk mengidentifikasi atau memverifikasi individu berdasarkan karakteristik unik yang terdapat pada wajah. Proses ini melibatkan serangkaian tahapan, mulai dari deteksi wajah (*face detection*), ekstraksi fitur, hingga klasifikasi (Zhao et al. 2003). Pengenalan wajah berbeda dengan deteksi wajah karena tidak hanya berfokus pada lokasi wajah dalam gambar, tetapi juga menganalisis ciri-ciri spesifik seperti struktur mata, hidung, dan mulut untuk membedakan satu individu dengan individu lainnya sebagai ilustrasi konfigurasi sistem pengenalan wajah dapat dilihat pada gambar 2.1.



Gambar 2. 1. Konfigurasi sistem pengenalan wajah

Teknologi ini telah berkembang pesat dalam beberapa dekade terakhir, terutama dengan adanya pendekatan berbasis *deep learning* seperti *Convolutional Neural Network* (CNN), yang mampu mengekstraksi fitur wajah secara otomatis dengan akurasi tinggi (Alhanaee et al. 2021) (Shukla and Tiwari 2023). Sistem pengenalan wajah menghadapi berbagai tantangan yang dapat memengaruhi kinerjanya. Salah satunya adalah variasi kondisi pencahayaan, di mana wajah yang terekam dalam lingkungan gelap atau terlalu terang dapat menyulitkan proses ekstraksi fitur. Selain itu, perubahan pose wajah (seperti menunduk atau menengok) serta ekspresi (tersenyum, cemberut) juga dapat mengurangi akurasi identifikasi (Nurkhamid et al. 2021).

Menurut (Ramdhani and Sela 2023) Proses face recognition secara umum terdiri dari:

1. Modul akuisisi, berupa input untuk proses face recognition yang dapat diperoleh dari citra digital maupun kamera.
2. Modul *pre-processing*, berupa proses penyesuaian citra input seperti melakukan normalisasi citra, *median filtering* untuk menghilangkan *noise* akibat pergeseran *frame* atau kamera, *histogram equalization* untuk memudahkan proses pengenalan citra dengan memperbaiki kualitas citra tanpa menghilangkan informasi utamanya, *high pass filtering* untuk mendapatkan sisi tepi suatu citra, *background removal* untuk menghilangkan bagian *background* sehingga bagian wajah saja yang diproses dan grayscaling untuk

mengkonversi citra RGB menjadi citra skala abu-abu. *Preprocessing* dilakukan untuk menghilangkan masalah yang akan timbul saat proses *face recognition*

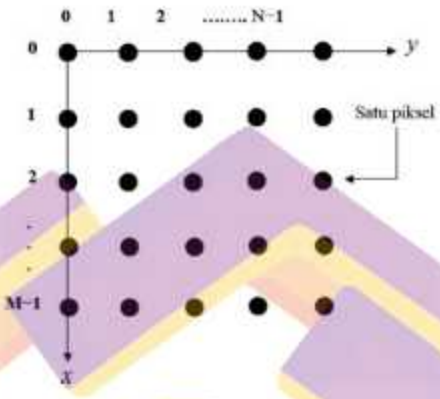
3. Modul ekstraksi fitur, untuk mendapatkan bagian terpenting sebagai suatu vektor yang merepresentasikan wajah dan bersifat unik.
4. Modul klasifikasi, dengan adanya pemisahan pola, fitur wajah dibandingkan dengan fitur yang tersimpan dalam database untuk mengetahui apakah citra wajah tersebut dapat dikenali.
5. *Training set*, modul ini digunakan selama proses pembelajaran (*training*), proses identifikasi/pengenalan, dan semakin kompleks dan sering proses pengenalan wajah akan semakin baik.

2.3.2. Pengolahan Citra Digital

Pengolahan citra digitala adalah ilmu yang mempelajari tentang algoritma transformasi citra (Asmara 2018). Citra dalah kumpulan piksel dalam ruang warna yang berbeda yang menjadi representasi visual dari objek kehidupan nyata dalam bentuk dua dimensi (Singh 2019) dan menurut (Kusumanto and Tompunu 2011) citra digital secara matematis merupakan fungsi kontinyu dengan intensitas cahaya pada bidang dua dimensi. Agar dapat diolah dengan komputer, citra digital harus melalui proses digitalisasi sehingga dapat dipresentasikan secara numerik dengan nilai-nilai diskrit.

Menurut (Andono, Sutojo, and Muljono 2017) citra digital merupakan matriks *M* baris dan *N* kolom yang terbentuk dari bilangan real. Perpotongan antara

baris dan kolom dari matriks tersebut, disebut piksel atau elemen terkecil dari sebuah citra seperti pada gambar 2.2



Gambar 2.2. Sistem kordinat yang mewakili citra

Piksel mempunya dua parameter, yaitu koordinat dan intensitas atau warna. Nilai yang terdapat pada koordinat (x,y) adalah *f(x,y)* sehingga sebiah citra digital dapat ditulis dalam bentuk matriks berikut:

(1)

Citra dalam fungsi matematis dapat ditulis , dan

Dimana : M = Jumlah piksel baris pada array citra

N = Jumlah piksel kolom pada array citra

G = Nilai skala keabuan (*graylevel*)

Besarnya nilai *M, N* dan *G* pada fungsi di atas umumnya merupakan perpangkatan dari dua sehingga dapat ditulis sebagai berikut.

(2)

Nilai *m, n* dan *k* adalah bilangan bilangan bulat positif. Interval (0,G) disebut skala keabuan (grayscale). Besar G tergantung pada proses digitalisasinya.

Biasanya keabuan 0 (nol) menyatakan intensitas hitam dan 1 (satu) menyatakan intensitas putih. Untuk citra 8 bit, nilai G sama dengan $2^8 = 256$ warna (derajat keabuan).

2.3.3. *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) adalah salah satu algoritma *deep learning* yang digunakan untuk kasus-kasus penggunaan *computer vision* seperti mengklasifikasikan gambar atau video dan mendeteksi objek di dalam gambar atau bahkan wilayah dalam gambar (Moolayil 2019). Dan menurut (Zufar and Setiyono 2016) CNN adalah variasi dari *Multilayer Perceptron* (MLP) yang terinspirasi dari jaringan syaraf pada manusia. CNN merupakan suatu layer yang memiliki susunan neuron 3D (lebar, tinggi dan kedalaman). Lebar dan tinggi merupakan ukuran layer sedangkan kedalaman mengacu pada jumlah layer. Secara umum jenis layer pada CNN dibedakan menjadi dua yaitu:

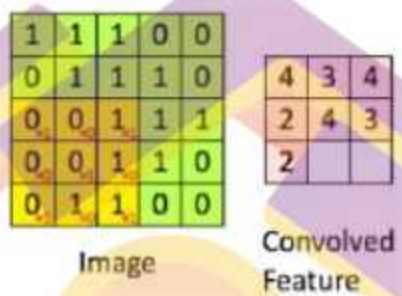
2.3.3.1. *Layer* Ekstraksi Fitur Gambar

Layer ini berada pada awal arsitektur tersusun atas beberapa layer dan setiap layer tersusun atas neuron yang terkoneksi pada daerah lokal (local region) *layer* sebelumnya. *Layer* ini menerima input gambar secara langsung dan memprosesnya hingga menghasilkan keluaran berupa vektor untuk diolah pada layer berikutnya. Lapisan ekstraksi fitur ini terdiri dari :

a. *Convolutional Layer*

Konvolusi adalah suatu istilah matematis yang berati mengaplikasikan sebuah fungsi pada *output* fungsi lain secara berulang. Dalam pengolahan citra,

konvolusi berati mengaplikasikan sebuah kernel (kotak kuning) pada citra disemua offset yang memungkinkan seperti yang ditunjukkan pada Gambar 2. Kotak hijau secara keseluruhan adalah citra yang akan dikonvolusi. Kernel bergerak dari sudut kiri atas ke kanan bawah. Sehingga hasil konvolusi dari citra tersebut dapat dilihat pada gambar disebelah kanan pada gambar 2.3.



Gambar 2.3. Operasi konvolusi

Tujuan dilakukannya konvolusi pada data citra adalah untuk mengekstraksi fitur dari citra input bahkan menurut (Liu 2018) konvolusi adalah cara yang efisien untuk ekstraksi fitur. Konvolusi akan menghasilkan transformasi linear dari data input sesuai informasi spasial pada data. Bobot pada layer tersebut menspesifikasikan kernel konvolusi yang digunakan, sehingga kernel konvolusi dapat dilatih berdasarkan input pada CNN.

b. *Rectified Linear Unit* (ReLU)

ReLU bertujuan untuk menjaga hasil citra proses konvolusi berada pada domain definit positif. ReLu merupakan salah satu fungsi aktivasi populer dalam deep neural network. Pada fungsi aktivasi dapat mengubah jumlah angka pembobotan dari input yang masuk ke dalam neuron buatan. Fungsi ini harus bersifat non-linear untuk mengkodean pola yang kompleks dari data. Aktivasi

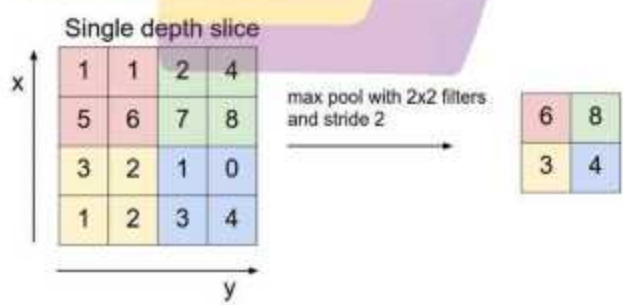
memiliki dua jenis yaitu *Sigmoid* dan *Tanh*. Persamaan yang biasa digunakan dalam fungsi ini adalah :

(3)

Dimana : x = nilai pada *feature map*

c. *Subsampling Layer* atau *Pooling Layer*

Proses ini bertujuan untuk mereduksi ukuran sebuah data citra. Dalam pengolahan citra, *subsampling* juga bertujuan untuk meningkatkan invariansi posisi dari fitur. Dalam sebagian besar CNN, metode *subsampling* yang digunakan adalah *max pooling*. *Max pooling* membagi *output* dari *convolution layer* menjadi beberapa *grid* kecil lalu mengambil nilai maksimal dari setiap *grid* untuk menyusun matriks citra yang telah direduksi seperti yang ditunjukkan pada gambar 2.3. *Grid* yang berwarna merah, hijau, kuning dan biru merupakan kelompok grid yang akan dipilih nilai maksimumnya. Sehingga hasil dari proses tersebut dapat dilihat pada kumpulan *grid* disebelah kanannya. Proses tersebut memastikan fitur yang didapatkan akan sama meskipun objek citra mengalami translasi (pergeseran) seperti pada gambar 2.4.



Gambar 2.4. Operasi *max pooling*

2.3.3.2. *Layer* klasifikasi

Layer ini menerima input dari hasil keluaran *layer* ekstrasi fitur gambar berupa vektor kemudian ditransformasikan seperti *Multi Neural Networks* dengan tambahan beberapa hidden layer. Hasil keluaran berupa skoring kelas untuk klasifikasi. Lapisan klasifikasi terdidiri dari :

a. *Flatten*

Proses ini bertujuan untuk membentuk ulang fitur (*reshape feature map*) menjadi sebuah vector agar bisa kita gunakan sebagai *input* dari *fully-connected layer*.

b. *Fully-connected*

Lapisan *Fully-connected* akan menghitung skor kelas. Seperti Jaringan Saraf biasa dan seperti namanya, setiap neuron dalam lapisan ini akan terhubung ke semua angka dalam volume.

c. *Softmax*

Fungsi *Softmax* menghitung probabilitas dari setiap kelas target atas semua kelas target yang memungkinkan dan akan membantu untuk menentukan kelas target untuk input yang diberikan. Keuntungan utama menggunakan *Softmax* adalah rentang probabilitas output dengan nilai 0 hingga 1, dan jumlah semua probabilitas akan sama dengan satu. Jika fungsi *softmax* digunakan untuk model multi-klasifikasi, dia akan mengembalikan peluang dari masing-masing kelas dan kelas target akan memiliki probabilitas tinggi. *Softmax* menggunakan eksponensial (e-

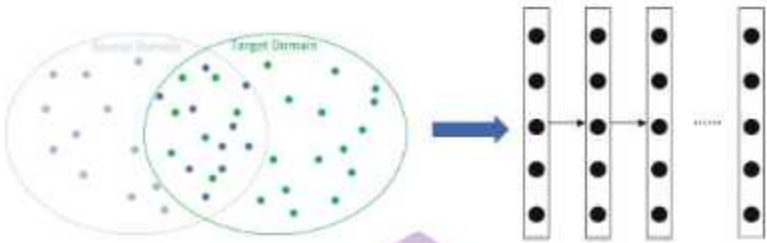
power) dari nilai input yang diberikan dan jumlah nilai eksponensial dari semua nilai dalam input. Maka rasio eksponensial dari nilai input dan jumlah nilai eksponensial adalah output dari fungsi *softmax*.

2.3.4. *Transfer Learning* dan *Fine-Tuning*

Transfer learning adalah suatu bentuk *machine learning* di mana model yang dibangun untuk tugas tertentu kemudian dimodifikasi untuk digunakan kembali pada tugas lain. Teknik ini banyak digunakan dalam *deep learning* sebagai model *pre-trained* untuk tugas *computer vision* dan *natural language processing* untuk mengembangkan model *neural network* (Tan et al. 2018). Adapun kategori *Transfer learning* adalah sebagai berikut:

2.3.4.1. *Instances-Based Deep Transfer Learning*

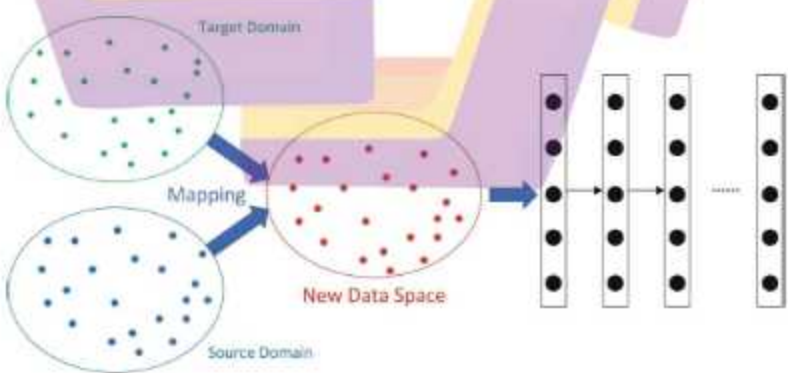
Pendekatan ini memilih sampel data dari *source domain* yang relevan untuk *target domain* dengan memberikan bobot tertentu, sementara sampel yang tidak relevan diabaikan. Teknik seperti *TrAdaBoost* menggunakan strategi *reweighting* untuk menyesuaikan distribusi data *source* agar lebih mirip dengan target. Keunggulannya adalah efisiensi komputasi, tetapi efektivitasnya bergantung pada kualitas seleksi sampel dan kesamaan dasar antara domain. Berikut gambar ilustrasi *instances-based deep transfer learning* terlihat pada gambar 2.5.



Gambar 2. 5. *Instances-based deep transfer learning*

2.3.4.2. *Mapping-Based Deep Transfer Learning*

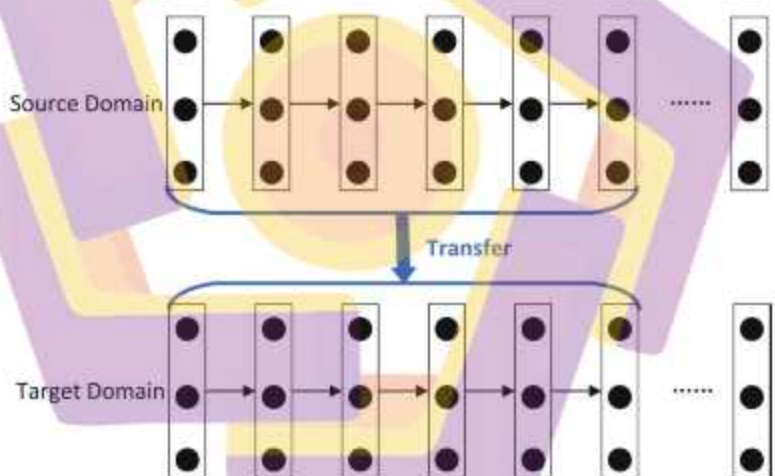
Metode ini mentransformasikan fitur dari *source domain* dan *target domain* ke suatu ruang bersama (*shared feature space*) di mana kedua domain menjadi lebih mirip. Teknik seperti *Maximum Mean Discrepancy* (MMD) atau *Wasserstein Distance* digunakan untuk meminimalkan perbedaan distribusi. Pendekatan ini cocok untuk domain dengan struktur berbeda, tetapi memerlukan komputasi tambahan untuk proses transformasi. Berikut gambar ilustrasi *mapping-based deep transfer learning* terlihat pada gambar 2.6



Gambar 2. 6. *Mapping-based deep transfer learning*

2.3.4.3. *Network-Based Deep Transfer Learning*

Pendekatan paling populer ini memanfaatkan sebagian atau seluruh arsitektur jaringan saraf seperti ResNet dan MobileNet yang telah dilatih sebelumnya pada dataset besar seperti ImageNet, lalu melakukan *fine-tuning* pada lapisan tertentu untuk tugas target. Asumsinya adalah lapisan awal DNN sebagai ekstraktor fitur umum yang dapat digunakan ulang. Efisien untuk dataset kecil, tetapi performa menurun jika domain sumber dan target terlalu berbeda. Berikut gambar ilustrasi *network-based deep transfer learning* terlihat pada gambar 2.7

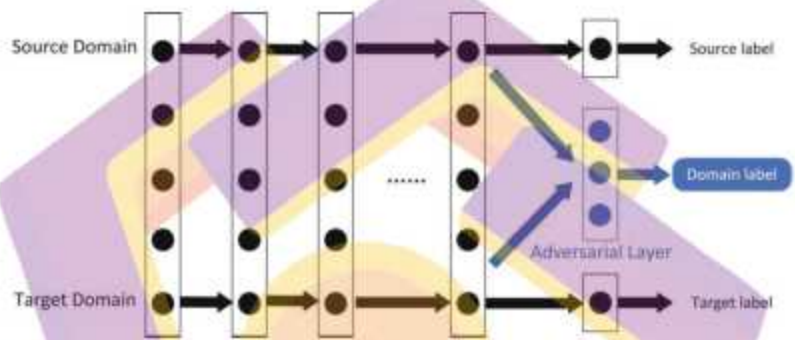


Gambar 2. 7. *Network-based deep transfer learning*

2.3.4.4. *Adversarial-Based Deep Transfer Learning*

Mengadopsi konsep *Generative Adversarial Networks* (GANs), pendekatan ini melatih model untuk menghasilkan fitur yang tidak bisa dibedakan asal

domainnya (*source domain*). Contohnya adalah *Domain-Adversarial Neural Networks* (DANN) yang menggunakan *adversarial loss* untuk menyelaraskan distribusi. Cocok untuk domain dengan perbedaan signifikan, tetapi pelatihannya lebih kompleks dan rentan tidak stabil. Berikut gambar ilustrasi *adversarial-based deep transfer learning* terlihat pada gambar 2.



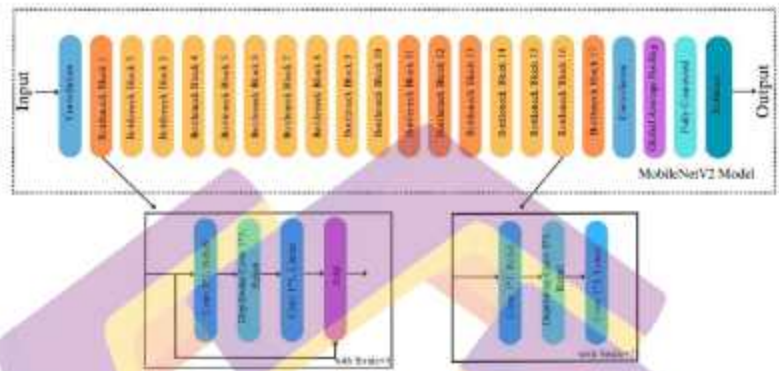
Gambar 2. 8. *Adversarial-based deep transfer learning*

2.3.5. Arsitektur CNN

2.3.5.1. MobileNetV2

MobileNetV2 merupakan penyempuranaan dari arsistektur MobileNet. Aristektur MobileNet dan arsitektur CNN pada umumnya memiliki perbedaan pada penggunaan lapisan atau *convolution layer*. arsitektur MobileNetV2 adalah penggunaan *depthwise separable convolution* dan *inverted residual blocks*. *Depthwise separable convolution* adalah teknik konvolusi yang membagi proses konvolusi menjadi dua tahap, yaitu *depthwise convolution* dan *pointwise convolution* sehingga dapat mengurangi jumlah parameter dan komputasi yang

dibutuhkan (Arief, Putra, and Pratama 2024). Berikut arsitektur MobileNetV2 dapat dilihat pada gambar 2.9.



Gambar 2. 9. Arsitektur MobileNetV2

Inverted residual blocks adalah struktur yang terdiri dari dua lapisan konvolusi, yaitu lapisan konvolusi dengan filter kecil dan lapisan konvolusi dengan filter besar. Lapisan konvolusi dengan filter kecil digunakan untuk mengurangi jumlah parameter dan komputasi, sedangkan lapisan konvolusi dengan filter besar digunakan untuk meningkatkan akurasi (Arief, Putra, and Pratama 2024).

MobileNetV2 dikembangkan sebagai arsitektur *lightweight* yang mengutamakan efisiensi dalam penggunaan memori dan kecepatan komputasi. Kelebihan utamanya terletak pada penggunaan *depthwise separable convolution*, yaitu teknik yang memisahkan proses konvolusi spasial dan konvolusi kanal. Dengan cara ini, jumlah parameter dan kebutuhan komputasi dapat dikurangi secara signifikan dibandingkan CNN konvensional. Selain itu, MobileNetV2 memperkenalkan *inverted residuals* dan *linear bottlenecks* yang menjaga representasi fitur tetap efisien tanpa kehilangan informasi penting. Karakteristik ini

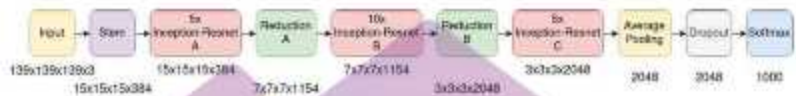
membuat MobileNetV2 sangat sesuai untuk perangkat dengan keterbatasan sumber daya, seperti smartphone atau sistem embedded, namun tetap mempertahankan kemampuan generalisasi yang baik (Sandler et al. 2018)

2.3.5.2. Inception ResNetV2

Inception-Resnet-V2 adalah variasi dari model Inception V3, dan ini jauh lebih dalam dari Inception V3 sebelumnya. Pada *block* konvolusi pada Inception Resnet-V2 memiliki jaringan yang sama di mana blok sisa yang berulang telah dikompresi. Kesuksesan *Residual Block* yang awal mula di ciptakan pada arsitektur resnet yang menjadi juara LSVRC 2015 (*ImageNet Large Scale Visual Recognition Challenge*) yan mengatasi *Vanishing Gradient* yang terjadi karena banyaknya layer konvolusi. Kelebihannya adalah penggunaan *residual connections*, yaitu jalur pintas (*shortcut*) yang memungkinkan *gradien* mengalir lebih baik melalui banyak lapisan. Pendekatan ini membuat ResNetV2 dapat dilatih dengan kedalaman ratusan bahkan ribuan lapisan tanpa mengalami degradasi performa. Selain itu, ResNetV2 melakukan normalisasi *batch* di awal setiap blok residual, yang meningkatkan stabilitas pelatihan dan mempercepat konvergensi. Secara teoretis, desain ini membuat ResNetV2 lebih unggul dalam mengekstraksi fitur yang kompleks dan hierarkis, menjadikannya pilihan ideal untuk tugas pengenalan wajah dengan variasi data yang tinggi (Nugraha, Komarudin, and Ramadhan 2022).

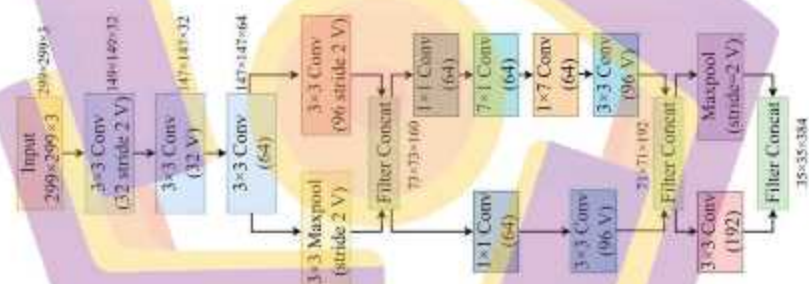
Oleh karena itu di perlukan koneksi Residual melatih arsitektur yang sangat dalam. Inception ResNetV2 adalah arsitektur CNN yang dibangun di atas keluarga

arsitektur *Inception* tetapi menggabungkan koneksi *Residual* (menggantikan tahap rangkaian filter dari arsitektur *Inception*). Inception-ResNet-V2 mendapat semua manfaat dari pendekatan residual sambil mempertahankan efisiensi komputasinya, berikut skema Inception Resnet-V2 dapat dilihat pada gambar 2.10.



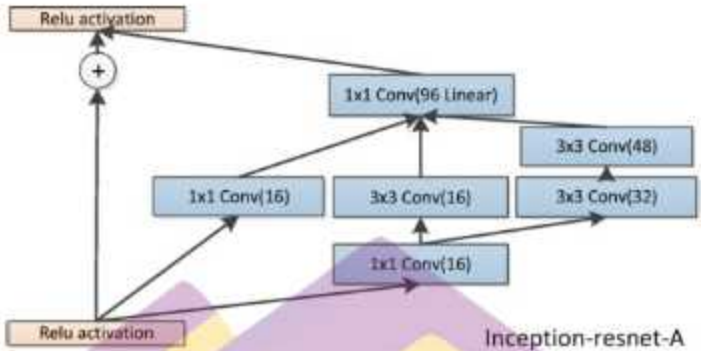
Gambar 2. 10. Skema Inception ResNetV2

Pada Stem blok ini berguna untuk memberikan inisial set yang akan dilakukan sebelum masuk ke dalam modul Block Inception selanjutnya, *Stem Block* dapat dilihat pada gambar 2.11.

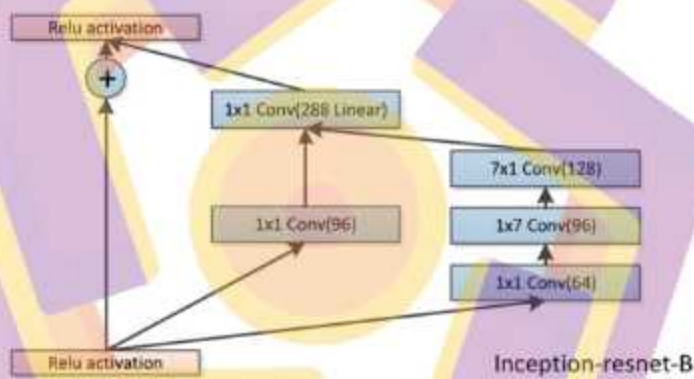


Gambar 2. 11. *Stem Block*

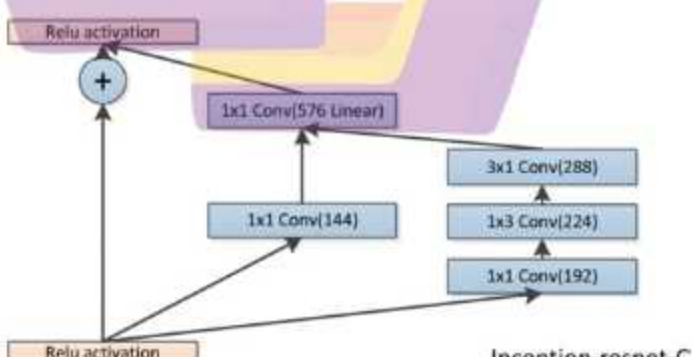
Pada *Inception Resnet Block* memperkenalkan *Residual Connection* yang dibangun di atas keluarga arsitektur *Inception* tetapi menggabungkan koneksi residual menggantikan tahap rangkaian filter dari arsitektur *Inception* yang terdiri modul A, B dan C, arsitektur *Inception Resnet Block* A,B dan C dapat dilihat pada gambar 2.12, 2.13 dan 2.14.



Gambar 2. 12. *Inception Resnet* A

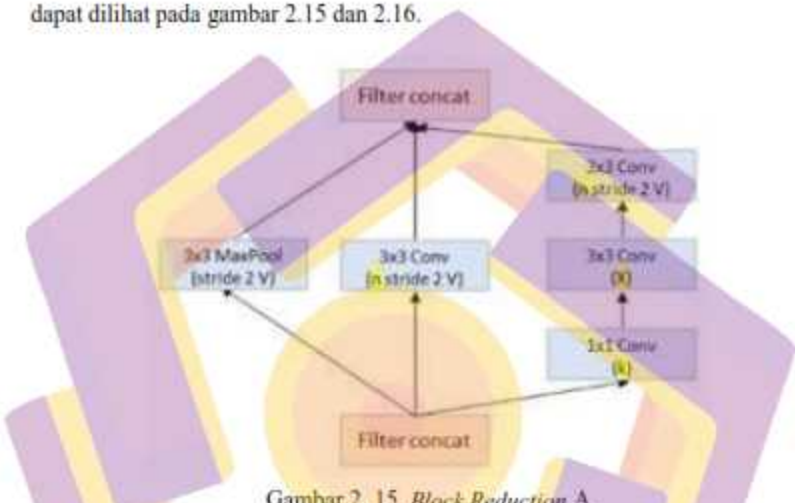


Gambar 2. 13. *Inception Resnet* B

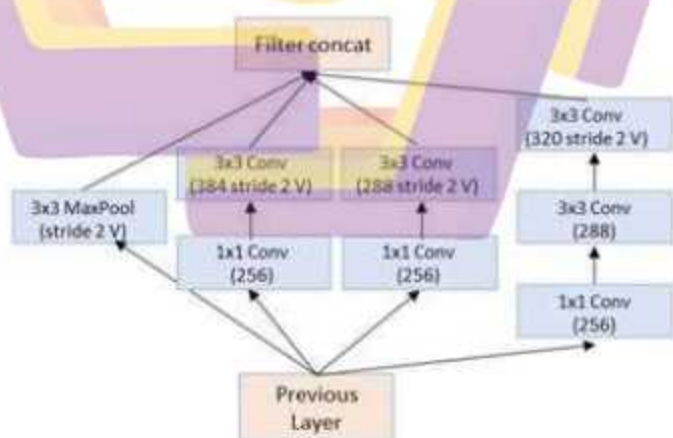


Gambar 2. 14. *Inception Resnet* C

Tahap selanjutnya ekstraksi fitur masuk kedalam *reduction block*, pada *block* ini akan melakukan reduksi atau pengecilan ukuran fitur dengan melakukan *maxpooling* pada setiap *block reduction* dengan di ikuti dengan pengabungan fitur atau *Concatenate* yang terdiri dari *Reduction* A dan B. *Block Reduction* A dan B dapat dilihat pada gambar 2.15 dan 2.16.



Gambar 2. 15. *Block Reduction* A

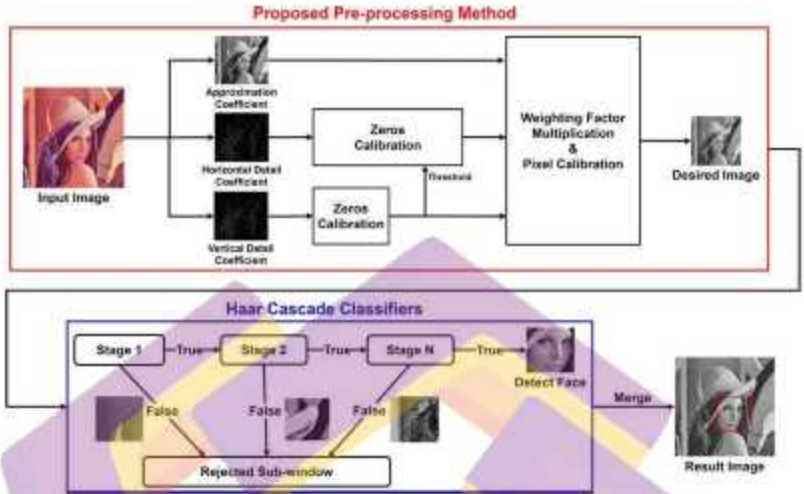


Gambar 2. 16. *Block Reduction* B

2.3.6. *Haar Cascade*

Haar Cascade adalah sebuah metode deteksi objek yang dibuat oleh Paul Viola dan Michael Jones. Pada tahun 2001, mereka mempresentasikan makalah yang disebut "*Rapid Object Detection using a Boosted Cascade of Simple*" (Anarki, Auliasari, and Orisa 2021). *Haar Cascade* adalah kumpulan fungsi *Haar-Like* yang digabungkan untuk membentuk pengklasifikasi. Fiturnya adalah jumlah nilai piksel putih yang dikurangkan dari nilai piksel pada area hitam.

Fungsi *Haar-like-feature* atau biasa disebut juga dengan *Haar cascade classifier* adalah fungsi persegi panjang (persegi) yang memberikan indikasi spesifik pada gambar. Pengklasifikasi *Haar Cascade* berasal dari gabungan piksel hitam dan piksel putih yang membentuk kotak. Kata *Haar* sendiri dinyatakan sebagai fungsi matematika dalam bentuk kotak (Wavelet Hhaar). Awalnya, pengolahan citra hanya didasarkan pada nilai RGB dari setiap piksel, tetapi cara pengolahan seperti itu tidak efektif. Kemudian, Viola dan Jones mengembangkan dan membentuk fungsi *Haar-Like*. Fitur *Haar-like* menangani gambar dalam kotak, di mana ada beberapa piksel dalam satu bingkai. Kemudian proses setiap kotak dan hasilkan nilai yang berbeda untuk menunjukkan area gelap dan terang. Nilai-nilai ini akan digunakan sebagai dasar untuk pengolahan citra. Berikut ilustrasi deteksi wajah menggunakan *Haar Cascade* terlihat pada gambar 2.17.



Gambar 2. 17. Proses deteksi wajah dengan *Haar Cascade*

2.3.7. Pengelompokan Data

Agar dapar berfungsi dengan baik, *Convolutional Neural Networks* memerlukan *dataset* yang besar yang berisi berbagai macam kondisi dan variasi (Garcia and Barbedo 2018). Menurut (Putra 2020) pengelompokan data sangat diperlikan untuk proses *training* dan *testing* dalam membangun sebuah model Pembelajaran Mesin atau *Deep Learning*. *Training* merupakan proses konstruksi model yang membutuhkan *training set* (data trening) yang merupakan himpunan data yang digunakan untuk melatih atau membangun model dan *validation set* (data validasi) yang merupakan himpunan data untuk mengoptimisasi saat melatih model. *Testing* merupakan proses menguji kinerja model pembelajaran yang membutuhkan *training set* (data uji) yang merupakan himpunan data untuk menguji model setelah proses latihan.

Training, validation dan *testing data* harus memeliki karakteristik yang sama dan sebaiknya didistribusikan dengan jumlah yang seimbang 50% kasus positif dan 50% kasus negatif. Rasio pembagian *dataset* (*training: validation: testing*) pada umumnya dibagi menjadi (80%: 10%: 10%) atau (90%: 5%: 5%) dan apabila *dataset* berukuran kecil, dapat dibagi menjad *training* dan *testing* dengan rasio (90%: 10%), (80%: 20%), (70%: 30%) atau (50%: 50%).

2.3.8. *Tensorflow*

Framework Tensorflow dikembangkan oleh *Google Brain Team* pada tahun 2015 untuk perhitungan numerik, dan sekarang banyak digunakan oleh perusahaan besar untuk pengembangan aplikasi AI, seperti klasifikasi citra, penyematan kata, dan pengembangan *chatbot*. *Tensorflow* menyediakan antarmuka yang dapat mengekspresikan algoritma *Machine Learning* dan aplikasi untuk mengeksekusi algoritma. *Tensorflow* mendukung pemodelan *Recurrent Neural Network* (RNN), *Restricted Boltzmann Machine* (RBM), *Convolutional Neural Network* (CNN) dan *Dynamic Bayesian Network* (DBN), serta eksekusi paralel. Nama *Tensorflow* didasarkan pada bagaimana tensor di seluruh jaringan. Tensor merupakan jenis array multidimensi (Wiranda, Purba, and Sukmawati 2020).

2.3.9. *Confusion Matrix*

Confusion matrix merupakan sebuah tabel yang terdiri dari atas banyaknya baris data uji yang diprediksi benar dan tidak benar oleh model klasifikasi, tabel ini diperlukan untuk menentukan kinerja suatu model klasifikasi (Wijayanto 2015). Berikut merupakan tabel dari sebuah *confusion matrix*:

Tabel 2.2. *Confusion matrix*

		Kelas Prediksi	
		Positive	*Negative*
Hasil Prediksi	*Positive*	*TP (True Positive)*	*FP (False Positive)*
	Negative	*FN (False Negative)*	*TN (True Negative)*

True Positive (TP) merupakan data yang diprediksi dengan tepat sebagai keluaran positive atau benar, *True Negative* (TN) merupakan data yang diprediksi tepat sebagai keluaran negatif atau salah, *False Positive* (FP) merupakan data prediksi dengan kurang tepat apabila keluaran berupa positif atau benar dan *False Negative* (FN) merupakan data yang diprediksi kurang tepat.

Untuk mengukur kinerja model sebuah *matrix* dari *confusion matrix*, ada beberapa model kinerja yang biasanya digunakan yaitu, *accuracy*, *precission* dan *specificity*.

a. *Accuracy*

Accuracy akan menghitung seberapa tepat dan akurat sebuah model atau arsitektur mengklasifikasi dengan benar.

(5)

Persamaan (5) merupakan rasio prediksi benar (positif dan negatif) dari keseluruhan data. Dapat dikatakan bahwa akurasi adala tingkat kedekatan nilai prediksi dengan nilai sebenarnya.

b. *Precission*

Precission merupakan gambaran dari tingkat keakuratan data permintaan dengan hasil prediksi yang diberikan model.

(6)

Persamaan (6) merupakan rasio *Precission* benar positif dibandingkan dengan keseluruhan hasil yang telah diprediksi positif.

c. *Specificity*

Specificity merupakan gambaran perhtingan kebenaran memprediksi negatif dibandingkan keseluruhan data negatif

(7)

BAB III

METODE PENELITIAN

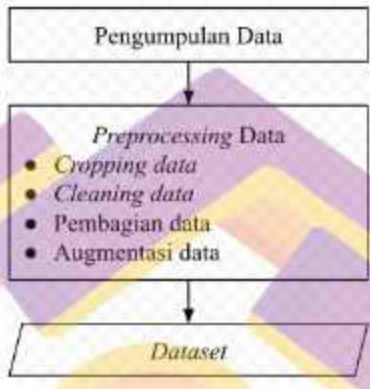
3.1. Jenis, Sifat, dan Pendekatan Penelitian

Pendekatan yang dilakukan pada penelitian ini bersifat kuantitatif dengan jenis penelitian eksperimental, dimana penelitian ini melakukan skenario pengujian dengan membandingkan beberapa arsitektur dari algoritma *convolutional neural network* (CNN) sebagai model yang berfungsi untuk pengenalan wajah. Skenario percobaan kemudian akan dievaluasi untuk menilai keakuratan dalam mendeteksi dan mengklasifikasikan citra dari setiap skenario untuk diketahui fakta-fakta dari hasil penelitian.

3.2. Metode Pengumpulan Data

Metode pengumpulan data dalam penelitian ini melibatkan serangkaian tahapan yang sistematis untuk memastikan kualitas dan kelayakan data. Pertama, Pengumpulan data yang bertujusn untuk mendapatkan gambar sebagai data mentah yang nantinya akan diolah untuk menjadi dataset. Kedua, *preprocessing* data dilakukan untuk menyiapkan data mentah dengan format yang konsisten. Tahapan *preprocessing* data seperti *cropping* data untuk memfokuskan pada area yang relevan dengan menghilangkan bagian yang tidak diperlukan, *cleaning* data bertujuan menghapus noise atau kesalahan yang mungkin mengganggu analisis, pembagian data dilakukan untuk memisahkan data menjadi *training* dan *testing* set guna evaluasi model yang akurat dan augmentasi data diterapkan untuk

memperbanyak variasi data terutama jika jumlah data terbatas dengan teknik seperti rotasi atau flipping. Untuk lebih jelas alur pengumpulan data dapat dilihat pada gambar 3.1.



Gambar 3. 1 Alur pengumpulan data

Dataset yang dihasilkan dari proses ini akan menjadi dasar untuk pengembangan model yang diharapkan. Setiap tahap dirancang untuk memastikan data yang digunakan representatif dan siap untuk analisis lebih lanjut.

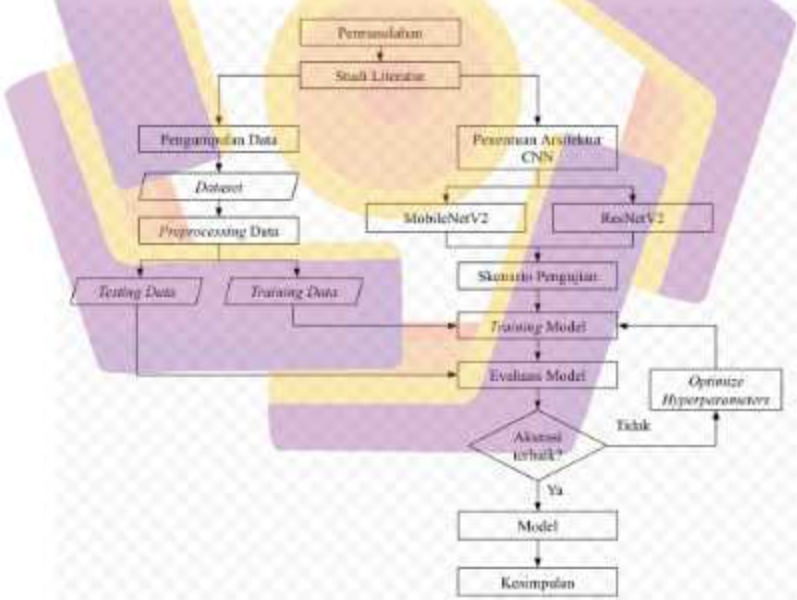
3.3. Metode Analisis Data

Metode analisis yang digunakan dalam penelitian ini adalah analisis kuantitatif menggunakan algoritma *Convolutional Neural Network* (CNN) dengan membandigkan kinerja dari beberapa arsitektur. Metode analisis data dalam penelitian ini dirancang untuk mengevaluasi secara komprehensif kinerja model CNN dalam mengidentifikasi wajah. Analisis eksperimental dilakukan dengan membuat skenario pelatihan dimana setiap skenario diberikan tindakan yang berbeda seperti penentuan arsitektur, penerapan tektik *fine-tuning* atau *retrain*

lapisan arsitektur tertentu. Selanjutnya, analisis kuantitatif menggunakan metrik standar seperti *accuracy*, *precision*, dan *recall* untuk mengukur performa model secara objektif. *Accuracy* digunakan untuk menilai persentase prediksi benar secara keseluruhan, sementara *precision* dan *recall* fokus pada kemampuan model dalam mengidentifikasi wajah secara akurat dan menghindari *false positive* atau *false negative*.

3.4. Alur Penelitian

Untuk lebih memahami alur yang dilakukan pada penelitian ini, dapat dilihat pada gambar 3.2.



Gambar 3. 2. Alur Penelitian

Alur penelitian secara sistematis dapat dilihat pada gambar 3.2 dan akan dijelaskan sebagai berikut:

a. Permasalahan

Tahap ini merupakan tahap awal untuk memulai sebuah penelitan dengan melihat permasalahan yang berada di tengah masyarakat.

b. Studi Literatur

Setelah menentukan permasalah, proses selanjutnya adalah mencari informasi mengenai hal yang berhubungan dengan masalah tersebut melalui studi literatur. Studi literartur dilakukan dengan membaca jurnal penelitian dan buku yang diaggap relevan dengan permasalahan yang akan diangkat. Proses ini juga sebagai bahan rujukan penelitian untuk memilih metode atau algoritma yang dianggap sesuai dengan permasalahan.

c. Penentukan algoritma

Dari studi literatur, penulis telah menentukan algoritma dan arsitektur yang digunakan untuk mengenali wajah menggunakan algoritma *Convolutional Neural Network* (CNN) menggunakan arsitektur MobileNetV2 dan ResNerV2 pada penelitian ini.

d. Skenario

Setelah mentukan algoritma dan arsitektur, diperlukan skenario untuk penelitian yang akan dilakukan. Dari proses ini ditentukan empat skenario yang akan dilakukan. Skenario pertama, menggunakan arsitektur MobileNetV2 yang memiliki 154 *layers* ditambah dengan 4 *layers custom* hasil *fine-tuning*. Proses training hanya dilakukan untuk 4 layers *custom* dengan menggunakan 100 *epoch*

dan ukuran gambar 224 x 224 *pixel*. Skenario kedua, menggunakan arsitektur MobileNetV2 seperti pada skenario pertama tetapi proses *training* dilakukan untuk 50 *layers* akhir dan 4 *layers custom*. Skenario ketiga, menggunakan arsitektur ResNetV2 yang memiliki 780 *layers* ditambah dengan 4 *layers custom hasil fine-tuning*. Proses training hanya dilakukan untuk 4 *layers custom* dengan menggunakan 100 *epoch* dan ukuran gambar 160 x 160 *pixel*. Skenario keempat, menggunakan arsitektur ResNetV2 seperti pada skenario ketiga tetapi proses *training* dilakukan untuk 50 *layers* akhir dan 4 *layers custom*.

e. Pengumpulan dataset

Machine Learning membutuhkan data untuk memperoleh kecerdasan, sehingga proses pengumpulan data sangat penting pada penelitian ini. Data diperoleh dari media internet yang berupa gambar yang telah melalui proses pelabelan. Akan tetapi, gambar tersebut masi memerlukan *preprocessing* terlebih dahulu.

f. Dataset

Data yang telah dikumpulkan dari tahap pengumpulan dataset akan diolah dengan cara melabelkan sesuai dengan informasi citra. Data selanjutnya dikelompokkan kedalam kedalam folder sesuai dengan kelasnya.

g. Training Data dan Model

Setelah menentukan skenario, peneliti mulai melakukan percoban menggunakan dataset yang telah disediakan sebelumnya. *Training* data ini juga bertujuan untuk mencari yang mana dari keempat skenario ini yang mempunyai akurasi atau hasil yang terbaik untuk direkomendasikan.

h. Evaluasi

Model yang telah terbentuk dari proses training, akan di evaluasi dengan menggunakan data testing untuk mendapatkan nilai akurasi menggunakan *Confusion Matrix*.

i. Kesimpulan

Setelah melakukan proses evaluasi dari skenario yang telah ditentukan sebelumnya, proses ini akan menyajikan hasil dari penelitian. Hasil penelitian berupa data fakta yang dihasilkan oleh akurasi dari *confusion matrix* terkait dengan Arsitektur dari algoritma.

BAB IV

HASIL PENELITIAN DAN PEMBAHASAN

4.1. Membangun *Dataset*

4.1.1. Pengumpulan Data

Kekhawatiran tentang sistem pengenalan wajah mencakup privasi, keamanan data, dan efek psikologis dan etika (Razaqa et al. 2024) (Rambe and Abdurrahman 2024). Untuk membangun model pengenalan wajah akan menggunakan data publik sebagai dataset. Dataset yang digunakan merupakan gambar aktor *Hollywood* yang diperolah dari *Kaggle* sebagai platform online yang menyediakan sumber daya untuk data *science* dan *machine learning* dengan link https://www.kaggle.com/datasets/vishesh1412/celebrity-face-image-dataset. dataset ini telah digunakan pada penelitian sebelumnya oleh (Kumar and Kumar 2025) Sampel dataset yang digunakan pada penelitian ini dapat dilihat pada gamber 4.1

Gambar 4.1 Contoh dataset

Dataset yang digunakan adalah 16 aktor yang dan masing-masing aktor memiligi 100 gambar sehingga total gambar wajah yang digunakan pada penelitian ini adalah 1600 gambar.

4.1.2. *Preprocessing* Data

4.1.2.1. *Cropping Data*

Setelah mendapatkan dataset, selanjutnya dilakukan proses *cropping* data untuk mendapatkan bagian wajah. Proses *cropping* dilakukan secara otomatis dengan menggunakan *Haar Cascade* yang merupakan metode untuk mengidentifikasi objek, terutama wajah yang diciptakan oleh Paul Viola dan Michael Jones (Anarki, Auliasari, and Orisa 2021). Sebagai ilustrasi proses *cropping* diperlihatkan pada Gambar 4.1.

Gambar 4. 2. Proses *cropping*

Pada langkah pertama, peneliti mengimpor berbagai *library* yang akan digunakan dalam proses deteksi dan *cropping* wajah. *Library* utama yang dibutuhkan adalah OpenCV (cv2), yang menyediakan fungsi-fungsi untuk pengolahan citra dan *computer vision*, termasuk deteksi wajah menggunakan *Haar Cascade*. Selain itu, *library* OS untuk berinteraksi dengan sistem file. *Import library* ini merupakan fondasi awal karena tanpa library tersebut, kode tidak dapat berjalan dengan baik. Kode untuk mengimport *library* adalah sebagai berikut:

```
# Step 1: Import Libraries
import cv2
import os
```

Langkah kedua melibatkan *mounting Google Drive* ke lingkungan *Google Colab* agar kita dapat mengakses dataset gambar yang disimpan di dalamnya. Proses *mounting* dilakukan dengan menggunakan modul *google.colab.drive*, yang memungkinkan kita untuk mengautentikasi dan mengaitkan *Google Drive* dengan *notebook Colab*. Kode untuk melakukan *mounting* adalah sebagai berikut:

```
# Step 2: Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')
```

Setelah mounting selesai, kita perlu memeriksa apakah folder *output* untuk menyimpan hasil *cropping* sudah ada. Jika folder tersebut belum ada, kita akan membuatnya menggunakan modul OS. Hal ini penting untuk memastikan bahwa hasil pemrosesan gambar dapat disimpan dengan rapi dan terorganisir. Tanpa folder *output*, file hasil *cropping* mungkin akan tersebar atau bahkan tidak tersimpan sama sekali, sehingga langkah ini sangat krusial untuk alur kerja yang efisien menggunakan kode berikut:

```
# Step 3: Set Input and Output Directory
input_dir =
'/content/drive/MyDrive/COLAB/presensi_wajah_02/Celebrity
Faces Dataset/' # Folder utama dengan subfolder gambar asli
output_dir =
'/content/drive/MyDrive/COLAB/presensi_wajah_02/cropped_face
s_05/' # Folder utama untuk menyimpan gambar crop

# Membuat folder output jika belum ada
if not os.path.exists(output_dir):
    os.makedirs(output_dir)
```

Langkah selanjutnya adalah memuat model *Haar Cascade* yang telah dilatih sebelumnya untuk mendeteksi wajah. Model ini disediakan dalam file XML yang berisi parameter-parameter untuk mengenali pola wajah. Di *OpenCV*, model ini dapat dimuat menggunakan fungsi *cv2.CascadeClassifier()*. File XML yang digunakan adalah *haarcascade_frontalface_default.xml*, yang khusus dirancang untuk mendeteksi wajah dari depan. Memuat model ini adalah langkah penting karena model ini akan menjadi dasar untuk mendeteksi wajah dalam gambar. Kode untuk memuat model *Haar Cascade* adalah sebagai berikut:

```
# Step 4: Load Haar Cascade Model untuk Deteksi Wajah
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
```

Setelah mendapatkan model *Haar Cascade*, selanjutnya membuat fungsi yang bertugas untuk mendeteksi wajah dalam gambar dan melakukan *cropping* dengan *bounding box*. Fungsi ini juga akan menangani kasus di mana wajah tidak terdeteksi atau gambar tidak valid. Dengan membuat fungsi ini, kita dapat memproses banyak gambar secara otomatis tanpa harus menulis ulang kode untuk setiap gambar. Kode fungsinya adalah sebagai berikut:

```
# Step 5: Fungsi untuk Mendeteksi dan Crop Wajah dengan
Bounding Box
def crop_faces_with_padding(input_dir, output_dir,
padding=0.2):
    for root, dirs, files in os.walk(input_dir):
        # Menjaga struktur folder asli di folder output
        relative_path = os.path.relpath(root, input_dir)
        output_subdir = os.path.join(output_dir,
                                     relative_path)
        if not os.path.exists(output_subdir):
            os.makedirs(output_subdir)

        for filename in files:
            if filename.lower().endswith(('.jpg', '.jpeg',
'.png')):
                # Baca gambar
                img_path = os.path.join(root, filename)
                img = cv2.imread(img_path)
                height, width, _ = img.shape

                # Konversi ke grayscale untuk deteksi wajah
                gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
                faces = face_cascade.detectMultiScale(gray,
scaleFactor=1.1, minNeighbors=5, minSize=(224, 224))

                # Proses setiap wajah yang terdeteksi
                for i, (x, y, w, h) in enumerate(faces):
                    # Tambahkan padding ke bounding box
                    x_padding = int(padding * w)
                    y_padding = int(padding * h)
                    x1 = max(x - x_padding, 0)
                    y1 = max(y - y_padding, 0)
                    x2 = min(x + w + x_padding, width)
                    y2 = min(y + h + y_padding, height)

                    # Crop wajah dengan bounding box yang
                    cropped_face = img[y1:y2, x1:x2]

                    # Simpan hasil crop
                    cropped_filename =
f"{os.path.splitext(filename)[0]}_face{i+1}.jpg"
                    cropped_path =
os.path.join(output_subdir, cropped_filename)
                    cv2.imwrite(cropped_path, cropped_face)

                print(f"Processed {img_path} - {len(faces)}
                      face(s) detected.")
```

Langkah terakhir adalah menjalankan fungsi *cropping* yang telah dibuat pada gambar-gambar yang ada di direktori input. Proses ini membaca setiap

gambar, menerapkan fungsi deteksi dan *cropping*, lalu menyimpan hasilnya ke folder *output*. Proses ini melibatkan *loop* melalui semua file dalam direktori dan memastikan bahwa hanya file gambar yang diproses. Setelah *cropping* selesai, peneliti dapat memeriksa hasilnya untuk memastikan bahwa wajah telah terdeteksi dengan benar dan bounding box. Kode untuk menjalankan fungsi *cropping* adalah sebagai berikut:

```
# Step 6: Jalankan Fungsi dengan Padding
crop_faces_with_padding(input_dir, output_dir, padding=0)  # 
Padding 0%

print("Proses selesai! Wajah yang dicrop disimpan di 
folder:", output_dir)
```

Langkah ini adalah puncak dari seluruh proses, di mana semua persiapan dan fungsi yang telah dibuat sebelumnya dijalankan untuk menghasilkan *output* yang diharapkan. Dengan menyelesaikan langkah ini membuat *pipeline* otomatis untuk deteksi dan *cropping* wajah menggunakan *Haar Cascade* di *Python*. Contoh hasil *cropping* dapat dilihat pada gambar 4.3 berikut.

Gambar 4.3 Contoh hasil *cropping*

Proses *cropping* yang dilakukan secera otomatis menggunakan *haar cascade* mengakibatkan penambahan pada jumlah dataset dikarenakan terdapat beberapa sumber gambar yang dideteksi lebih dari satu wajah. Sebagai ilustrasi dapat dilihat pada gambar 4.4 berikut.

Gambar 4. 4. *Cropping* lebih dari satu wajah

Untuk menghilangkan cinta yang tidak dibubutuhkan, peneliti perlu untuk melakukan *cleaning* data.

4.1.2.2. *Cleaning Data*

Untuk mengatasi ketidakkonsistenan data atau data yang tidak valid, diperlukan tahapan pembersihan data (*data cleaning*) yang sistematis (Varkarakis and Corcoran 2020). Setelah proses *cropping* wajah dilakukan, data yang dihasilkan sering kali masih mengandung *noise* seperti gambar buram, pencahayaan yang terlalu ekstrem, wajah yang terpotong sebagian, maupun gambar yang tidak sesuai dengan kategori target. Oleh karena itu, pembersihan data dilanjutkan dengan

seleksi manual untuk menghapus citra berkualitas rendah, citra yang terlalu muda atau tidak relevan, serta duplikasi data yang berpotensi menyebabkan bias dalam pelatihan model. Proses ini tidak hanya menjaga konsistensi kualitas dataset, tetapi juga memastikan bahwa model CNN tidak belajar dari data yang menyesatkan. Tujuan utamanya adalah meningkatkan reliabilitas model, memperkuat kemampuan generalisasi pada data baru, serta mengurangi risiko *overfitting*. Seihingga, pembersihan data *pasca-cropping* merupakan tahap krusial untuk memastikan validitas hasil eksperimen pengenalan wajah. Berikut ilustrasi diperlihatkan pada Gambar 4.5.

Gambar 4. 5. Pembersihan data

Sebelumnya julah dataset sebanyak 1600 gambar yang terbagi kedalam 16 kelas dan masing-masing kela sebanyak 100 gambar, setelah melalui proses *cropping* dan *cleaning* data jumlah data menjadi lebih sedikit. Transfomasi jumlah dataset dapat dilihat pada tabel 4.1 berikut.

Tabel 4.1 Transformasi jumlah dataset

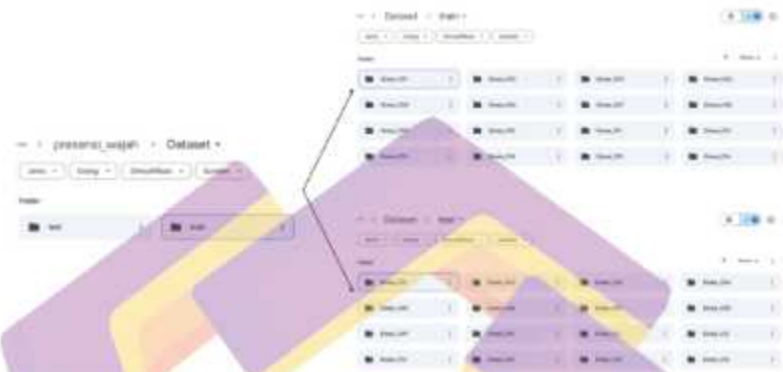
No.	Klas	Jumlah data	Setelah Proses *Cropping*	*Cleaning* data	Jumlah Data
1	Siswa 1	100	111	26	85
2	Siswa 2	100	113	34	79
3	Siswa 3	100	109	29	80
4	Siswa 4	100	108	30	78
5	Siswa 5	100	124	44	80
6	Siswa 6	100	111	33	78
7	Siswa 7	100	117	39	78
8	Siswa 8	100	112	42	70
9	Siswa 9	100	125	47	78
10	Siswa 10	100	114	29	85
11	Siswa 11	100	128	47	81
12	Siswa 12	100	116	39	77
13	Siswa 13	100	120	43	77
14	Siswa 14	100	114	30	84
15	Siswa 15	100	108	32	76
16	Siswa 16	100	105	25	80
Jumlah		**1600**	**1835**	**569**	**1266**

Setelah melakukan pembersihan pada 569 gambar, jumlah data menjadi 1266 gambar dan masing-masing kelas memiliki jumlah yang bervariasi. Diketahui, kelas dengan jumlah data terbesar adalah 85 gambar dan data terkecil adalah 70 gamba sehingga untuk menyamakan jumlah data pada setiap kelas digunakan 70 gambar untuk proses selanjutnya.

4.1.2.3. Pembagian Data

Setelah melakukan pembersihan terhadap *dataset* selanjutnya dilakukan proses pembagian data. Proses ini bertujuan untuk membagi dataset menjadi dua yatiu data latih dan data uji. dataset dibagi menjadi 50 data latih dan 20 data uji

yang disimpan kedalam *google drive*. Berikut contoh pembagian data diperlihatkan pada Gambar 4.6.



Gambar 4. 6. Pengelompokan data

4.1.2.4. Augmentasi Data

Tujuan dari augmentasi data adalah untuk memperluas variasi data dan memperluas generalisasi model *deep learning* (DLY et al. 2023). Pada penelitian ini, augmentasi dilakukan secara otomatis menggunakan *tensorflow* yang merupakan *library python* yang dijalankan di *google colaboratory* dengan kode berikut:

```
# Data augmentation
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest',
)

test_datagen = ImageDataGenerator(rescale=1./255)
```

Augmentasi yang dilakukan meliputi "*rotation_range*=30" untuk memutar gambar secara acak dalam rentang 30 derajat, "*width_shift_range*=0.2" untuk menggeser gambar secara horizontal hingga 20% dari lebar gamba, "*height_shift_range*=0.2" untuk menggeser gambar secara vertikal hingga 20% dari tinggi gambar, "*shear_range*=0.2" untuk menerapkan transformasi *shear* (geser sudut) hingga 20%, "*zoom_range*=0.2" untuk memperbesar atau memperkecil gambar hingga 20%, "*horizontal_flip*=True" untuk membalik gambar secara horizontal secara acak, dan yang terakhir "*fill_mode='nearest'*" untuk mengisi area kosong dengan *pixel* terdekat akibat augmentasi sebelumnya. Jika divisualisasikan, maka hasil augmetasi dapat dilihat pada gambar 4.7 berikut.

Gambar 4.7 Visualisai hasil augmentasi

Satu gambar dapat menghasilkan gambra yang bervariasi tetapi secara teknis proses augmentasi tidak menambah jumlah dataset tetapi setiap kali model mengambil satu *batch* gambar pada proses *training*, *ImageDataGenerator* akan mengambil gambar asli lalu mengubahnya secara acak sesuai dengan aturan augmentasi sehingga setiap *epoch*, model akan melihat versi berbeda dari gambar yang sama.

Sementara itu, data uji "*test_data*" diproses dengan cara yang lebih sederhana karena augmentasi tidak diperlukan pada tahap evaluasi. *test_datagen* hanya melakukan normalisasi piksel "*rescale=1./255*" tanpa augmentasi, karena tujuan data uji adalah untuk mengukur performa model pada gambar yang belum pernah dilihat sebelumnya.

4.2. Analisis Data

Proses analisa terhadap data diawali dengan menentukan skenario percobaan yang bertujuan untuk mendapatkan model klasifikasi yang terbaik. Dalam melakukan percobaan pada setiap skenario, *dataset* yang telah dikelompokan akan dijadikan masukan dimana data latih akan digunakan untuk proses pembangunan model dan data uji digunakan untk menguji model yang telah dibangun. Setiap skenario akan diukur dan dilakukan perbandingan untuk dijadikan sebagai hasil penelitian.

4.2.1. Skenario Pengujian

Ekperimen yang dilakukan oleh (Shukla and Tiwari 2023) dengan arsitektur VGG16, VGG19, ResNet50, ResNet101 dan MobileNetV2, menyimpilkan bahwa arsitektur MobileNetV2 menghasilkan akurasi terbaik untuk pengenalan wajah pengguna masker dengan akurasi mencapai 99,82%. Eksperimen yang dilakukan oleh (Ai et al. 2022) dengan arsitektur AlexNet, VGG16, MobileNet V2, InceptionV2, dan ResNetV2 menyimpulkan bahwa arsitektur ResNetV2 dan MobileNetV2 menghasilkan akurasi yang cukup baik dalam melakukan deteksi wajah dengan akurasi ResNetV2 90,49% dan MobileNetV2 89,18%.

Berdasarkan hasil penelitain terdahulu peneliti dalam upayah untuk mendapatkan model dengan akurasi terbaik, dilakukan beberapa skenario pengujian dengan membandingkan dua arsitektur yaitu MobileNetV2 dan ResNetV2. *Fine-tuning* dilakukan pada masing-masing arsitektur untuk melakukan klasifikasi sesui dengan jumlah wajah yang dilatih. Proses *fine-tuning* dilakukan dengan dua skenario yang pertama hanya melakukan *training* pada 4 *layers custom* tanpa

melakukan *retrain* ulang *layer* arsitektur MobileNetV2 dan ResNetV2. Yang kedua melakukan *retrain* ulang pada 50 *layers* terakhir arsitektur MobileNetV2 dan ResNetV2 beserta 4 *layers custom*. Skenario pengujian diperlihatkan pada tabel 4.2.

Tabel 4. 2. Skenario pengujian

Skenario	Arsitektur	Retrain arsitektur	Epoch
S1	MobileNetV2	False	100
S2	MobileNetV2	-50	100
S3	ResNetV2	Flase	100
S4	ResNetV2	-50	100

Stelah melalui proses di atas, selanjutnya dilakukan proses training dan testing sesui dengan skenario pengujian. Setip skenario menggunakan 100 epoch, 50 data latih dan 20 data uji. Hasil pengujian akan dievalusi menggunakan confution matrix. Confusion matrix digunakan untuk memvisualisasikan hasil penilaian kinerja model klasifikasi dengan menampilkan hasil kelas masing-masing dan membandingkannya dengan nilai yang sebenarnya (Rahim and Kusrini 2021).

4.2.2. *Training* dan *Testing* Data

Setelah *dataset* dan arsitektur telah siap untuk digunakan, tahap selanjutnya pada penelitian ini adalah melakukan *training* dengan menggunakan data latih sebagai masukan untuk menghasilkan sebuah model sesui dengan skenario pengujian. Selanjutnya, model yang telah dibagun akan diuji menggunakan data uji.

4.2.2.1. Skenario Satu

a. Menyiapkan dataset

Sebelum melakukan *training* dan *testing*, terlebih dahulu menyiapkan dataset yang telah melalui augmentasi. Menyiapkan dataset dilakukan dengan kode berikut:

```
# Data latih
train_data = train_datagen.flow_from_directory(
    train_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    shuffle=True,
)

# Data Uji
test_data = test_datagen.flow_from_directory(
    test_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    shuffle=False,
)
```

Data latih dimuat menggunakan fungsi *flow_from_directory*, di mana gambar diubah ukurannya menjadi 224x224 piksel (*target_size*), dikelompokkan dalam *batch* berisi 32 gambar (*batch_size*), dan labelnya dikonversi ke format *one-hot encoding* (*class_mode='categorical'*). Pengacakan data (*shuffle=True*) juga dilakukan untuk mencegah model menghafal urutan gambar. Gambar dalam data uji juga diubah ukurannya menjadi 224x224 piksel untuk konsistensi dengan data latih. Namun, berbeda dengan data latih, data uji tidak diacak (*shuffle=False*) agar urutan gambar tetap konsisten selama evaluasi, yang memudahkan analisis hasil.

b. Menyiapkan arsitektur

Untuk menerapkan *transfer learning* sesuai dengan skenario pengujian, perlu untuk meyiapkan arsitektur yang telah dilatih sebelumnya (*pre-trained*) menggunakan kode berikut:

```
# Load MobileNetV2 pre-trained model
base_model = tf.keras.applications.MobileNetV2(
    weights='imagenet', include_top=False, input_shape=(224,
224, 3)
```

Skenario satu menggunakan arsitektur MobileNetV2 dengan dataset *ImageNet* sebagai model dasar (*base model*) untuk tugas *transfer learning*. Model ini sudah dioptimalkan dari *ImageNet* (*weights='imagenet'*), sehingga memiliki kemampuan awal untuk mengekstrak fitur-fitur penting dari gambar. *Input shape* yang digunakan adalah (224, 224, 3), yang berarti model menerima gambar berukuran 224x224 piksel dengan 3 saluran warna (RGB). Parameter (*include_top=False*) berfungsi untuk menghilangkan lapisan terakhir (*fully connected layer*) dari model asli karena lapisan tersebut spesifik untuk klasifikasi 1000 kelas *ImageNet*. Kita ingin menyesuaikan model untuk tugas baru dengan jumlah kelas berbeda.

c. *Fine-Tuning*

Pada skenario satu, proses *fine-tuning* terhadap model dasar MobileNetV2 deilakukan dengan membekukan (*freeze*) seluruh *layer* terlebih dahulu menggunakan kode berikut:

```
# Unfreeze beberapa layer terakhir
for layer in base_model.layers:
    layer.trainable = False
```

Melalui perulangan *for layer* in, *base_model.layers* diberikan parameter "*layer.trainable = False*" dengan tujuan pembekuan *layer* untuk mempertahankan pengetahuan yang sudah dipelajari model dari *ImageNet* selama pelatihan awal, sekaligus mencegah pembaruan bobot yang tidak perlu pada lapisan awal.

Proses *fine-tuning* selanjutnya adalah membangun lapisan kustom di atas model dasar MobileNetV2 sehingga dapat digukan kembali sesuai dengan jumlah kelas dari dataset pada penelitian ini. Penambakan *layer* dilakukan dengan kode berikut:

```
from tensorflow.keras.models import Model

x = base_model.output
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dense(256, activation='relu')(x)
x = layers.Dropout(0.5)(x)
x = layers.Dense(train_data.num_classes,
activation='softmax')(x)  # Output sesuai jumlah wajah
model = Model(inputs=base_model.input, outputs=x)
```

Layer pertama adalah *GlobalAveragePooling2D* yang berfungsi untuk mengurangi dimensi fitur dengan mengambil rata-rata setiap *feature map*, sehingga menghasilkan vektor 1D yang lebih ringkas. Kedua, ditambahkan *layer Dense* dengan 256 *neuron* dan aktivasi *ReLU*. Ketiga, *layer Dropout* dengan *rate* 0.5. Terakhir, lapisan *output* dengan aktivasi *softmax* ditambahkan, dimana jumlah neuronnya disesuaikan dengan jumlah kelas pada data latih.

d. Kompilasi dan *Training*

Sebelum melakukan *training*, model perlu dikompilasi untuk menentukan cara model belajar dan cara mengevaluasi performanya. Proses kompilasi dilakukan dengan kode berikut:

```
# Compile model
model.compile(
    optimizer=Adam(learning_rate=0.0001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

Model dikompilasi dengan menyiapkan tiga komponen utama yaitu *optimizer*, *loss function*, dan *metrics*. *Optimizer Adam* dipilih dengan learning rate 0.0001 yang relatif kecil untuk memastikan pembaruan bobot yang stabil selama *fine-tuning*. Fungsi *loss categorical_crossentropy* digunakan karena ini adalah masalah klasifikasi multi-kelas dengan label *one-hot encoded*. Metrik *accuracy* dipantau untuk mengevaluasi performa model selama pelatihan dan validasi.

Setelah melakukan kompilasi, model siap untuk dilatih menggunakan data latih yang telah disiapkan sebelumnya. Kode untuk melatiha model adalah sebagai berikut:

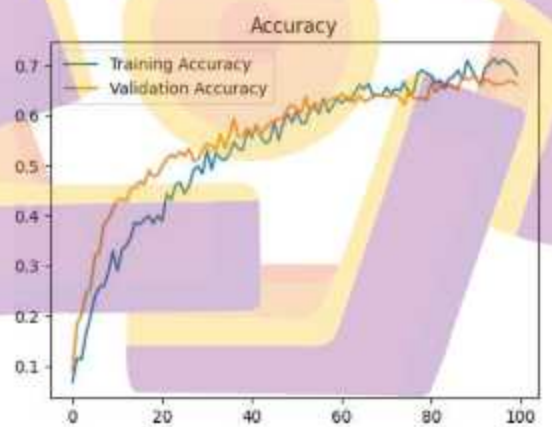
```
# Training CNN
history = model.fit(
    train_data,
    validation_data=test_data,
    epochs=100,
    verbose=1
)
```

Proses pelatihan (*training*) model *Convolutional Neural Network* (CNN) menggunakan *library Keras* dari *TensorFlow* di mana, fungsi *model.fit()*

bertanggung jawab untuk melatih model pada data yang telah dipersiapkan sebelumnya. Parameter utama yang digunakan meliputi *train_data* sebagai data latih yang dihasilkan dari *ImageDataGenerator*, *validation_data=test_data* untuk memantau performa model pada data uji selama pelatihan, dan *epochs*=100 yang menentukan jumlah iterasi lengkap melalui seluruh dataset.

e. Hasil *Training* dan *Testing*

Skenario satu menggunakan arsitektur MobileNetV2 yang memiliki 154 layers ditambah dengan 4 *layers custom* hasil *fine-tuning*. Proses *training* hanya dilakukan untuk 4 *layers custom* dengan menggunakan 100 *epoch* dan ukuran gambar 224 x 224 pixel. Hasil training diperlihatkan pada gambar 4.8



Gambar 4. 8. Proses *Training* Skenario Satu

Gambar 4.6 menunjukan akurasi *trainig* mencapai 68% dan akurasi validasi mencapai 66%. Selanjutnya hasil testing digambarkan dalam *confusion matrix* pada gambar 4.9 dan dijelaskan dengan *classification report* pada gambar 4.10.

Gambar 4. 9. *Confusion Matrix* Skenario Satu

Classification Report:

	precision	recall	f1-score	support
Siswa_001	1.00	0.55	0.71	20
Siswa_002	0.79	0.55	0.65	20
Siswa_003	0.71	1.00	0.83	20
Siswa_004	0.65	0.65	0.65	20
Siswa_005	0.61	0.55	0.58	20
Siswa_006	0.93	0.65	0.76	20
Siswa_007	0.56	0.90	0.69	20
Siswa_008	0.52	0.55	0.54	20
Siswa_009	0.53	0.95	0.68	20
Siswa_010	0.71	0.50	0.59	20
Siswa_011	0.61	0.70	0.65	20
Siswa_012	0.82	0.45	0.58	20
Siswa_013	0.87	0.65	0.74	20
Siswa_014	0.55	0.55	0.55	20
Siswa_015	0.50	0.75	0.60	20
Siswa_016	1.00	0.65	0.79	20
accuracy			0.66	320
macro avg	0.71	0.66	0.66	320
weighted avg	0.71	0.66	0.66	320

Gambar 4. 10. *Classification Report* Skenario Satu

Gambar 4.10 menunjukan hasil *testing* untuk model skenario satu mendapat nilai *accuracy* 66%, *precision* 71% dan *recall* 66%.

4.2.2.2. Skenario Dua

a. Menyiapkan dataset

Sebelum melakukan *training* dan *testing*, terlebih dahulu menyiapkan dataset yang telah melalui augmentasi. Menyiapkan dataset dilakukan dengan kode berikut:

```
# Data latih
train_data = train_datagen.flow_from_directory(
    train_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    shuffle=True,
)

# Data Uji
test_data = test_datagen.flow_from_directory(
    test_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    shuffle=False,
)
```

Data latih dimuat menggunakan fungsi *flow_from_directory*, di mana gambar diubah ukurannya menjadi 224x224 piksel (*target_size*), dikelompokkan dalam *batch* berisi 32 gambar (*batch_size*), dan labelnya dikonversi ke format *one-hot encoding* (*class_mode='categorical'*). Pengacakan data (*shuffle=True*) juga dilakukan untuk mencegah model menghafal urutan gambar. Gambar dalam data uji juga diubah ukurannya menjadi 224x224 piksel untuk konsistensi dengan data

latih. Namun, berbeda dengan data latih, data uji tidak diacak (*shuffle=False*) agar urutan gambar tetap konsisten selama evaluasi, yang memudahkan analisis hasil.

b. Menyiapkan arsitektur

Untuk menerapkan *transfer learning* sesuai dengan skenario pengujian, perlu untuk meyiapkan arsitektur yang telah dilatih sebelumnya (*pre-trained*) menggunakan kode berikut:

```
# Load MobileNetV2 pre-trained model
base_model = tf.keras.applications.MobileNetV2(
    weights='imagenet', include_top=False, input_shape=(224,
224, 3)
```

Skenario dua menggunakan arsitektur MobileNetV2 dengan dataset *ImageNet* sebagai model dasar (*base model*) untuk tugas *transfer learning*. Model ini sudah dioptimalkan dari *ImageNet* (*weights='imagenet'*), sehingga memiliki kemampuan awal untuk mengekstrak fitur-fitur penting dari gambar. *Input shape* yang digunakan adalah (224, 224, 3), yang berarti model menerima gambar berukuran 224x224 piksel dengan 3 saluran warna (RGB). Parameter (*include_top=False*) berfungsi untuk menghilangkan lapisan terakhir (*fully connected layer*) dari model asli karena lapisan tersebut spesifik untuk klasifikasi 1000 kelas *ImageNet*. Kita ingin menyesuaikan model untuk tugas baru dengan jumlah kelas berbeda.

c. *Fine-Tuning*

Pada skenario dua, proses *fine-tuning* terhadap model dasar MobileNetV2 deilakukan dengan mengaktifkan (*unfreeze*) seluruh *layer* terlebih dahulu menggunakan kode berikut:

```
# Unfreeze beberapa layer terakhir
base_model.trainable = True
for layer in base_model.layers[:-50]: # Freeze semua
kecuali 50 layer terakhir
    layer.trainable = False
```

Kode pertama, *base_model.trainable = True* mengaktifkan kembali semua lapisan dalam model dasar agar bisa diperbarui. Selanjutnya, perulangan *for layer in base_model.layers[:-50]* kemudian membekukan (*freeze*) seluruh lapisan kecuali 50 *layer* terakhir dengan menetapkan *layer.trainable = False,* sehingga 50 *layer* terakhir akan dilatih kembali sesuai dengan data lataih yang digunakan pada penelitian ini.

Proses *fine-tuning* selanjutnya adalah membangun lapisan kustom di atas model dasar MobileNetV2 sehingga dapat digukan kembali sesuai dengan jumlah kelas dari dataset pada penelitian ini. Penambakan *layer* dilakukan dengan kode berikut:

```
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(train_data.num_classes,
activation='softmax')
])
```

Layer pertama adalah *GlobalAveragePooling2D* yang berfungsi untuk mengurangi dimensi fitur dengan mengambil rata-rata setiap *feature map*, sehingga menghasilkan vektor 1D yang lebih ringkas. Kedua, ditambahkan *layer Dense* dengan 256 *neuron* dan aktivasi *ReLU*. Ketiga, *layer Dropout* dengan *rate* 0.5. Terakhir, lapisan *output* dengan aktivasi *softmax* ditambahkan, dimana jumlah neuronnya disesuaikan dengan jumlah kelas pada data latih.

d. Kompilasi dan *Training*

Sebelum melakukan *training*, model perlu dikompilasi untuk menentukan cara model belajar dan cara mengevaluasi performanya. Proses kompilasi dilakukan dengan kode berikut:

```
# Compile model
model.compile(
    optimizer=Adam(learning_rate=0.0001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

Model dikompilasi dengan menyiapkan tiga komponen utama yaitu *optimizer*, *loss function*, dan *metrics*. *Optimizer Adam* dipilih dengan learning rate 0.0001 yang relatif kecil untuk memastikan pembaruan bobot yang stabil selama *fine-tuning*. Fungsi *loss categorical_crossentropy* digunakan karena ini adalah masalah klasifikasi multi-kelas dengan label *one-hot encoded*. Metrik *accuracy* dipantau untuk mengevaluasi performa model selama pelatihan dan validasi.

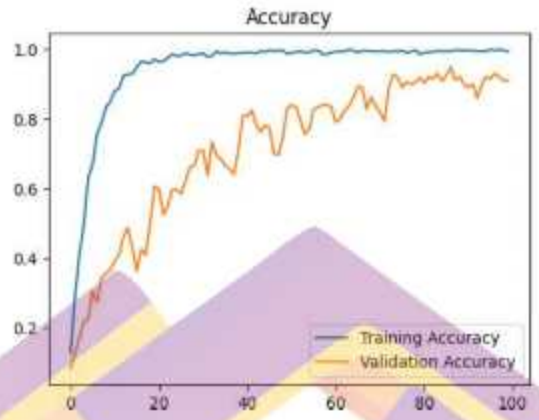
Setelah melakukan kompilasi, model siap untuk dilatih menggunakan data latih yang telah disiapkan sebelumnya. Kode untuk melatiha model adalah sebagai berikut:

```
# Training CNN
history = model.fit(
    train_data,
    validation_data=test_data,
    epochs=100,
    verbose=1
)
```

Proses pelatihan (*training*) model *Convolutional Neural Network* (CNN) menggunakan *library Keras* dari *TensorFlow* di mana, fungsi *model.fit()* bertanggung jawab untuk melatih model pada data yang telah dipersiapkan sebelumnya. Parameter utama yang digunakan meliputi *train_data* sebagai data latih yang dihasilkan dari *ImageDataGenerator*, *validation_data=test_data* untuk memantau performa model pada data uji selama pelatihan, dan *epochs*=100 yang menentukan jumlah iterasi lengkap melalui seluruh dataset.

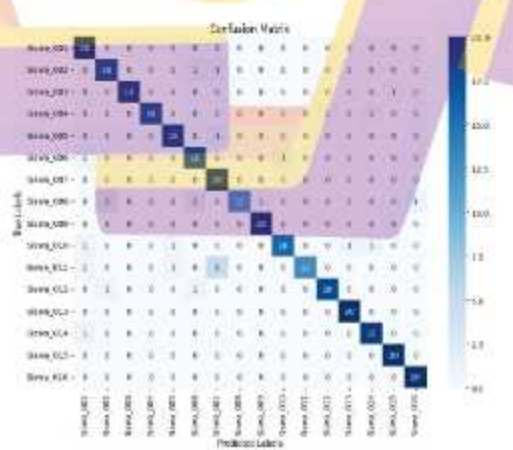
e. Hasil *Training* dan *Testing*

Skenario dua menggunakan arsitektur MobileNetV2 yang memiliki 154 *layers* ditambah dengan 4 *layers custom* hasil *fine-tuning*. Proses *training* dilakukan untuk 50 *layers* akhir dan 4 *layers custom* dengan menggunakan 100 *epoch* dan ukuran gambar 224 x 224 pixel. Hasil *training* diperlihatkan pada gambar 4.11.



Gambar 4. 11. Proses *Training* Skenario Dua

Gambar 4.11 menunjukan *accuracy* trainig mencapai 99% dan *accuracy* validasi mencapai 91%. Selanjutnya hasil testing digambarkan dalam *confusion matrix* pada gambar 4.12 dan dijelaskan dengan *classification report* pada gambar 4.13.



Gambar 4. 12. *Confusion Matrix* Skenario Dua

```
Classification Report:
```

	precision	recall	f1-score	support
Siswa_001	0.83	1.00	0.91	20
Siswa_002	0.82	0.90	0.86	20
Siswa_003	1.00	0.95	0.97	20
Siswa_004	1.00	0.90	0.95	20
Siswa_005	0.90	0.95	0.93	20
Siswa_006	0.82	0.90	0.86	20
Siswa_007	0.74	1.00	0.85	20
Siswa_008	1.00	0.65	0.79	20
Siswa_009	0.95	1.00	0.98	20
Siswa_010	0.94	0.80	0.86	20
Siswa_011	1.00	0.65	0.79	20
Siswa_012	0.95	0.90	0.92	20
Siswa_013	0.95	1.00	0.98	20
Siswa_014	0.90	0.95	0.93	20
Siswa_015	0.95	1.00	0.98	20
Siswa_016	0.95	1.00	0.98	20
accuracy			0.91	320
macro avg	0.92	0.91	0.91	320
weighted avg	0.92	0.91	0.91	320

Gambar 4. 13. *Classification Report* Skenario Dua

Gambar 4.13 menunjukan hasil testing untuk model skenario dua mendapat nilai *accuracy* 91%, *precision* 92% dan *recall* 91%.

4.2.2.3. Skenario Tiga

a. Menyiapkan dataset

Sebelum melakukan *training* dan *testing*, terlebih dahulu menyiapkan dataset yang telah melalui augmentasi. Menyiapkan dataset dilakukan dengan kode berikut:

```
# Data latih
train_data = train_datagen.flow_from_directory(
    train_dir,
    target_size=(160, 160),
    batch_size=32,
    class_mode='categorical',
    shuffle=True,
)

# Data Uji
test_data = test_datagen.flow_from_directory(
    test_dir,
    target_size=(160, 160),
    batch_size=32,
    class_mode='categorical',
    shuffle=False,
)
```

Data latih dimuat menggunakan fungsi *flow_from_directory*, di mana gambar diubah ukurannya menjadi 160x160 piksel (*target_size*), dikelompokkan dalam *batch* berisi 32 gambar (*batch_size*), dan labelnya dikonversi ke format *one-hot encoding* (*class_mode='categorical'*). Pengacakan data (*shuffle=True*) juga dilakukan untuk mencegah model menghafal urutan gambar. Gambar dalam data uji juga diubah ukurannya menjadi 160x160 piksel untuk konsistensi dengan data latih. Namun, berbeda dengan data latih, data uji tidak diacak (*shuffle=False*) agar urutan gambar tetap konsisten selama evaluasi, yang memudahkan analisis hasil.

b. Menyiapkan arsitektur

Untuk menerapkan *transfer learning* sesuai dengan skenario pengujian, perlu untuk meyiapkan arsitektur yang telah dilatih sebelumnya (*pre-trained*) menggunakan kode berikut:

```
# Load MobileNetV2 pre-trained model
base_model = InceptionResNetV2(
    weights='imagenet', include_top=False, input_shape=(160,
160, 3)
```

Skenario tiga menggunakan arsitektur ResNetV2 dengan dataset *ImageNet* sebagai model dasar (*base model*) untuk tugas *transfer learning*. Model ini sudah dioptimalkan dari *ImageNet* (*weights='imagenet'*), sehingga memiliki kemampuan awal untuk mengekstrak fitur-fitur penting dari gambar. *Input shape* yang digunakan adalah (160, 160, 3), yang berarti model menerima gambar berukuran 160x160 piksel dengan 3 saluran warna (RGB). Parameter (*include_top=False*) berfungsi untuk menghilangkan lapisan terakhir (*fully connected layer*) dari model asli karena lapisan tersebut spesifik untuk klasifikasi 1000 kelas *ImageNet*. Kita ingin menyesuaikan model untuk tugas baru dengan jumlah kelas berbeda.

c. *Fine-Tuning*

Pada skenario tiga, proses *fine-tuning* terhadap model dasar ResNetV2 dcilakukan dengan membekukan (*freeze*) seluruh *layer* terlebih dahulu menggunakan kode berikut:

```
# Unfreeze beberapa layer terakhir
for layer in base_model.layers:
    layer.trainable = False
```

Melalui perulangan *for layer* in, *base_model.layers* diberikan parameter "*layer.trainable = False*" dengan tujuan pembekuan *layer* untuk mempertahankan pengetahuan yang sudah dipelajari model dari *ImageNet* selama pelatihan awal, sekaligus mencegah pembaruan bobot yang tidak perlu pada lapisan awal.

Proses *fine-tuning* selanjutnya adalah membangun lapisan kustom di atas model dasar ResNetV2 sehingga dapat digukan kembali sesuai dengan jumlah kelas dari dataset pada penelitian ini. Penambakan *layer* dilakukan dengan kode berikut:

```
# Tambahkan layer klasifikasi sementara
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(train_data.num_classes,
activation='softmax')
])
```

Layer pertama adalah *GlobalAveragePooling2D* yang berfungsi untuk mengurangi dimensi fitur dengan mengambil rata-rata setiap *feature map*, sehingga menghasilkan vektor 1D yang lebih ringkas. Kedua, ditambahkan *layer Dense* dengan 256 *neuron* dan aktivasi *ReLU*. Ketiga, *layer Dropout* dengan *rate* 0.5. Terakhir, lapisan *output* dengan aktivasi *softmax* ditambahkan, dimana jumlah neuronnya disesuaikan dengan jumlah kelas pada data latih.

d. Kompilasi dan *Training*

Sebelum melakukan *training*, model perlu dikompilasi untuk menentukan cara model belajar dan cara mengevaluasi performanya. Proses kompilasi dilakukan dengan kode berikut:

```
# Compile model
model.compile(
    optimizer=Adam(learning_rate=0.0001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

Model dikompilasi dengan menyiapkan tiga komponen utama yaitu *optimizer*, *loss function*, dan *metrics*. *Optimizer Adam* dipilih dengan learning rate 0.0001 yang relatif kecil untuk memastikan pembaruan bobot yang stabil selama *fine-tuning*. Fungsi *loss categorical_crossentropy* digunakan karena ini adalah masalah klasifikasi multi-kelas dengan label *one-hot encoded*. Metrik *accuracy* dipantau untuk mengevaluasi performa model selama pelatihan dan validasi.

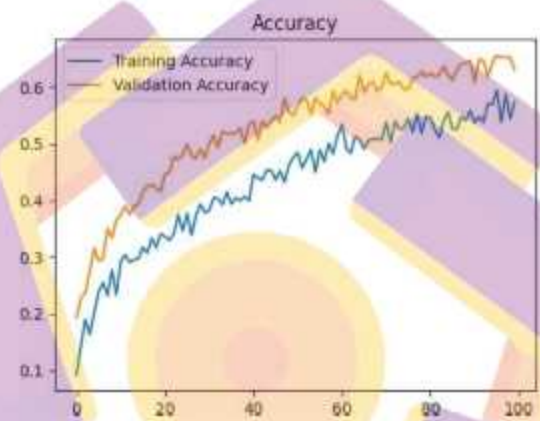
Setelah melakukan kompilasi, model siap untuk dilatih menggunakan data latih yang telah disiapkan sebelumnya. Kode untuk melatiha model adalah sebagai berikut:

```
# Training CNN
history = model.fit(
    train_data,
    validation_data=test_data,
    epochs=100,
    verbose=1
)
```

Proses pelatihan (*training*) model *Convolutional Neural Network* (CNN) menggunakan *library Keras* dari *TensorFlow* di mana, fungsi *model.fit()* bertanggung jawab untuk melatih model pada data yang telah dipersiapkan sebelumnya. Parameter utama yang digunakan meliputi *train_data* sebagai data latih yang dihasilkan dari *ImageDataGenerator*, *validation_data=test_data* untuk memantau performa model pada data uji selama pelatihan, dan *epochs*=100 yang menentukan jumlah iterasi lengkap melalui seluruh dataset.

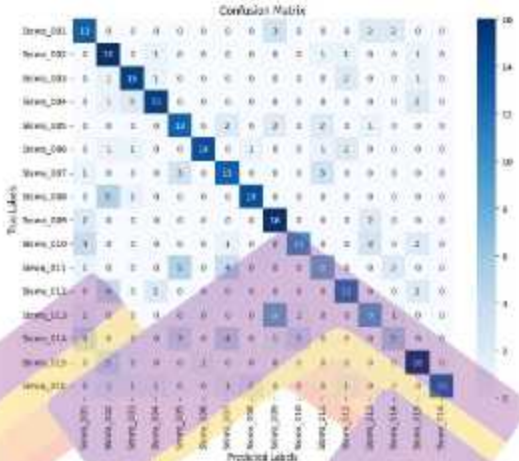
e. Hasil *Training* dan *Testing*

Skenario tiga menggunakan arsitektur ResNetV2 yang memiliki 780 *layers* ditambah dengan 4 *layers custom* hasil *fine-tuning*. Proses *training* hanya dilakukan untuk 4 *layers custom* dengan menggunakan 100 *epoch* dan ukuran gambar 160 x 160 pixel. Hasil *training* diperlihatkan pada gambar 4.14.



Gambar 4. 14. Proses *Training* Skenario Tiga

Gambar 4.14 menunjukan *accuracy training* mencapai 57% dan *accuracy* validasi mencapai 63%. Selanjutnya hasil *testing* digambarkan dalam *confusion matrix* pada gambar 4.15 dan dijelaskan dengan *classification report* pada gambar 4.16.



Gambar 4. 15. *Confusion Matrix* Skenario Tiga

Classification Report:

	precision	recall	f1-score	support
Siswa_001	0.50	0.65	0.57	20
Siswa_002	0.50	0.80	0.62	20
Siswa_003	0.75	0.75	0.75	20
Siswa_004	0.75	0.75	0.75	20
Siswa_005	0.54	0.65	0.59	20
Siswa_006	0.93	0.70	0.80	20
Siswa_007	0.52	0.65	0.58	20
Siswa_008	0.82	0.70	0.76	20
Siswa_009	0.53	0.80	0.64	20
Siswa_010	0.71	0.50	0.59	20
Siswa_011	0.53	0.40	0.46	20
Siswa_012	0.67	0.60	0.63	20
Siswa_013	0.53	0.45	0.49	20
Siswa_014	0.50	0.25	0.33	20
Siswa_015	0.67	0.80	0.73	20
Siswa_016	1.00	0.65	0.79	20
accuracy			0.63	320
macro avg	0.65	0.63	0.63	320
weighted avg	0.65	0.63	0.63	320

Gambar 4. 16. *Classification Report* Skenario Tiga

Gambar 4.16 menunjukan hasil testing untuk model skenario tiga mendapat nilai *accuracy* 63%, *precision* 65% dan *recall* 63%.

4.2.2.4. Skenario Empat

a. Menyiapkan dataset

Sebelum melakukan *training* dan *testing*, terlebih dahulu menyiapkan dataset yang telah melalui augmentasi. Menyiapkan dataset dilakukan dengan kode berikut:

```
# Data latih
train_data = train_datagen.flow_from_directory(
    train_dir,
    target_size=(160, 160),
    batch_size=32,
    class_mode='categorical',
    shuffle=True,
)

# Data Uji
test_data = test_datagen.flow_from_directory(
    test_dir,
    target_size=(160, 160),
    batch_size=32,
    class_mode='categorical',
    shuffle=False,
)
```

Data latih dimuat menggunakan fungsi *flow_from_directory*, di mana gambar diubah ukurannya menjadi 160x160 piksel (*target_size*), dikelompokkan dalam *batch* berisi 32 gambar (*batch_size*), dan labelnya dikonversi ke format *one-hot encoding* (*class_mode='categorical'*). Pengacakan data (*shuffle=True*) juga dilakukan untuk mencegah model menghafal urutan gambar. Gambar dalam data uji juga diubah ukurannya menjadi 160x160 piksel untuk konsistensi dengan data latih. Namun, berbeda dengan data latih, data uji tidak diacak (*shuffle=False*) agar urutan gambar tetap konsisten selama evaluasi, yang memudahkan analisis hasil.

b. Menyiapkan arsitektur

Untuk menerapkan *transfer learning* sesuai dengan skenario pengujian, perlu untuk meyiapkan arsitektur yang telah dilatih sebelumnya (*pre-trained*) menggunakan kode berikut:

```
# Load MobileNetV2 pre-trained model
base_model = InceptionResNetV2(
    weights='imagenet', include_top=False, input_shape=(160,
160, 3)
```

Skenario empat menggunakan arsitektur ResNetV2 dengan dataset *ImageNet* sebagai model dasar (*base model*) untuk tugas *transfer learning*. Model ini sudah dioptimalkan dari *ImageNet* (*weights='imagenet'*), sehingga memiliki kemampuan awal untuk mengekstrak fitur-fitur penting dari gambar. *Input shape* yang digunakan adalah (160, 160, 3), yang berarti model menerima gambar berukuran 160x160 piksel dengan 3 saluran warna (RGB). Parameter (*include_top=False*) berfungsi untuk menghilangkan lapisan terakhir (*fully connected layer*) dari model asli karena lapisan tersebut spesifik untuk klasifikasi 1000 kelas *ImageNet*. Kita ingin menyesuaikan model untuk tugas baru dengan jumlah kelas berbeda.

c. *Fine-Tuning*

Pada skenario empat, proses *fine-tuning* terhadap model dasar ResNetV2 deilakukan dengan mengaktifkan (*unfreeze*) seluruh *layer* terlebih dahulu menggunakan kode berikut:

```
# Unfreeze beberapa layer terakhir
base_model.trainable = True
for layer in base_model.layers[:-50]: # Freeze semua
kecuali 50 layer terakhir
    layer.trainable = False
```

Kode pertama, *base_model.trainable = True* mengaktifkan kembali semua lapisan dalam model dasar agar bisa diperbarui. Selanjutnya, perulangan *for layer in base_model.layers[:-50]* kemudian membekukan (*freeze*) seluruh lapisan kecuali 50 *layer* terakhir dengan menetapkan *layer.trainable = False*, sehingga 50 *layer* terakhir akan dilatih kembali sesuai dengan data latih yang digunakan pada penelitian ini.

Proses *fine-tuning* selanjutnya adalah membangun lapisan kustom di atas model dasar ResNetV2 sehingga dapat digukan kembali sesuai dengan jumlah kelas dari dataset pada penelitian ini. Penambakan *layer* dilakukan dengan kode berikut:

```
# Tambahkan layer klasifikasi sementara
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(train_data.num_classes,
activation='softmax')
])
```

Layer pertama adalah *GlobalAveragePooling2D* yang berfungsi untuk mengurangi dimensi fitur dengan mengambil rata-rata setiap *feature map*, sehingga menghasilkan vektor 1D yang lebih ringkas. Kedua, ditambahkan *layer Dense* dengan 256 *neuron* dan aktivasi *ReLU*. Ketiga, *layer Dropout* dengan *rate* 0.5.

Terakhir, lapisan *output* dengan aktivasi *softmax* ditambahkan, dimana jumlah neuronnya disesuaikan dengan jumlah kelas pada data latih.

d. Kompilasi dan *Training*

Sebelum melakukan *training*, model perlu dikompilasi untuk menentukan cara model belajar dan cara mengevaluasi performanya. Proses kompilasi dilakukan dengan kode berikut:

```
# Compile model
model.compile(
    optimizer=Adam(learning_rate=0.0001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

Model dikompilasi dengan menyiapkan tiga komponen utama yaitu *optimizer*, *loss function*, dan *metrics*. *Optimizer Adam* dipilih dengan learning rate 0.0001 yang relatif kecil untuk memastikan pembaruan bobot yang stabil selama *fine-tuning*. Fungsi *loss categorical_crossentropy* digunakan karena ini adalah masalah klasifikasi multi-kelas dengan label *one-hot encoded*. Metrik *accuracy* dipantau untuk mengevaluasi performa model selama pelatihan dan validasi.

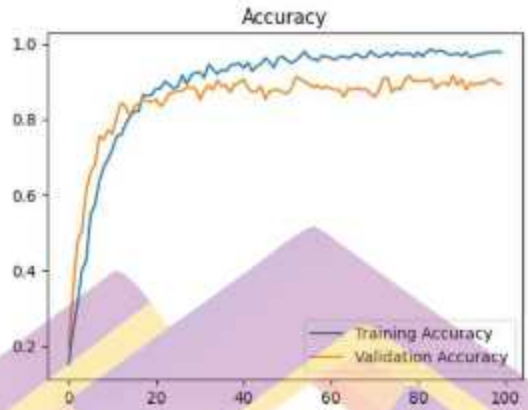
Setelah melakukan kompilasi, model siap untuk dilatih menggunakan data latih yang telah disiapkan sebelumnya. Kode untuk melatiha model adalah sebagai berikut:

```
# Training CNN
history = model.fit(
    train_data,
    validation_data=test_data,
    epochs=100,
    verbose=1
)
```

Proses pelatihan (*training*) model *Convolutional Neural Network* (CNN) menggunakan *library Keras* dari *TensorFlow* di mana, fungsi *model.fit()* bertanggung jawab untuk melatih model pada data yang telah dipersiapkan sebelumnya. Parameter utama yang digunakan meliputi *train_data* sebagai data latih yang dihasilkan dari *ImageDataGenerator*, *validation_data=test_data* untuk memantau performa model pada data uji selama pelatihan, dan *epochs*=100 yang menentukan jumlah iterasi lengkap melalui seluruh dataset.

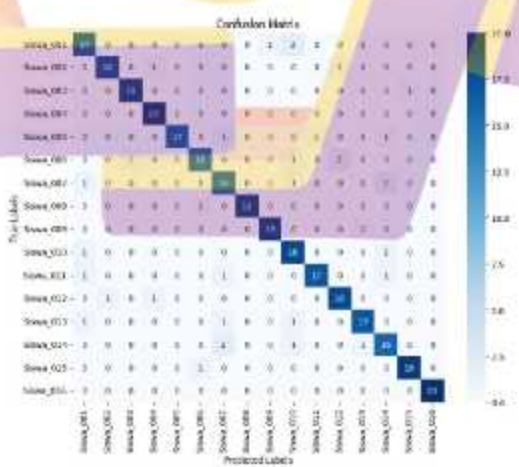
e. Hasil *Training* dan *Testing*

Skenario empat menggunakan arsitektur ResNetV2 yang memiliki 780 *layers* ditambah dengan 4 *layers custom* hasil *fine-tuning*. Proses *training* dilakukan untuk 50 *layers* akhir dan 4 *layers custom* dengan menggunakan 100 *epoch* dan ukuran gambar 160 x 160 pixel. Hasil *training* diperlihatkan pada gambar 4.17.

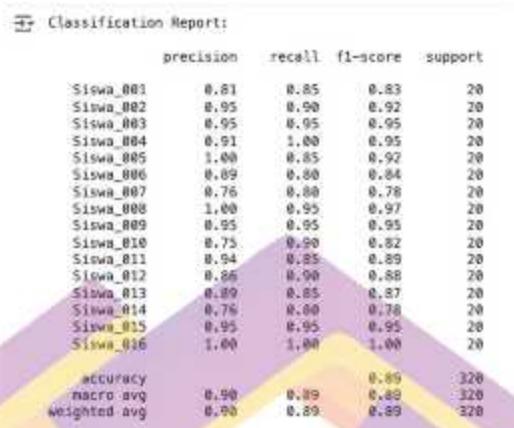


Gambar 4. 17. Proses *Training* Skenario Empat

Gambar 4.17 menunjukan *accuracy* trainig mencapai 98% dan *accuracy* validasi mencapai 89%. Selanjutnya hasil *testing* digambarkan dalam *confusion matrix* pada gambar 4.18 dan dijelaskan dengan *classification report* pada gambar 4.19.



Gambar 4. 18. *Confusion Matrix* Skenario Empat



Classification Report:

	precision	recall	f1-score	support
Siswa_001	0.81	0.85	0.83	20
Siswa_002	0.95	0.90	0.92	20
Siswa_003	0.95	0.95	0.95	20
Siswa_004	0.91	1.00	0.95	20
Siswa_005	1.00	0.85	0.92	20
Siswa_006	0.89	0.80	0.84	20
Siswa_007	0.76	0.80	0.78	20
Siswa_008	1.00	0.95	0.97	20
Siswa_009	0.95	0.95	0.95	20
Siswa_010	0.75	0.90	0.82	20
Siswa_011	0.94	0.85	0.89	20
Siswa_012	0.86	0.90	0.88	20
Siswa_013	0.89	0.85	0.87	20
Siswa_014	0.76	0.80	0.78	20
Siswa_015	0.95	0.95	0.95	20
Siswa_016	1.00	1.00	1.00	20
accuracy			0.89	320
macro avg	0.90	0.89	0.89	320
weighted avg	0.90	0.89	0.89	320

Gambar 4. 19. *Classification Report* Skenario Empat

Gambar 4.19 menunjukan hasil testing untuk model skenario empat mendapat nilai *accuracy* 89%, *precision* 90% dan *recall* 89%.

4.3. Analisis Hasil Penelitian

Untuk mendapatkan model terbaik, nilai *precision*, *recall* dan *accuracy* hasil pengujian semua skenario akan dibandingkan. Hasil pengujian untuk semua skenario diperlihatkan pada tabel 4.3 dan gambar 4.20.

Tabel 4. 3. Hasil Pengujian Penerapan Algoritma CNN

Skenario	*Precision*	*Recall*	*Accuracy*
S1	71%	66%	66%
S2	92%	91%	91%
S3	65%	63%	63%
S4	90%	89%	89%

Gambar 4. 20. Grafik Hasil Pengujian Algoritma CNN

Model pengenalan wajah yang dibagun dengan empat skenario, diketahui skenario dua mendapatkan hasil *precision*, *recall* dan *accuracy* yang lebih unggul yaitu *precision* 92%, *recall* 91% dan *accuracy* 91%. Skenario terendah adalah skenario tiga dengan nilai *precision* 65%, *recall* 63% dan *accuracy* 63%.

Berdasarkan hasil pengujian dari empat skenario, terlihat bahwa variasi teknik *fine-tuning* pada *MobileNetV2* dan *ResNetV2* menghasilkan perbedaan akurasi yang cukup signifikan. Pada skenario pertama dan ketiga (hanya *custom layer*), akurasi yang diperoleh relatif stabil namun tidak terlalu tinggi. Hal ini dapat dijelaskan karena sebagian besar layer *pretrained* tetap dibekukan, sehingga model hanya belajar dari lapisan tambahan. *MobileNetV2* dan *ResNetV2* kurang mampu beradaptasi secara penuh pada dataset baru jika *fine-tuning* terlalu terbatas.

Pada skenario kedua dan keempat (*fine-tuning* 50 *layers* terakhir ditambah dengan *custom layer*), terjadi peningkatan akurasi yang signifikan. Peningkatan ini terjadi karena 50 *layers* terakhir ikut belajar menyesuaikan diri dengan pola wajah dari dataset baru, sehingga fitur yang diekstrak menjadi lebih relevan.

Secara umum, *MobileNetV2* dengan *fine-tuning* 50 layer terakhir ditambah dengan *custom layer* lebih cocok digunakan ketika sumber daya komputasi terbatas dan dataset relatif kecil temuan ini sejalan dengan penelitian (Mikolaj et al. 2024), yang menekankan bahwa *fine-tuning* lapisan tertentu dapat meningkatkan akurasi secara signifikan.

Pembahasan hasil penelitian ini menunjukkan bahwa strategi *fine-tuning multi-layer* pada arsitektur CNN memberikan peningkatan performa yang signifikan. *MobileNetV2* terbukti mampu menghasilkan akurasi di atas 90% ketika dipadukan dengan augmentasi data dan pembersihan dataset *pasca-cropping*. Faktor lain yang berkontribusi terhadap peningkatan ini adalah penerapan strategi augmentasi data untuk memperluas variasi wajah dan pencahayaan, serta pembersihan data yang mengurangi *noise* dan *data outlier*. Dengan demikian, hasil eksperimen ini tidak hanya menegaskan keunggulan arsitektur yang digunakan, tetapi juga menunjukkan bahwa kombinasi teknik *preprocessing*, augmentasi, dan *fine-tuning* dapat menghasilkan sistem pengenalan wajah yang lebih andal dalam berbagai kondisi nyata.

Penelitian yang dilakukan oleh (Mikolaj et al. 2024) yang bertujuan untuk meningkatkan akurasi klasifikasi model CNN dengan mengoptimalkan strategi *fine-tuning* pada arsitektur *Xception*. Penelitian ini menyimpulkan bahwa melatih ulang sebagian layer terakhir pada model CNN pretrained sekaligus membekukan (*freeze*) layer awal secara signifikan dapat meningkatkan akurasi klasifikasi hingga 6% dibandingkan dengan pelatihan tanpa *fine-tuning*, dari 77,3 menjadi 83% untuk dataset CIFAR-100 sedangkan dataset lain seperi *Stanford Dogs* peningkatan

akurasi hanya mencapai 0,8% hasil penelitian ini menyatakan tidak ada metode tunggal yang selalu terbaik di semua dataset, sehingga pemilihan metode harus disesuaikan dengan dataset dan tugas klasifikasi. Jika melihat hasil penelitian yang telah dilakukan untuk aristektus *MobileNetV2, fine tuning* dengan melatih ulang 50 *layers* terakhir dapat meningkatkan akurasi mencapai 25% dan ResNetV2, *fine tuning* dengan melatih ulang 50 *layers* terakhir dapat meningkatkan akurasi mencapai 26%.

Jika dibandingkan dengan penelitian terdahulu yang dilakukan oleh (Kumar and Kumar 2025) yang juga menggunakan dataset serupa dengan menggunakan *transfer learning* dengan VGG16, penelitian ini hanya mendapatkan akurasi validasi sebeser 76,39% sedangkan penelitian yang telah dilakukan menggunakan arsitektur *MobileNetV2* ditambah dengan metode *fine-tuning* pada 50 *layers* terakhir mencapai akurasi yang lebih tinggi yaitu 91%. Ini menunjukkan bahwa penelitian yang telah dilakukan menghasilkan performa yang cukup kompetitif dimana pendekatan *transfer learning* dengan *fine-tuning* memberikan peningkatan performa yang signifikan, terutama dalam menyesuaikan model terhadap variasi wajah. Oleh karena itu, klaim bahwa penelitian ini memberikan hasil yang lebih baik memiliki dasar kuat dalam literatur dan bukti empiris dari eksperimen.

BAB V

PENUTUP

5.1. Kesimpulan

Berdasarkan penelitian yang dilakukan mengenai klasifikasi pengenalan wajah untuk sistem kehadiran menggunakan *Convolutional Neural Network* (CNN) dengan teknik *transfer learning*, dapat disimpulkan bahwa:

a. Untuk membagun model pengenalan wajah menggunakan algoritma *Convolutional Neural Network* (CNN) dengan teknik *transfer learning* dari arsitektur *MobileNetV2* dan *ResNetV2* diawali dengan mengumpilkan dataset. Dataset yang terkumpul akan melalui *preprocessing* seperti *cropping* wajah otomatis menggunakan *haar cascade*, *cleaning* data, pembagian data dan augmentasi data untuk meningkatkan kualitas dataset. Selanjutnya dilakukan prose *training* model denga empat skenario pelatihan untuk mendapatkan model dengan akurasi terbaik.

b. *MobileNetV2* terbukti unggul dibanding *ResNetV2* pada konteks dataset ini, mencapai *accuracy* 91%, *precision* 92%, dan *recall* 91% pada skenario dua. Sedangkan arsitektur *ResNetV2* pada skenario empat mendapat nilai *accuracy* 89%, *precision* 89%, dan *recall* 90%.

c. Beberapa faktor kunci yang memengaruhi akurasi yang pertama, *fine-tuning* yang melatih ulang sebagian lapisan arsitektur yang meningkatkan adaptasi model. Berdasarkan skenario pengujian dua dan empat bahwa semakin dalam *fine-tuning* dilakuka hingga menyentuh 50 lapisan terakhir, semakin besar

kemampuan model dalam menyesuaikan representasi fitur terhadap karakteristik dataset. Kedua, augmentasi data dimana variasi gambar wajah memperkuat generalisasi model. Ketiga, kualitas dataset karena meskipun jumlah data terbatas, pembersihan dan pembagian data yang tepat berkontribusi pada hasil optimal. Dan keempat pemilihan arsitektur dimana *MobileNetV2* lebih baik untuk kasus ini dibanding *ResNetV2*.

5.2. Saran

Penerapan model *deep learning* dalam sistem presensi berbasis pengenalan wajah memiliki potensi besar untuk meningkatkan efisiensi dan akurasi pencatatan kehadiran siswa ditunjukkan dari hasil pengujian yang cukup baik. Namun, beberapa keterbatasan penelitian ini harus dipertimbangkan seperti ukuran dataset yang relatif kecil yang dapat mempengaruhi generalisasi model terhadap variasi wajah yang lebih luas seperti perubahan pencahayaan, ekspresi wajah, atau aksesori yang digunakan oleh siswa. Oleh karena itu, studi lanjutan merekomendasikan penggunaan dataset yang lebih besar dan beragam, serta mengeksplorasi teknik regularisasi guna meningkatkan kemampuan generalisasi model. Selain itu, evaluasi performa model dalam kondisi nyata perlu dilakukan untuk memastikan keandalan sistem presensi dalam berbagai skenario penggunaan.

DAFTAR PUSTAKA

Ahmed, Shams Forruque et al. 2023. 56 Artificial Intelligence Review *Deep Learning Modelling Techniques : Current Progress , Applications , Advantages , and Challenges*. Springer Netherlands. https://doi.org/10.1007/s10462-023-10466-8.

Ai, Mona A. S. et al. 2022. "Real-Time Facemask Detection for Preventing COVID-19 Spread Using Transfer Learning Based Deep Neural Network." *Electronics* 11(14): 1–21.

Alhanaee, Khawla, Mitha Alhammadi, Nahla Almenhali, and Maad Shatnawi. 2021. "Face Recognition Smart Attendance System Using Deep Transfer Learning." *Procedia Computer Science* 192: 4093–4102. https://doi.org/10.1016/j.procs.2021.09.184.

Anarki, Galang Aprilian, Karina Auliasari, and Mira Orisa. 2021. "Penerapan Metode Haar Cascade Pada Aplikasi Deteksi Masker." *JATI (Jurnal Mahasiswa Teknik Informatika)* 5(1): 179–86.

Andono, Pulung Nurtantio, T. Sutojo, and Muljono. 2017. *Pengolahan Citra Digital*. Yogyakarta: Penerbit Andi.

Anggraini, Dyah Putri, Dina Eka Fitriani, Fachriza Ulfah, and Tinuk Agustin. 2024. "Klasifikasi Ekspresi Wajah Menggunakan Metode Convolutional Neural Network (CNN) Dengan Perbandingan Dua Model Yang Dimodifikasi." *Prosiding Seminar Nasional Amikom Surakarta* (November): 160–71.

Arief, Nindo Syafi'al, Wahyu Anggara Putra, and Dieky Septhian Rastra Pratama. 2024. "Implementasi Cnn Arsitektur Mobilenetv2 Untuk Klasifikasi Tulisan Aksara Jawa." *STAINS Seminar Nasional Teknologi & Sains* 3(1): 298–303.

Asmara, Rosa Andrie. 2018. *Pengolahan Citra Digital*. Malang: Polinema Press.

Astawa, I Nyoman Gede Arya, Made Leo Radhitya, I Wayan Raka Ardana, and Felix Andika Dwiyanto. 2021. "Face Images Classification Using VGG-CNN." *Knowledge Engineering and Data Science (KEDS)* 4(1): 49–54.

Chitrapu, Pavani, Mahesh Kumar Morampudi, and Hemantha Kumar Kalluri. 2025. "Robust Face Recognition Using Deep Learning and Ensemble Classification." *IEEE Access* 13(June): 99957–69.

DLY, Ikhwanul Akhmad et al. 2023. "Klasifikasi Citra Daging Sapi Dan Babi Menggunakan CNN Alexnet Dan Augmentasi Data." *Journal of Information System Research (JOSH)* 4(4): 1176–85.

Garcia, Jayme, and Arnal Barbedo. 2018. "Impact of Dataset Size and Variety on the e Ff Ectiveness of Deep Learning and Transfer Learning for Plant Disease

Classification." *Computers and Electronics in Agriculture* 153(July): 46–53.

Kumar, Abhishek, and Kapil Kumar. 2025. "Facial Recognition and Object Detection Using Machine Learning." *Journal of Recent Innovations in Computer Science and Technology* 2(2): 39–48.

Kusumanto, R D, and Alan Novi Tompunu. 2011. "Pengolahan Citra Digital Untuk Mendeteksi Obyek Menggunakan Pengolahan Warna Model Normalisasi RGB." *Seminar Nasional Teknologi Informasi & Komunikasi Terapan 2011* 2011(Semantik).

Liu, Yu Han. 2018. "Feature Extraction and Image Recognition with Convolutional Neural Networks Feature Extraction and Image Convolutional Neural Networks Recognition With." *First International Conference on Advanced Algorithms and Control Engineering*.

Mikolaj, Wojciuk, Swiderska-chadaj Zaneta, Siwek Krzysztof, and Gertych Arkadiusz. 2024. "Improving Classification Accuracy of Fine-Tuned CNN Models: Impact of Hyperparameter Optimization." *Heliyon* 10(5): 1–11.

Moolayil, Jojo. 2019. *Learn Keras for Deep Neural Networks*. Canada: Apress.

Nugraha, Prima, Agus Komarudin, and Edvin Ramadhan. 2022. "Deteksi Objek Dan Jenis Burung Menggunakan Convolutional Neural Network Dengan Arsitektur Inception Resnet-V2." *Infotech Journal* 8(2): 47–55.

Nurkhamid et al. 2021. "Intelligent Attendance System with Face Recognition Using the Deep Convolutional Neural Network Method." *Journal of Physics: Conference Series*.

Putra, Jan Wira Gotama. 2020. *Pengenalan Konsep Pembelajaran Mesin Dan Deep Learning*. 1.4. Tokyo.

Rahim, Arham, and Kusrini Kusrini. 2021. "Convolutional Neural Network Untuk Kalasifikasi Penggunaan Masker." *Inspiration : Jurnal Teknologi Informasi dan Komunikasi* 10(2): 109–15.

Ramadhani, Faadiyah, Septian Rahardiantoro, and Mohammad Masjkur. 2024. "Acne Severity Classification Study Using Convolutional Neural Network Algorithm with MobileNetV2 Architecture." *Indonesian Journal of Statistics and Its Applications* 8(2): 112–28.

Rambe, Rahmat, and Lukman Abdurrahman. 2024. "Implikasi Etika Dan Hukum Dalam Penggunaan Teknologi Pengenalan Wajah: Perlindungan Privasi Versus Keamanan Publik." *Jurnal Hukum Caraka Justitia* 4(2): 90–104.

Ramdhani, Syahrul Gunawan, and Enny Itje Sela. 2023. "Implementasi Face Recognition Untuk Sistem Presensi Universitas Menggunakan Convolutional

Neural Network." *Indonesian Journal of Computer Science* 12(6): 4098–4108.

Razaqa, Dhi'fan, Muhammad Rezka Al Maghribi, Nabilah Dwi Gunasti, and Theresia Wati. 2024. "Analisis Etika Dan Dampak Penggunaan Sistem Pengenalan Wajah Untuk Manajemen Kehadiran Di Lingkungan Sekolah." *Jurnal Informatika* 20(2): 50–57.

Sandler, Mark et al. 2018. "MobileNetV2: Inverted Residuals and Linear Bottlenecks." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*: 4510–4520.

Sasongko, Theopilus Bayu, and Agit Amrullah. 2023. "Analisis Efek Augmentasi Dataset Dan Fine Tune Pada Algoritma Pre-Trained Convolutional Neural Network (CNN)." *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK)* 10(4): 763–68.

Shukla, Ratnesh Kumar, and Arvind Kumar Tiwari. 2023. "Masked Face Recognition Using MobileNet V2 with Transfer Learning." *Computer Systems Science & Engineering* 45(1): 293–309.

Singh, Himanshu. 2019. *Practical Machine Learning and Image Processing For Facial Recognition, Object Detection, and Pattern Recognition Using Python*. Uttar Pradesh: Apress.

Tan, Chuanqi et al. 2018. "A Survey on Deep Transfer Learning." *Springer Nature Switzerland AG*: 270–79. http://dx.doi.org/10.1007/978-3-030-01424-7_27.

Tjahyanti, Luh Putu Ary Sri, Putu Satya Saputra, and Made Santo Gitakarma. 2022. "Peran Artificial Intelligence (Ai) Untuk Mendukung Pembelajaran Di Masa Pandemi Covid-19." *Jurnal Komputer dan Teknologi Sains (KOMTEKS)* 1(1): 15–21.

Varkarakis, Viktor, and Peter Corcoran. 2020. "Dataset Cleaning - A Cross Validation Methodology for Large Facial Datasets Using Face Recognition." *IEEE Access*: 0–5.

Wijayanto, Hanang. 2015. "Klasifikasi Batik Menggunakan Metode K-Nearest Neighbour Berdasarkan Gray Level Co-Occurrence Matrices (GLCM)." *Jurusan Teknik Informatika FIK UDINUS* (5).

Wiranda, Nuruddin, Harja Santana Purba, and R Ati Sukmawati. 2020. "Survei Penggunaan Tensorflow Pada Machine Learning Untuk Identifikasi Ikan Kawasan Lahan Basah." *Indonesian Journal of Electronics and Instrumentation Systems (IJEIS)* 10(2): 179–88.

Zhao, W, R. Chellappa, P. J. Phillips, and A. Rosenfeld. 2003. "Face Recognition : A Literature Survey." *ACM Computing Surveys* 35(4): 399–458.

Zufar, Muhammad, and Budi Setiyono. 2016. "Convolutional Neural Networks Untuk Pengenalan Wajah Secara Real - Time." *Jurnal Sains dan Seni ITS* 5(2): 72–77.