TESIS

KLASIFIKASI DETEKSI PENGGUNAAN MASKER MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK

Disusun oleh:

Nama	**: Nalda Kresimo Negoro**
NIM	**: 21.52.1019**
Konsentrasi	**: Business Intelligence**

PROGRAM STUDI S2 TEKNIK INFORMATIKA

PROGRAM PASCASARJANA UNIVERSITAS AMIKOM YOGYAKARTA

YOGYAKARTA

2023

TESIS

KLASIFIKASI DETEKSI PENGGUNAAN MASKER MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK

CLASSIFICATION OF MASK USAGE DETECTION USING CONVOLUTIONAL NEURAL NETWORK METHOD

Diajukan untuk memenuhi salah satu syarat memperoleh derajat Magister

Disusun oleh:

Nama : **Nalda Kresimo Negoro**
NIM : **21.52.1019**
Konsentrasi : **Business Intelligence**

PROGRAM STUDI S2 TEKNIK INFORMATIKA
PROGRAM PASCASARJANA UNIVERSITAS AMIKOM YOGYAKARTA
YOGYAKARTA
2023

HALAMAN PENGESAHAN

KLASIFIKASI DETEKSI PENGGUNAAN MASKER MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK

CLASSIFICATION OF MASK USAGE DETECTION USING CONVOLUTIONAL NEURAL NETWORK METHOD

Dipersiapkan dan Disusun oleh

Nalda Kresimo Negoro

21.52.1019

Telah Diujikan dan Dipertahankan dalam Sidang Ujian Tesis
Program Studi S2 Teknik Informatika
Program Pascasarjana Universitas AMIKOM Yogyakarta
pada hari senin, 3 juli 2023

Tesis ini telah diterima sebagai salah satu persyaratan
untuk memperoleh gelar Magister Komputer

Yogyakarta, 3 juli 2023
Rektor

Prof. Dr. M. Suyanto, M.M.
NIK. 190302001

HALAMAN PERSETUJUAN

KLASIFIKASI DETEKSI PENGGUNAAN MASKER MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK

CLASSIFICATION OF MASK USAGE DETECTION USING CONVOLUTIONAL NEURAL NETWORK METHOD

Dipersiapkan dan Disusun oleh

Nalda Kresimo Negoro

21.52.1019

Telah Diujikan dan Dipertahankan dalam Sidang Ujian Tesis
Program Studi S2 Teknik Informatika
Program Pascasarjana Universitas AMIKOM Yogyakarta
pada hari senin, 3 juli 2023

Pembimbing Utama	**Anggota Tim Penguji**
Prof. Dr. Ema Utami, S.Si., M.Kom.	**Dr. Andi Sunyoto, M.Kom.**
NIK. 190302035	**NIK. 190302052**
	Dr. Kumara Ari Yuana, S.T., M.T.
Pembimbing Pendamping	**NIK. 190302575**
Ainul Yaqin, M.Kom.	**Prof. Dr. Ema Utami, S.Si., M.Kom.**
NIK. 190302255	**NIK. 190302035**

Tesis ini telah diterima sebagai salah satu persyaratan
untuk memperoleh gelar Magister Komputer

Yogyakarta, 3 juli 2023
Direktur Program Pascasarjana

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

HALAMAN PERNYATAAN KEASLIAN TESIS

HALAMAN PERNYATAAN KEASLIAN TESIS

Yang bertandatangan di bawah ini,

Nama mahasiswa : Nalda Kresimo Negoro
NIM : 21.52.1019
Konsentrasi : Business Intelligence

Menyatakan bahwa Tesis dengan judul berikut:
KLASIFIKASI DETEKSI PENGGUNAAN MASKER MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK

Dosen Pembimbing Utama : Prof. Dr. Ema Utami, S.Si., M.Kom.
Dosen Pembimbing Pendamping : Ainul Yaqin, 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, tanggal ujian tesis
Yang Menyatakan,

Nalda Kresimo Negoro

HALAMAN PERSEMBAHAN

Puji syukur atas kehadirat Allah SWT yang telah melimpahkan rahmat dan karunianya sehingga penulis dapat menyelesaikan tesis ini, dan tesis ini dipersembahkan kepada :

1. Kepada kedua orangtua yang selalu mendoakan dan selalu memberi motivasi dalam segala hal.
2. Saudara dan teman-teman penulis yang selalu memeberikan motivasi, terima kasih atas dukungan dan doa kalian.
3. Dosen Pembimbing Prof. Dr. Ema Utami, S.Si., M.Kom. dan Ainul Yaqin, M.Kom. Terimakasih selalu memberikan bimbingan dan arahan terbaik kepada penulis, sehingga dapat menyelesaikan tesis ini.
4. Seluruh dosen dan staff dari Universitas Amikom Yogyakarta yang telah mengajar dan mendidik serta memberikan banyak hal pengalaman kepada penulis.
5. Teman-teman MTI angkatan 21 kelas A dan B Teknik Informatika, terima kasih atas segalanya.

HALAMAN MOTTO

"Sesungguhnya sesudah kesulitan itu ada kemudahan." – (QS. Al-Insyirah: 6)

"Orang optimis akan selalu melihat peluang dalam kesempitan"

(Nalda Kresimo Negoro)

KATA PENGANTAR

Puji syukur atas kehadirat Allah SWT yang telah melimpahkan rahmat dan hidayahnya sehingga penulis dapat menyelesaikan tesis yang berjudul "KLASIFIKASI DETEKSI PENGGUNAAN MASKER MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK".

Pembuatan tesis ini disusun sebagai salah satu syarat untuk menyelesaikan pendidikan jenjang S2 di Universitas Amikom Yogyakarta. Dengan selesainya tesis ini berkat dukungan dan bimbingan dari berbagai pihak. Oleh karena itu penulis mengucapkan terima kasih kepada :

1. Sujud sembah syukur kepada Allah SWT yang telah memberikan kesehatan, kekuatan, dorongan, semangat, rahmat, hidayah, rezeki dan semua yang penulis butuhkan. Allah SWT sutradara terhebat.
2. Kepada kedua orang tua dan keluarga yang selama ini mendukung dan memberi doa sehingga dapat menyelesaikan tesis ini dengan baik.
3. Terima kasih kepada dosen pembimbing Prof. Dr. Ema Utami, S.Si., M.Kom. dan Ainul Yaqin, M.Kom. yang telah membimbing dan mengarahkan penulis, sehingga dapat terselesaikannya tesis ini dengan baik.
4. Terima kasih kepada teman-teman dari jurusan S2 Teknik Informatika Universitas AMIKOM Yogyakarta yang banyak berbagi ilmu dan pengalaman-pengalaman mengesankan.
5. Bapak Ibu Dosen Universitas AMIKOM Yogyakarta yang telah ikhlas dan selalu sabar memberikan ilmu.

Yogyakarta, 3 juli 2023

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

INTISARI

Berhubungan dengan era revolusi industri 5.0 kita perlu bersyukur semua pekerjaan menjadi dimudahkan dengan terdigitalisasi. Berbagai pekerjaan dapat diselesaikan jauh lebih mudah, cepat dan secara otomatis. Konsep era industri 5.0 memiliki fokus pendayagunaan aspek dari manusia, data dan teknologi berbasis modern. Manusia dan sistem saling terhubung dan mendapatkan hasil maksimal dengan bantuan AI. Konsep ini memberikan dampak positif untuk menghadapi perubahan besar pada transformasi digital. Perkembangan pesat dari transformasi digital saat ini ada pada pendeteksian objek menggunakan machine learning. Khususnya pada beberapa penelitian pendeteksian objek sebagai klasifikasi wajah bermasker. Pada penelitian ini diterapkan klasifikasi deteksi objek dengan algoritma *Convolutional Neural Network* (CNN) dengan mengklasifikasikan wajah bermasker dan tidak bermasker. Dataset yang digunakan untuk proses training diperoleh dari kaggle berjumlah 2.750 data gambar. 1.380 data gambar menggunakan masker, 1.370 tidak menggunakan masker. Agar mendapatkan model terbaik penelitian ini menggunakan empat arsitektur sebagai perbandingan yaitu VGG16, ResNet50V2, Xception, dan EfficientNetV2S. Penelitian ini menyimpulkan arsitektur EfficientNetV2S mendapatkan nilai kinerja tertinggi dengan nilai *accuracy* 99,82%.

Kata kunci: *Artificial Neural Network, Convolutional Neural Network, Face Mask Detection,* Klasifikasi, *Machine Learning*

ABSTRACT

In connection with the era of the Industrial Revolution 5.0, we need to be grateful that all work has been made easy by being digitized. Various jobs can be completed much easier, faster, and automatically. The concept of the industrial era 5.0 has a focus on utilizing aspects of humans, data, and modern-based technology. Humans and systems are interconnected, and we can get the most out of it with the help of AI. This concept has a positive impact on facing major changes in digital transformation. The rapid development of digital transformation today is in object detection using machine learning. Especially in several studies of object detection as a classification of masked faces. In this study, object detection classification was applied using the Convolutional Neural Network (CNN) algorithm by classifying masked and non-masked faces. The dataset used for the training process was obtained from Kaggle with a total of 2,750 image files. 1,380 image data points use a mask, 1,370 do not use a mask. In order to get the best model, this research uses four architectures as a comparison, namely VGG16, ResNet50V2, Xception, and EfficientNetV2S. This study concludes that the EfficientNetV2S architecture gets the highest performance value with an accuracy value of 99.82%.

Keyword: Artificial Neural Network, Convolutional Neural Network, Face Mask Detection, Classification, Machine Learning

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Berhubungan dengan masuknya revolusi era industri 5.0 kita perlu bersyukur membuat semua pekerjaan menjadi lebih dimudahkan dengan hadirnya kemudahan yang diberikan dengan serba terdigitalisasi. Berbagai hal yang dikerjakan dapat diselesaikan dengan jauh lebih mudah, lebih cepat dan dapat diselesaikan secara otomatis. Revolusi era industri 5.0 yang digagas oleh negara Jepang ini bisa disebut sebagai era *society* 5.0 dan telah diresmikan pada 21 januari 2019. Konsep dari era industri 5.0 memiliki fokus terhadap kombinasi antara pendayagunaan antara aspek dari manusia, data serta kemajuan dari teknologi yang ada sampai saat ini dengan menggunakan ilmu pengetahuan yang berbasis modern seperti IoT, AI dan Robot (Universitas Medan Area, 2022). Walaupun tidak memiliki perbedaan yang jauh dengan era industri sebelumnya yaitu era industri 4.0, namun keduanya memiliki fokus yang berbeda. Secara sederhana pada era industri 4.0 yaitu tentang mengumpulkan informasi melalui jaringan dan manganalisanya, berbeda dengan fokus era industri 5.0 yaitu manusia dan sistem akan saling terhubung dan mendapatkan hasil yang maksimal dengan adanya bantuan dari *Artificial Intelligence* (AI). Hasil yang telah didapatkan tersebut kemudian diumpankan kembali ke ruang fisik (*real space*). Dengan hadirnya konsep ini memberikan dampak positif untuk menghadapi perubahan besar didunia dan pada transformasi digital dengan munculnya teknologi seperti AI, IoT, *Big*

Data, Data Science, Computer Vision, Robotika, *Machine Learning, Deep Learning,* sampai pada pengolahan data dapat memberikan kemudahan dalam kehidupan manusia (Nisa, 2022).

Salah satu bidang yang saat ini sedang berkembang pesat pada era industri 5.0 yaitu deteksi objek secara otomatis oleh sistem. Deteksi objek adalah salah satu teknik dari *computer vision* dalam melakukan pembacaan atau pengenalan objek pada gambar ataupun video. Penerapan dari *computer vision* yang dapat digunakan diberbagai bidang dapat memberikan kemudahan bagi manusia dalam membantu pekerjaan. *Computer vision* merupakan salah satu bagian dari *artificial intelligence* (AI) yang memungkinkan komputer untuk dapat melihat dan melakukan pengenalan hingga deteksi objek yang ada disekitarnya layaknya cara kerja pada mata manusia dan mamalia. Dalam perkembangan teknologi yang begitu pesat ini *computer vision* menjadi salah satu teknologi yang dapat dimanfaatkan dan dapat dikombinasikan dengan *data science* dengan bantuan *machine learning* hingga *deep learning* (Nursyafitri, 2022). Berdasarkan dari tujuannya computer vision memiliki beberapa jenis tujuan pada penerapannya, diantaranya *image segmentation, object detection, facial recognition, image classification, edge detection, pattern detection* dan *feature matching*.

Penerapan *computer vision* begitu booming pada beberapa tahun terakhir ini mengingat pada penerapannya memberikan kemudahan dan sangat membantu berbagai kegiatan manusia diberbagai bidang. Mulai dari pengembangan produk, kesehatan, operasional, pengawasan & keamanaan, dan keperluan lainnya. Beberapa contoh penerapan dari penggunaan *computer vision* yaitu pada bidang

pengembangan produk seperti mobil Tesla yang pada penggunaannya dapat berjalan tanpa bantuan kemudi (*autopilot*), pada bidang kesehatan untuk alat pendeteksi penyakit dari gambar CT-scan atau citra Xray, hingga alat pendeteksi penyakit tanaman dari gambar/foto daun yang terserang hama. Kemudian pada bidang operasional, keamanan dan pengawasan lingkungan yaitu deteksi objek (Yudianto, 2021). Berbicara tentang deteksi objek yang saat ini sedang mengalami perkembangan yang begitu pesat. Menurut Ivan Vasilev pada tahun 2019, deteksi objek merupakan proses dalam menentukan instance objek dari kelas tertentu seperti mobil, pohon dan wajah dalam gambar atau video. Jika klasifikasi adalah melakukan pembacaan pada satu objek saja dalam sebuah gambar/frame, deteksi objek dapat digunakan untuk mendeteksi banyak objek pada sebuah gambar/frame video. Detektor objek akan mengembalikan daftar objek yang telah terdeteksi dengan memberikan informasi kelas objek, probabilitas dan koordinatnya untuk setiap posisi objek yang terdeteksi pada gambar (Salim, 2020).

Namun, untuk dapat mengenali objek pada suatu gambar maupun video, sebuah sistem memerlukan dataset yang perlu dilatih terlebih dahulu terkait dengan kelas objek tersebut. Sistem akan memperlajari dataset tersebut untuk menemukan ciri khusus dari suatu objek tersebut yang dapat dijadikan pola oleh sistem dalam mengenali objek yang sama pada kondisi yang berbeda. Proses dari pembelajaran sistem dengan dataset tersebut termasuk pada bidang keilmuan *machine learning* atau *deep learning* (Yudianto, 2021). Terdapat banyak algoritma *machine learning* atau *deep learning* yang dapat digunakan dalam proses pembelajaran sistem terhadap objek citra diantaranya yaitu CNN, Faster R-CNN, YOLO, MobileNet,

Mask R-CNN dan sebagainya. Seperti pada penelitian sebelumnya yang dilakukan oleh (Rahim, 2021) yaitu melakukan analisis komparasi untuk klasifikasi citra pengguna masker dengan membandingkan kinerja dari metode yang digunakan yaitu metode CNN dibandingkan dengan kombinasi metode CNN dan *Support Vector Machine* (SVM). Agar mendapatkan model terbaik penelitian berikut menggunakan tiga arsitektur yaitu VGG16, ResNet50 dan MobileNet. Kesimpulan dari penelitian ini didapatkan nilai kinerja tertinggi yaitu dengan mengkombinasikan CNN dan ResNet50 didapatkan nilai kinerja tertinggi dengan *accuracy* 99,41%.

Berdasarkan dari penelitian sebelumnya, tidak begitu mengherankan jika algoritma CNN banyak digunakan pada proses klasifikasi citra. Beberapa penelitian lainnya yang dilakukan oleh (Putri, Fikih, & Setyawan, 2020) menggunakan metode algoritma yang sama yaitu dengan metode CNN untuk memanfaatkan ekstraksi fitur dari citra yang selanjutnya dapat dipelajari oleh beberapa *hidden layer*. Sistem yang dibangun pada penelitian ini menggunakan kombinasi klasifikasi deteksi gambar, objek serta pelacakan objek. Hingga dapat dikembangkannya sistem untuk mendeteksi wajah bermasker dan tidak bermasker dalam bentuk gambar dan video. Pengujian sistem pada penelitian berikut mendapatkan nilai akurasi sebesar 0,9933% atau 99,33%.

Namun dari penelitian yang dilakukan oleh (Budiman, Lubis, & Perdana, 2021) yaitu mendeteksi penggunaan masker wajah menggunakan metode CNN untuk dapat mengklasifikasi manusia apakah menggunakan masker atau tidak dengan akses melalui kamera *webcam*. Program tersebut dibuat menggunakan

bahasa pemrograman python dan metode CNN dengan arsitektur MobileNetV2 sedangkan pendeteksian wajah manusia menggunakan *Haarcascade Classifier*. Dari hasil pengujiannya pada penelitian ini didapatkan nilai kinerja dengan *accuracy* rata-rata 88,53% untuk deteksi objek dan klasifikasi penggunaan masker rata-rata didapatkan nilai akurasi 84,45%.

Dalam pengembangannya menurut (Dharmadi, Convolutional Neural Net untuk Deteksi Objek, 2018) CNN dipublikasikan oleh peneliti dari New York University pada tahun 2013. Dalam pembahasan berikut CNN untuk keperluan *object detection* memiliki teknik populer yaitu *OverFeat* merupakan arsitektur yang menggunakan *deep learning* yang berguna untuk mendeteksi objek. Teknik OverFeat ini dapat memanfaatkan algoritma CNN dan *multi-scale sliding window*. Namun, permasalahan pada penggunaan teknik *sliding window* yaitu akan menciptakan banyaknya potongan gambar yang harus diproses oleh algoritma CNN. Pada setiap potongan gambar yang perlu diproses harus melewati proses konvolusi kemudian diklasifikasikan menjadi objek dan background. Maka dari itu, banyaknya dari gambar dan ukuran *sliding window* akan melewati pemrosesan komputasi keseluruhan yang sangat berat. Setahun kemudian pada tahun 2014 muncul teknik *Regions with CNN features* atau bisa disebut dengan R-CNN. Pengembangan algoritma ini diperkenalkan oleh Ross Girshick, et al. Yaitu peneliti dari UC Berkeley. Proses dari algoritma R-CNN memiliki 3 tahap yaitu mencari *region* pada bagian gambar yang kemungkinan dibaca sebagai objek dengan menggunakan metode *region proposal*. Kemudian tahap ke 2 setiap *region* tersebut digunakan dan dijadikan input untuk algoritma CNN sebagai *feature extractors*.

Tahap ke 3 setiap fitur yang dihasilkan akan dijadikan input untuk SVM dan menghasilkan kelas dari *region* tersebut serta *linear regressor*. Dengan ketiga tahapan proses yang dilakukan maka R-CNN berhasil meningkatkan performa *OverFeat* yang dapat mencapai 50%. Walaupun memiliki peningkatan yang begitu signifikan, Pada tahun 2015 algoritma ini kembali dikembangkan lagi oleh Ross Girschick dari algoritma R-CNN menjadi Fast R-CNN dan tidak lama kemudian muncul pengembangan baru lagi yaitu metode Faster R-CNN yang dikembangkan oleh peneliti yang sama yaitu Ross Girschick dan dibantu oleh Shaoqing Ren. Metode Faster R-CNN telah diklaim dapat menghasilkan performa lebih cepat dan lebih akurat dibandingkan dengan metode yang telah dikembangkan sebelumnya.

Dari penelitian yang dilakukan oleh (Permana, 2021) dari hasil penelitian yang dilakukan setelah training data yang menggunakan dataset total 750 gambar wajah bermasker dan tidak bermasker, sistem yang dikembangkan dapat mengklasifikasikan menjadi 3 kelas yaitu tidak memakai masker, memakai masker dan memakai masker tapi tidak sesuai dengan aturan. Setelah melalui proses testing menggunakan metode Faster R-CNN mendapatkan hasil nilai kinerja dengan tingkat akurasi 89,94%. Hal ini menunjukan bahwa metode Faster R-CNN yang telah digunakan pada penelitian berikut belum tentu bisa melampaui keakuratan dari metode yang dikembangkan sebelumnya yaitu metode CNN. Tapi berdasar dari hasil penelitian ini, penulis juga menyebutkan bahwa penelitian masih memerlukan pengembangan lagi khususnya untuk menambah dataset yang digunakan agar lebih variatif agar dapat meningkatkan tingkat akurasi yang lebih akurat.

Adanya penelitian lain yaitu penelitian pendeteksian penggunaan masker menggunakan metode yang berbeda dari metode CNN yang diterapkan oleh (Muharram, 2021) melalui penelitian deteksi masker menggunakan tensorflow dan SSD MobileNet berbasis python. Pada penelitian berikut menunjukan hasil pengujian dengan mendapatkan nilai kinerja tertinggi dengan *accuracy* 100% dan nilai kinerja terendah dengan *accuracy* 96%, Jika dirata-rata hasil pengujian berikut mendapatkan nilai kinerja dengan nilai akurasi 97,5%. Pada hasil penelitian ini dapat dilihat dari kinerja algoritma yang dapat menghasilkan nilai kinerja yang tinggi walaupun juga disebutkan jika dalam pengujiannya masih terdapat *false detection* dan pendeteksian *overlap* dapat disebabkan oleh kualitas pada sampel dataset yang digunakan masih belum optimal dengan pembacaan nilai akurasi 75%. Kesimpulannya yang dapat diambil dari penelitian ini disebutkan bahwa dataset memiliki peran sangat penting dalam pengembangan model. Karena peran dari dataset akan berpengaruh pada model yang dilatih agar memiliki banyak opsi untuk mengenal objek yang bervariasi.

Berdasarkan dari uraian diatas dalam melakukan pertimbangan kinerja dari model yang berdasarkan nilai akurasi. Pada metode/algoritma yang digunakan yaitu pengembangan dari CNN memiliki nilai kinerja akurasi yang tinggi. Terlebih dari penelitian yang dilakukan oleh (Rahim, 2021) yang melakukan analisis komparasi dengan kesimpulan hasil pengujian menggunakan metode algoritma CNN dengan ResNet50 didapatkan nilai akurasi 99,41%. Namun referensi dari (Dharmadi, Convolutional Neural Net untuk Deteksi Objek, 2018) telah menuliskan metode pengembangan mulai dari CNN, R-CNN dan Faster R-CNN disebutkan bahwa

metode Faster R-CNN memiliki performa lebih akurat jika dibandingkan dengan metode pengembangan sebelumnya. Tetapi dari referensi yang telah didapatkan sesuai dengan penelitian yang dilakukan oleh (Permana, 2021) dengan metode Faster R-CNN nilai akurasi mencapai 89,94%. Pada penelitian tersebut dapat dibuktikan bahwa metode Faster R-CNN belum tentu mendapatkan kinerja nilai akurasi yang lebih tinggi dibandingkan dengan pengembangan metode sebelumnya.

Penelitian yang dilakukan menurut (Harjoseputro, 2018) CNN selalu dipilih karena menjadi metode *state of the art* dan secara *de facto* menjadi metode yang selalu digunakan menjadi pilihan untuk menjawab berbagai permasalahan pada *computer vision*. Selain melakukan pertimbangan dari kinerja model berdasarkan dengan nilai akurasi, waktu komputasi juga perlu dipertimbangkan mengingat saat melakukan pelatihan model pada metode CNN menggunakan dataset dalam jumlah banyak akan memberikan beban komputasi pada proses pembelajaran.

Seperti penelitian yang dilakukan oleh (Halim, 2022) penelitian yang dilakukan menggunakan metode CNN dan arsitektur Xception. Dari penelitian ini diketahui performa arsitektur yang digunakan dapat memberikan efisiensi komputasi lebih baik. Total data citra yang digunakan 7300 data mendapatkan hasil kinerja akurasi, sensitivitas, spesifisitas, dan *f1-score* masing-masing mendapatkan hasil 87,58%, 81,43%, 90,73%, dan 81,37%. Dari penelitian berikut membuktikan bahwa arsitektur yang digunakan dapat memberikan efisiensi komputasi dalam pelatihan model. Selain itu penelitian lain juga menyebutkan dari (Tan & Le, 2021) dengan meneliti arsitektur lebih baru yaitu EfficientNetV2 yang dirilis pada tahun

2021. Menyebutkan bahwa arsitektur tersebut lebih efisien karena memiliki model yang lebih kecil dengan waktu komputasi yang lebih cepat.

Dari beberapa uraian diatas dan dikarenakan keingintahuan dari penulis untuk melakukan penelitian terkait klasifikasi dan pendeteksian penggunaan masker berbekal dengan membaca beberapa penelitian sebelumnya maka penulis ingin melakukan penelitian dengan metode CNN dengan melakukan komparasi menggunakan beberapa arsitektur yang beberapa telah digunakan untuk penelitian terdahulu seperti VGG16, ResNet50V2, Xception, dan EfficientNetV2S. Metode CNN dengan beberapa arsitektur yang dikomparasikan tersebut akan diterapkan supaya dapat mengukur nilai kinerja *accuracy* tertinggi yang didapatkan dari metode tersebut. Selain itu penelitian ini juga akan mengukur kinerja algoritma dari nilai kinerja *precision*, *recall*, dan waktu komputasi.

1.2. Rumusan Masalah

Berdasarkan uraian dari latar belakang yang telah dijabarkan diatas didapatkan poin-poin rumusan masalah sebagai berikut :

a. Berapa nilai *accuracy*, *precision* dan *recall* yang didapatkan dari pelatihan algoritma CNN untuk klasifikasi deteksi citra wajah menggunakan masker dan tidak menggunakan masker?
b. Apakah penggunaan arsitektur lebih baru dapat memberikan hasil lebih baik dari segi nilai akurasi dan waktu komputasi pada pelatihan model?
c. Jika arsitektur lebih baru dapat memberikan performa hasil lebih baik, parameter apa yang membuat arsitektur tersebut memberikan performa hasil lebih baik?

1.3. Batasan Masalah

Berdasarkan rumusan masalah diatas, maka penelitian yang dilakukan memiliki batasan masalah sebagai berikut :

a. Algoritma yang digunakan untuk klasifikasi deteksi masker adalah CNN.
b. Penelitian ini akan mengklasifikasikan wajah yang menggunakan masker dan tidak menggunakan masker.
c. Pembuatan implementasi model dengan menggunakan algoritma CNN dibuat dengan menggunakan bahasa pemrograman *Python* dengan antarmuka dan infrastruktur *Google Colaboratory*.
d. Dataset yang akan digunakan untuk pembuatan model diperoleh dari kaggle.
e. Evaluasi jaringan dari hasil pelatihan model klasifikasi deteksi penggunaan masker yaitu menggunakan *confusion matrix*.

1.4. Tujuan Penelitian

Tujuan penelitian ini yang akan dicapai adalah sebagai berikut :

a. Mengetahui nilai *accuracy, precision,* dan *recall* dari algoritma CNN untuk klasifikasi deteksi wajah menggunakan masker dan tidak menggunakan masker.
b. Mengetahui penggunaan arsitektur lebih baru dapat memberikan hasil lebih baik atau tidak dari segi nilai akurasi dan waktu komputasi pada pelatihan model.
c. Mengetahui parameter apa yang membuat arsitektur tersebut memberikan performa hasil lebih baik.

1.5. Manfaat Penelitian

Manfaat yang didapatkan dari penelitian ini yaitu sebagai berikut :

a. Rancangan arsitektur yang diterapkan dapat dijalankan dan dapat melakukan klasifikasi deteksi penggunaan masker dengan mendapatkan nilai akurasi yang baik sesuai dengan batasan yang telah ditentukan.
b. Bagi peneliti dapat mengetahui nilai tingkat akurasi tertinggi dari penggunaan algoritma CNN.
c. Memberikan kontribusi penelitian terhadap arsitektur CNN dan dapat menghasilkan nilai akurasi yang tinggi.
d. Dapat memberikan pengetahuan baru mengenai algoritma atau metode yang dapat melakukan klasifikasi wajah yang menggunakan masker dan tidak menggunakan masker.

BAB II

TINJAUAN PUSTAKA

2.1. Tinjauan Pustaka

Terdapat beberapa penelitian deteksi penggunaan masker saat ini semakin berkembang pada beberapa tahun terakhir ini. Sehingga pada era industri / era *society* 5.0 yang serba terdigitalisasi maka mendorong berbagai penelitian khususnya pada bidang *computer vision* dan *machine learning*, terlebih pada deteksi objek seperti deteksi penggunaan masker untuk keperluan keamanan kesehatan dan pengawasan dilingkungan sekitar dalam aktifitas sehari-hari. Penelitian yang dilakukan oleh (Kocacinar, Tas, Akbulut, Catal, & Mishra, 2022) memiliki tujuan yaitu mengidentifikasi individu yang tidak menggunakan masker atau menggunakan masker tetapi tidak digunakan dengan tidak benar dan untuk memverifikasi identitas dengan membangun dataset wajah bermasker. Layanan deteksi masker menggunakan aplikasi mobile dikembangkan dengan algoritma Convolutional Neural Network (CNN). Dalam penelitian ini data diperoleh dari dua set data yaitu VGGFace2 dan MaskedFace-Net. VGGFace2 berisi gambar lebih dari 9000 individu dari berbagai etnis, profesi, dan usia. Lebih dari 3,3 juta foto telah direkam secara total. Setiap individu memiliki sekitar 362 foto. Kumpulan data MaskedFace-Net berisi 67.193 gambar individu yang memakai masker dan 69.823 gambar individu yang menggunakan masker secara tidak benar. Semua gambar tersebut di sortir menjadi 12 subjek sampel wajah dari total 1849 gambar. Pengujian dilakukan menggunakan arsitektur MobileNet, VGG16, dan ResNet. Hasil dari

penelitian ini mendapatkan performa terbaik dari arsitektur MobileNet dengan epoch 10 dan batch size 8 didapatkan validasi akurasi 90,40%, arsitektur VGG16 dengan epoch 400 dan batch size 32 didapatkan validasi akurasi 87,60%, dan arsitektur ResNet dengan epoch 20 dan batch size 16 didapatkan validasi akurasi 51,74%.

Penelitian yang dilakukan oleh (Shamrat, et al., 2021) dataset yang diambil dan digunakan sebagai data training dari sumber Real-World Masked Face Dataset (RMFD) dan Simulated Masked Face Dataset (SMFD). Total dataset yang digunakan yaitu 1845 gambar. Data training tersebut kemudian diproses dengan menggunakan data image generator dengan mengubah ukuran semua gambar menjadi 256 x 256 piksel agar proses perhitungan saat pelatihan model menjadi lebih cepat kemudian diolah menggunakan data augmentasi dengan merotasi, *zooming, shearing* dan *horizontal flipping*. Setelah itu gambar diubah ukurannya menjadi 128 x 128 untuk melewati lapisan konvolusi kedua dan diubah ke 64 x 64 untuk diproses ke lapisan konvolusi ketiga. Pada penelitian ini skenario yang digunakan yaitu skenario pertama menggunakan lapisan konvolusi dengan max pooling, skenario kedua menggunakan lapisan konvolusi dengan average pooling, dan skenario ketiga menggunakan arsitektur dari MobileNetV2. Hasil dari penelitian ini yaitu skenario pertama mendapatkan validasi akurasi 98,49%, skenario kedua 96,23% dan skenario ketiga 99,82%.

Penelitian yang dilakukan oleh (Zhang, Tang, Wu, Li, & Zeng, 2023) yaitu melakukan analisis komparasi untuk klasifikasi citra pengguna masker dengan membandingkan kinerja dari AI-Yolo dengan arsitektur pendeteksian objek lainnya

seperti VGG16, ResNet50, dan EfficientNet. Metode yang digunakan yaitu metode Faster R-CNN. Perbandingan ini yaitu mencari perbedaan hasil kinerja dari model dengan dataset gambar yang didapatkan dari Kaggle dengan link : https://www.kaggle.com/datasets/andrewmvd/face-mask-detection. Dataset yang digunakan yaitu 8318 gambar wajah menggunakan masker dan 7820 gambar wajah tidak menggunakan masker. Hasil dari penelitian berikut yaitu metode Faster R-CNN dengan arsitektur VGG16 mendapatkan nilai rata-rata 0,846 atau 84,6%, arsitektur ResNet50 mendapatkan nilai 0,832 atau 83,2%, dan arsitektur EfficientNet-B0 mendapatkan nilai 0,845 atau 84,5%. Sedangkan untuk AI-Yolo mendapatkan nilai 0,941 atau 94,1%. Kesimpulan dari penelitian ini didapatkan nilai kinerja tertinggi yaitu AI-Yolo didapatkan nilai kinerja tertinggi dengan *accuracy* 0,941 atau 94,1%.

Penelitian lain terkait dengan deteksi penggunaan masker yang diteliti oleh (Agarwal, Itondia, & Mishra, 2023). Pada penelitian ini berfokus pada pengembangan kerangka kinerja dari metode DCNN (*Depp Convolutional Neural Network*) dikombinasikan dengan ELM (*Extreme Learning* Machine) untuk deteksi penggunaan masker. Arsitektur terdahulu yang digunakan yaitu VGG16, VGG19, ResNet50, ResNet101, ResNet152, Xception. Hasil yang diperoleh dari dataset 4653 citra yang digunakan untuk pelatihan model didapatkan hasil akurasi dengan arsitektur Xception mendapatkan nilai 97,74%, VGG16 dengan nilai 99,14%, VGG19 dengan nilai 98,92%, ResNet50 dengan nilai 99,03%, ResNet101 dengan nilai 99,57%, dan akurasi tertinggi didapatkan oleh arsitektur ResNet152 dengan nilai 99,78%.

Selanjutnya penelitian lain terkait yang dilakukan oleh (Ramadhan, et al., 2023) memiliki tujuan membuat sistem yang dapat melakukan pengenalan dalam penggunaan masker untuk menerapkan protokol kesehatan, Penelitian ini menggunakan arsitektur seperti VGG11, ResNet50, InceptionV3, EfficientNetB4, dan YOLO. Pada penelitian ini dataset yang digunakan yaitu dari MaskedFace-Net dengan jumlah 1000 gambar dengan 500 gambar bermasker dan 500 gambar tidak bermasker. Dataset dilatih menggunakan *epoch* 16 dan *batch size* 4 untuk arsitektur VGG11, ResNet50, InceptionV3, EfficientNetB4 selain itu untuk arsitektur YOLOv4 menggunakan *batch size* 64. Hasil uji coba dilakukan menghasilkan nilai akurasi VGG11 84,38% memerlukan estimasi waktu training 31 menit, arsitektur ResNet50 mendapatkan nilai 84,41% dengan waktu 25 menit, InceptionV3 mendapatkan nilai 87,30% dengan waktu 4 jam 41 menit, EfficientNetB4 mendapatkan nilai tertinggi 95,77% dengan waktu 52 menit, dan YOLOv4 mendapatkan nilai 93,4% dengan waktu 3 jam 45 menit.

Penelitian lain terkait yang ditulis oleh (Naufal & Kusuma, 2021) memiliki tujuan yaitu dapat melakukan pendeteksian citra wajah menggunakan beberapa arsitektur CNN yaitu MobileNetV2, VGG16, DenseNet201, dan Xception yang dikombinasikan dengan transfer learning untuk mendeteksi wajah tidak bermasker. Kombinasi yang dilakukan tersebut menggunakan datset/database yang didapatkan dari kaggle Larxel pada tahun 2020 dengan jumlah data gambar bermasker 3725 dan data gambar tidak bermasker 3828 digunakan sebagai pelatihan model. Hasil dari penelitian ini dari arsitektur Xception mendapatkan nilai akurasi terbaik dengan validasi akurasi dan waktu komputasi 0.988 dan 18.274 detik. Arsitektur

MobileNetV2 yang dikombinasikan dengan transfer learning memiliki waktu total komputasi tercepat dapat mengklasifikasikan citra masker wajah dengan akurasi dan waktu komputasi masing-masing 0.981 dan 4.081 detik.

2.2. Keaslian Penelitian

Tabel 2.1. Matriks literatur review dan posisi penelitian

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
1	A Real-Time CNN-Based Lightweight Mobile Masked Face Recognition System	Busra Kocacinar, Bilal Tas, Fatma Patlar Akbulut, Cagatay Catal, Deepti Mishra., IEEE, 2022	Mengidentifikasi individu yang tidak menggunakan masker atau menggunakannya secara tidak benar dan untuk memverifikasi identitas dengan membangun dataset wajah bermasker. Layanan deteksi masker menggunakan aplikasi mobile dikembangkan dengan algoritma CNN.	Pengenalan wajah yang dikembangkan menggunakan metode CNN dapat disesuaikan dengan baik. Arsitektur yang diusulkan menggunakan MobileNet, VGG16, dan ResNet. Didapatkan performa terbaik dari arsitektur MobileNet mencapai validasi akurasi 90,40%. Pengujian dilakukan dengan menggunakan 12 sampel wajah dari 1849 individu.	Peneliti : 1. Penelitian dilakukan pada tahun 2022 namun tidak mengusulkan menggunakan arsitektur yang lebih baru seperti EfficientNet.	Tindak lanjut : 1. Pada penelitian ini mengusulkan menggunakan arsitektur lebih baru diantaranya yaitu EfficientNetV2S.
2	Face Mask Detection using Convolutional Neural Network (CNN) to reduce the spread of Covid-19	F.M. Javed Mehedi Shamrat; Md. Masum Billah; Md Saidul Islam; Sovon Chakraborty;	Melakukan klasifikasi deteksi penggunaan masker wajah menggunakan metode CNN dengan 3 skenario	Hasil dari penelitian ini yaitu skenario pertama mendapatkan validasi akurasi 98,49%, skenario kedua 96,23% dan skenario ketiga arsitektur MobileNetV2	Peneliti : 1. Tidak melakukan komparasi pengujian dengan menggunakan arsitektur berbeda. Pengujian hanya	Tindak lanjut : 1. Melakukan komparasi pengujian skenario dengan menggunakan 4 arsitektur berbeda yaitu VGG16, ResNet50V2,

Tabel 2.1. Matriks literatur review dan posisi penelitian (Lanjutan)

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
		Md. Al Jubair; Rumesh Ranjan., International Conference on Trends in Electronics and Informatics, 2021	berbeda. Skenario pertama menggunakan Max pooling, kedua Average pooling, dan ketiga arsitektur MobileNetV2. Dengan menggunakan dataset 1845 gambar yang digunakan sebagai pelatihan model.	mendapatkan nilai validasi akurasi tertinggi yaitu 99,82%.	dilakukan menggunakan arsitektur MobileNetV2. 2. Tidak menunjukan estimasi waktu komputasi yang diperlukan dalam proses pelatihan model.	Xception, dan EfficientNetV2S. 2. Menunjukan estimasi waktu komputasi yang diperlukan untuk proses pelatihan model disetiap skenario pelatihannya.
3	A novel attention-based enhancement framework for face mask detection in complicated scenarios	Hongyi Zhang, Jun Tang, Peishu Wu, Han Li, Nianyin Zeng., Signal Processing: Image Communication (Q2), 2023	Melakukan analisis komparasi untuk klasifikasi citra pengguna masker dengan membandingkan kinerja dari AI-Yolo dengan arsitektur pendeteksian objek lainnya seperti VGG16, ResNet50, dan EfficientNet. Metode yang digunakan yaitu metode Faster R-CNN. Dataset yang	Hasil dari penelitian berikut yaitu metode Faster R-CNN dengan arsitektur VGG16 mendapatkan nilai rata-rata 0,846 atau 84,6%, arsitektur ResNet50 mendapatkan nilai 0,832 atau 83,2%, dan arsitektur EfficientNet-B0 mendapatkan nilai 0,845 atau 84,5%. Sedangkan untuk AI-Yolo mendapatkan nilai 0,941 atau 94,1%.	Peneliti : 1. Penggunaan metode lebih baru namun beberapa penelitian mengungkapkan bahwa metode Faster R-CNN belum tentu mendapatkan hasil lebih optimal dari pada metode CNN. 2. Penelitian dilakukan tahun 2023 tidak dikomparasikan dengan arsitektur lebih baru seperti	Tindak lanjut : 1. Menggunakan metode CNN yang diharapkan dapat memberikan hasil evaluasi pelatihan model dalam klasifikasi deteksi masker lebih optimal. 2. Melakukan komparasi menggunakan arsitektur yang lebih baru untuk skenario pengujiannya. Seperti arsitektur ResNet50V2 dan EfficientNetV2S.

Tabel 2.1. Matriks literatur review dan posisi penelitian (Lanjutan)

No	Judul	Pencliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
			digunakan yaitu 8318 gambar wajah menggunakan masker dan 7820 gambar wajah tidak menggunakan masker.		ResNet50V2, EfficientNetV2.	
4	A novel DCNN-ELM hybrid framework for face mask detection	Charu Agarwal, Pranjul Itondia, Anurag Mishra., Intelligent Systems with Applications (Q1), 2023	Pada penelitian ini berfokus pada pengembangan kerangka kinerja dari metode DCNN (Depp Convolutional Neural Network) dikombinasikan dengan ELM (Extreme Learning Machine) untuk deteksi penggunaan masker. Arsitektur terdahulu yang digunakan yaitu VGG16, VGG19, ResNet50, ResNet101, ResNet152, Xception.	Hasil yang diperoleh dari dataset 4653 citra yang digunakan untuk pelatihan model didapatkan hasil akurasi dengan arsitektur Xception mendapatkan nilai 97,74%, VGG16 dengan nilai 99,14%, VGG19 dengan nilai 98,92%, ResNet50 dengan nilai 99,03%, ResNet101 dengan nilai 99,57%, dan akurasi tertinggi didapatkan oleh arsitektur ResNet152 dengan nilai 99,78%.	Peneliti : 1. Penelitian dilakukan tahun 2023 tidak dikomparasikan dengan arsitektur lebih baru seperti ResNet50V2, EfficientNetV2. 2. Tidak menunjukan waktu komputasi yang dihabiskan untuk pelatihan model	Tindak lanjut : 1. Melakukan komparasi menggunakan arsitektur yang lebih baru untuk skenario pengujiannya. Seperti arsitektur ResNet50V2 dan EfficientNetV2S. 2. Menunjukan estimasi waktu komputasi yang diperlukan untuk proses pelatihan model disetiap skenario pelatihannya.

Tabel 2.1. Matriks literatur review dan posisi penelitian (Lanjutan)

No	Judul	Peneliti, Media Publikasi, dan Tahun	Tujuan Penelitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
5	Comparative analysis of deep learning models for detecting face mask	M. Vickya Ramadhan, Kahlil Muchtar, Yudha Nurdin, Maulisa Oktiana, Maya Fitria, Novi Maulina, Gregorius Natanael Elwirehardja, Bens Pardamean., Procedia Computer Science, 2023	Bertujuan membuat sistem yang dapat melakukan pengenalan dalam penggunaan masker untuk menerapkan protokol kesehatan, Penelitian ini menggunakan arsitektur seperti VGG11, ResNet50, InceptionV3, EfficientNetB4, dan YOLO. Pada penelitian ini dataset yang digunakan yaitu dari MaskedFace-Net dengan jumlah 1000 gambar dengan 500 gambar bermasker dan 500 gambar tidak bermasker.	Hasil uji coba dilakukan menghasilkan nilai akurasi VGG11 84,38% memerlukan estimasi waktu training 31 menit, arsitektur ResNet50 mendapatkan nilai 84,41% dengan waktu 25 menit, InceptionV3 mendapatkan nilai 87,30% dengan waktu 4 jam 41 menit, EfficientNetB4 mendapatkan nilai tertinggi 95,77% dengan waktu 52 menit, dan YOLOv4 mendapatkan nilai 93,4% dengan waktu 3 jam 45 menit.	Peneliti : 1. Penelitian tidak mengkomparasikan dengan arsitektur lebih baru seperti EfficientNetV2 dan arsitektur yang lebih efektif dalam waktu komputasi seperti Xception.	Tindak lanjut : 1. Melakukan komparasi menggunakan arsitektur yang lebih baru untuk skenario pengujiannya. Seperti EfficientNetV2S. Selain itu skenario percobaan menggunakan arsitetur Xception.
6	PENDETEKSI CITRA MASKER WAJAH MENGGUNAKAN CNN DAN	Mohammad Farid Naufal, Selvia Ferdiana Kusuma., Jurnal Teknologi	Melakukan pendeteksian citra wajah menggunakan beberapa arsitektur CNN yaitu	Hasil dari penelitian ini dari arsitektur Xception mendapatkan nilai akurasi terbaik dengan validasi akurasi dan waktu komputasi 0.988	Peneliti : 1. Penelitian dilakukan pada tahun 2021 namun tidak mengusulkan menggunakan	Tindak lanjut : 1. Pada penelitian ini mengusulkan menggunakan arsitektur lebih baru diantaranya yaitu EfficientNetV2S.

Tabel 2.1. Matriks literatur review dan posisi penelitian (Lanjutan)

No	Judul	Pencliti, Media Publikasi, dan Tahun	Tujuan Penclitian	Kesimpulan	Saran atau Kelemahan	Perbandingan
	TRANSFER LEARNING	Informasi dan Ilmu Komputer (JTIIK), (2021)	MobileNetV2, VGG16, DenseNet201, dan Xception yang dikombinasikan dengan transfer learning untuk mendeteksi wajah tidak bermasker. Kombinasi yang dilakukan tersebut menggunakan datset/database yang didapatkan dari kaggle Larxel pada tahun 2020 dengan jumlah data gambar bermasker 3725 dan data gambar tidak bermasker 3828 digunakan sebagai pelatihan model.	dan 18.274 detik. Arsitektur MobileNetV2 yang dikombinasikan dengan transfer learning memiliki waktu total komputasi tercepat dapat mengklasifikasikan citra masker wajah dengan akurasi dan waktu komputasi masing-masing 0.981 dan 4.081 detik.	arsitektur yang lebih baru seperti EfficientNet.	

2.3. Landasan Teori

Terdapat beberapa landasan teori yang diperlukan dalam mendukung penelitian ini, dimulai dari teori pada *computer vision*, *deep learning* dan *machine learning*, pengolahan citra digital, dataset, CNN, arsitektur CNN, serta Tensorflow.

2.3.1. Computer Vision

Computer vision adalah suatu ilmu dari kemajuan teknologi yang mampu memproses, menganalisa, dan mendeteksi suatu citra kemudian merubahnya menjadi suatu informasi bagi sistem untuk mengambil suatu keputusan. Dengan kata lain teknologi computer vision terinspirasi dari fungsi mata manusia untuk mengenali suatu objek untuk dijadikan suatu informasi yang dapat diolah pada sistem komputer (Firdaus, 2020). Pada penerapannya tujuan utama dari computer vision yaitu untuk dapat melihat dan melakukan analisa pada gambar digital secara otomatis kemudian memberikan informasi dari gambar tersebut. Dengan kata lain komputer akan menafsirkan dan melakukan analisa pada visual dengan simulasi seperti cara kerja manusia dalam melihat dan memahami lingkungan sekitar (Agustiani, 2019). Dalam hal ini menerapkan model machine learning (ML) untuk melakukan identifikasi dan klasifikasi objek pada gambar dan video digital kemudian memungkinkan komputer untuk menganalisa terhadap apa yang dilihat. Jika dijabarkan, computer vision memiliki beberapa jenis tujuan yaitu sebagai berikut.

a. Image segmentation

Segmentasi gambar semantik atau bisa disebut dengan klasifikasi tingkat piksel merupakan tugas mengelompokan tiap bagian pada gambar yang termasuk kedalam kelas objek yang sama. Ada dua tugas dalam image segmentation ini yang pertama yaitu melakukan klasifikasi pada gambar. Klasifikasi berarti mengklasifikasikan pada setiap gambar sebagai kategori identik dan yang kedua yaitu pendeteksian pada gambar yang mengacu pada pengenalan objek dan lokalisasi. Segmentasi gambar dapat dilakukan sebagai prediksi tingkat piksel karena melakukan klasifikasi setiap piksel kedalam kategorinya. Hal ini bertujuan agar dapat dilakukan identifikasi objek dalam menyederhanakan gambar dan melakukan analisa agar lebih efisien (Liu, Deng, & Yang, 2018).

b. Object detection

Object detection atau deteksi objek merupakan bagian dari pengenalan visual dalam *computer vision* dan telah dipelajari secara luas dalam beberapa tahun terakhir. Deteksi objek visual memiliki tujuan untuk menemukan objek dari kelas target tertentu dengan lokalisasi yang tepat dalam gambar yang diberikan dan menetapkan setiap instance objek label pada kelas yang sesuai. Maka, teknik ini dapat memungkinkan sistem untuk melakukan identifikasi serta menemukan objek pada gambar ataupun video. Identifikasi deteksi objek dapat digunakan dalam melakukan perhitungan jumlah objek dalam melihat dan menentukan

keakuratannya dengan memberikan label secara akurat (Wu, Sahoo, & C.H.Hoi, 2020).

c. Facial recognition

Teknologi pengenalan wajah kini digunakan dan diperkenalkan di berbagai aspek kehidupan pada masyarakat. Teknologi yang terintegrasi pengenalan wajah dan deteksi wajah yang sedang berkembang saat ini dapat digunakan untuk mengatasi masalah seperti keamanan, pendaftaran secara otomatis, dan deteksi emosi seseorang (Andrejevic & Selwyn, 2019). Sistem pengenalan wajah ini dapat melakukan identifikasi kepada seseorang dalam bentuk gambar maupun video bahkan bisa juga secara realtime. Facial recognition biasa diterapkan juga pada keamanan biometrik serta bentuk lain dalam perangkat lunak keamanan biometrik seperti pengenalan fingerprint dan retina mata.

d. Image classification

Bisa disebut dengan klasifikasi citra, klasifikasi citra memiliki banyak kegunaan yaitu dapat digunakan untuk melakukan identifikasi area yang berbeda berdasarkan jenis penggunaannya. Klasifikasi citra memegang peranan penting dan digunakan untuk berbagai aplikasi seperti perubahan lingkungan, pertanian, lahan/perencanaan lahan, tata kota, pengawasan, pemetaan geografis, pengendalian bencana hingga pada deteksi objek (Deepan & Sudha, 2020). Citra beresolusi tinggi juga dapat digunakan dalam membantu penanganan bencana alam seperti banjir, gunung berapi, dan kekeringan parah untuk melihat dampak dan

kerusakan. Menyajikan arsitektur komputasi terdistribusi berbasis Hadoop untuk identifikasi penggunaan lahan skala besar dari citra satelit. Untuk menentukan penggunaan lahan, kategori taksonomi semantik seperti vegetasi, bangunan, trotoar, dan lainnya (Sinha, 2016).

e. Edge detection

Deteksi tepi (*edge detection*) memiliki peran penting untuk keandalan dan keamanannya yang memberikan pemahaman tentang pengenalan objek dalam visi komputer. Contohnya seperti deteksi pejalan kaki, deteksi wajah, dan pengawasan video. *Edge detection* secara langsung digabungkan dengan variasi bentuk dalam distribusi intensitas piksel. Deteksi tepi secara terus-menerus dianggap sebagai operasi dasar yang dicapai pada pemrosesan gambar tingkat rendah dan berbagai aplikasi visi komputer. Dua pendekatan utama pada *edge detection* dari sebuah citra yaitu teknik thresholding/enhancement dan teknik fitting tepi. Yang pertama, menggambarkan diskontinuitas pada atribut citra yang ditingkatkan dengan pemrosesan menggunakan operator. Contohnya, melakukan perubahan ketajaman dan kecerahan gambar. Yang kedua yaitu melibatkan pemasangan / piksel tepi yang ideal (Mittal, et al., 2019).

f. Pattern detection

Pengenalan Pola (*pattern detection*) merupakan bidang yang berkembang pesat dan mendukung perkembangan di bidang visi komputer, pemrosesan gambar, analisis teks dan dokumen, serta

jaringan saraf. Pattern detection merupakan teknik dibidang pembelajaran mesin (*machine learning*) yang difokuskan pada proses klasifikasi objek ke dalam kelas – kelas tertentu untuk mengelompokkan data sesuai dengan kelasnya. Pattern recognition digunakan untuk mengetahui atau mengindentifikasi objek dari sebuah media yang pada penelitian ini menggunakan media foto. Foto wajah bisa diambil polanya untuk dikenali dan digunakan sebagai pengenalan identitas (Kurniadi, Sugiyono, & Wardaya, 2021). Selain itu pattern detection juga dapat digunakan pada beberapa media lainnya seperti identifikasi suara, pengenalan suara, pengenalan dalam dokumen multimedia, bisa disebut dengan *multimedia document recognition* (MDR) serta diagnosis medis dengan deteksi otomatis.

g. Feature matching

Pencocokan gambar (*feature matching*) dapat dikenal sebagai korespondensi gambar yang memiliki tujuan untuk melakukan identifikasi kemudian menyesuaikan struktur/konten yang sama/serupa dari dua gambar atau lebih. Teknik ini digunakan untuk pemulihan struktur dimensi tinggi pada gambar serta identifikasi dan integrasi informasi, seperti rekonstruksi 3-D, lokalisasi dan pemetaan simultan visual (VSLAM), mosaik gambar, fusi gambar, pengambilan gambar, pengenalan dan pelacakan target, serta sebagai deteksi perubahan, dan kebutuhan lainnya (Ma, Jiang, Fan, Jiang, & Yan, 2021). Secara garis besarnya *feature matching* yaitu proses melakukan pencocokan gambar

dari dua objek gambar atau lebih. Pencocokan citra secara umum terdiri dari dua bagian, yaitu sifat fitur yang dicocokkan dan strategi pencocokan yang menunjukkan apa yang digunakan untuk mencocokkan dan bagaimana mencocokkannya dari masing-masing objek tersebut.

2.3.2. Deep Learning dan Machine Learning

Penjelasan tentang apa itu deep learning, deep learning memiliki akar yang sama dengan machine learning (bagian dari machine learning). Machine learning secara umum mengacu pada kategori algoritma yang secara otomatis dapat menghasilkan model prediksi dengan mendeteksi pola dalam data. Machine learning telah berhasil diterapkan untuk melakukan tugas-tugas seperti klasifikasi. Apa yang membuat algoritma deep learning berbeda yaitu terletak pada cara mesin mempelajari fitur dari kumpulan data. Pertama, pembelajaran dapat terjadi di mana komputer secara otomatis menemukan pola dan kesamaan dalam data yang tidak berlabel. Dengan metode ini, tidak ada keluaran spesifik yang diharapkan, dan ini sering digunakan sebagai alat eksplorasi untuk mendeteksi fitur dalam data, mengurangi dimensinya atau mengelompokkan data pada kelompok data yang serupa. Kedua, pembelajaran juga dapat dilakukan dengan pelatihan yang diawasi. Kumpulan data berlabel dengan objek target pertama kali diberikan ke komputer sehingga mesin dapat melatih model untuk mengaitkan label dengan contoh data yang diberikan. Mesin kemudian dapat mengenali dan mengidentifikasi objek-objek ini di kumpulan data lain dengan menggunakan prosedur pembelajaran

umum, algoritma deep learning dapat secara otomatis mendeteksi dan mengekstrak fitur dari data. Artinya, pengguna hanya perlu memberi tahu algoritma deep learning apakah dalam satu gambar ada objek hewan/manusia dalam gambar serta dengan memberikan contoh yang cukup dengan data, algoritma tersebut dapat mengetahui sendiri seperti apa bentuk objek tersebut (Christin, Hervet, & Lecomte, 2019).

Deep learning memungkinkan model melakukan komputasi dari beberapa lapisan pemrosesan untuk melakukan pembelajaran dan merepresentasikan data dengan berbagai tingkat abstraksi dengan meniru cara otak dalam memahami informasi. Sehingga dapat menangkap struktur rumit dari data yang berskala besar. Deep learning merupakan kumpulan metode yang mencakup jaringan saraf, model probabilistik hierarkis, dan berbagai algoritma pembelajaran fitur yang tidak diawasi dan diawasi. Saat ini metode deep learning dapat digunakan dalam mengerjakan beberapa tugas, dapat melakukan pembelajaran dari banyaknya data kompleks dari berbagai sumber (misalnya visual, audio hingga sensor) (Voulodimos, Doulamis, Doulamis, & Protopapadakis, 2018).

Pembelajaran mesin (machine learning) merupakan studi ilmiah tentang algoritma dan model statistik yang digunakan sistem komputer untuk melakukan tugas tertentu tanpa diprogram secara eksplisit (Mahesh, 2019). Pembelajaran mesin merupakan cabang ilmu komputer yang secara luas bertujuan untuk memungkinkan komputer "belajar" tanpa diprogram secara langsung. Ini berasal dari gerakan kecerdasan buatan pada tahun 1950-an dan menekankan tujuan dalam pembuatan aplikasi praktis terutama pada bidang prediksi, klasifikasi dan

optimalisasi. Dalam pembelajaran mesin komputer akan "belajar" dengan meningkatkan kinerja mereka pada tugas melalui "pengalaman". Dalam praktiknya, "pengalaman" biasanya berarti menyesuaikan dengan data. Karenanya, tidak ada batasan yang jelas antara pembelajaran mesin dan pendekatan statistik (Bi, Goodman, Kaminsky, & Lessler, 2019).

2.3.3. Pengolahan Citra Digital

Pengolahan citra digital adalah ilmu yang mempelajari hal-hal berkaitan dengan perbaikan kualitas terhadap suatu gambar (meningkatkan kontras, perubahan warna, restorasi citra), transformasi gambar (translasi, rotasi transformasi, skala, geometrik), melakukan pemilihan citra ciri (feature images) yang optimal untuk tujuan analisis, melakukan penyimpanan data yang sebelumnya dilakukan reduksi dan kompresi, transmisi data, dan waktu proses data. Adapun contoh proses dari konsep kerja dari pengolahan citra digital secara sederhana seperti pada gambar berikut.

Gambar 2.1. Proses sederhana pengolahan citra digital

Secara umum pengolahan citra digital dapat diartikan sebagai pemrosesan gambar dua dimensi dengan menggunakan komputer. Citra digital merupakan sebuah array (larik) yang berisikan nilai-nilai real maupun komplek yang dapat direpresentasikan dengan deretan bit tertentu (Munantri, Sofyan, & F, 2019).

2.3.4. Dataset

Dataset merupakan sebuah komponen penting dalam bidang kecerdasan buatan, khususnya untuk digunakan pada tahap pelatihan maupun pengujian (Iglesias, Zseby, Ferreira, & Zimek, 2019). Dataset dapat disebutkan sebagai dokumen atau kumpulan data yang memiliki satu bahkan lebih catatan (*record*). Tiap dari kelompok *record* ini disebut dengan dataset yang memiliki peranan menyimpan informasi. Contohnya seperti program, asuransi, catatan medis, data institusi dan lain sebagainya. Dataset diperlukan sebagai penyimpanan informasi untuk keperluan aplikasi maupun sistem operasi seperti pustaka, pemrograman maupun variabel dan parameter sistem. Dataset dapat disebut juga sebagai himpunan maupun kumpulan data yang biasanya disajikan dalam bentuk tabel. Secara teknis merupakan sekumpulan item yang dapat diakses secara individual untuk pengelolaan tertentu dalam satu kesatuan serta dapat diatur dalam beberapa jenis struktur data. Contohnya dataset pada perusahaan dapat berupa nama karyawan, informasi kontak, gaji, alamat, bahkan sampai pada produk serta angka penjualan dan lain sebagainya. Kesimpulannya ialah, dataset adalah sekumpulan data yang berurutan dan bisa didapatkan dari berbagai informasi pengamatan, studi, pengukuran hingga analisa untuk kegiatan data mining dan membantu para data scientist menganalisa dan mengolah data menjadi suatu informasi (Raharja, 2022).

2.3.5. Convolutional Neural Network

CNN merupakan kombinasi dari jaringan syaraf tiruan dan metode deep learning serta digunakan pada klasifikasi gambar. Metode CNN terinspirasi dari

cara manusia dan mamalia dalam melihat visual. CNN terdiri dari satu atau lebih lapisan konvoluonal, seringnya dengan suatu lapisan sub sampling yang diikuti oleh satu atau lebih lapisan yang terhubung penuh sebagai standar jaringan syaraf (Fonda, 2020). Metode CNN mampu melakukan proses pembelajaran mandiri untuk pengenalan objek, ekstraksi objek dan klasifikasi serta dapat diterapkan pada citra resolusi tinggi yang memiliki model distribusi non parametrik. CNN merupakan operasi konvolusi yang menggabungkan beberapa lapisan pemrosesan, menggunakan beberapa elemen yang beroperasi secara paralel dan terinspirasi oleh sistem saraf biologis. Pada CNN setiap neuron dipresentasikan dalam bentuk 2 dimensi, sehingga metode ini cocok untuk pemrosesan dengan input berupa citra/gambar (Arrofiqoh & Harintaka, 2018).

Gambar 2.2. Contoh deteksi objek pada metode CNN

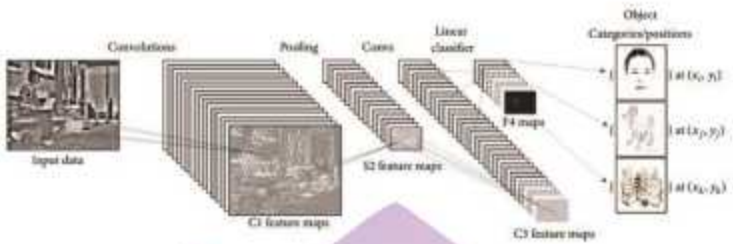
Sampai saat ini CNN sering digunakan dan diterapkan salah satunya pada klasifikasi dan deteksi gambar. *Image classification* merupakan teknik mengkategorikan suatu gambar kedalam suatu kategori tertentu sedangkan *image*

localisation merupakan bagian dari pengembangan dari teknik tersebut. Ketika *output* yang dapat dihasilkan terdapat letak dari objek kelas pada gambar tersebut yang bisa disebut dengan *bounding box*. Object detection merupakan klasifikasi dan lokalisasi pada beberapa objek pada suatu gambar (Dharmadi, CNN: Beyond Image Classification, 2018).

2.3.6. Arsitektur CNN

Konsep kerja CNN mempunnyai kesamaan seperti MLP (*Multi-Layer Perceptron*). Namun setiap neuron pada CNN direpresentasikan dalam bentuk dua dimensi seperti citra dan suara, sedangkan pada MLP hanya dalam bentuk satu dimensi. Karena dalam bentuk dua dimensi CNN dapat melakukan pemrosesan secara linier menggunakan operasi konvolusi dan bobot memiliki bentuk empat dimensi yang merupakan kernel konvolusi (Rahman & Afifana, 2020).

Proses kinerja dari metode CNN yaitu memanfaatkan proses dari lapisan konvolusi CNN terdiri dari tiga jenis utama lapisan saraf yaitu (i) lapisan konvolusional, (ii) lapisan penyatuan, dan (iii) lapisan yang terhubung sepenuhnya. Setiap jenis lapisan memainkan peran yang berbeda. Gambar dibawah ini menunjukkan arsitektur CNN untuk deteksi objek dalam bentuk data citra/gambar. Setiap lapisan CNN mengubah volume masukan menjadi volume keluaran aktivasi neuron, yang pada akhirnya mengarah ke lapisan terakhir yang terhubung sepenuhnya, menghasilkan pemetaan data masukan ke vektor fitur 1D. CNN saat ini sangat familiar dalam pembuatan aplikasi visi komputer, seperti pengenalan wajah, deteksi objek, penglihatan bertenaga dalam robotika, dan mobil self-driving.



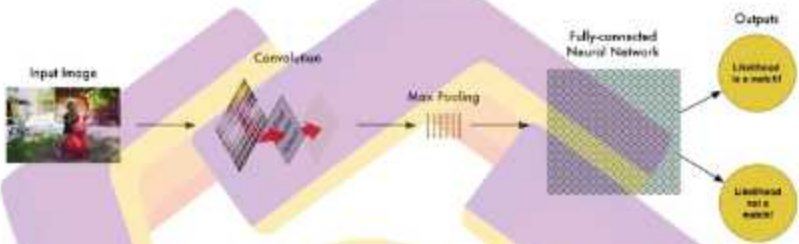
Gambar 2. 3. Contoh arsitektur CNN dari visi komputer (deteksi objek)

(i) Lapisan Konvolusional. Di lapisan convolutional, CNN menggunakan berbagai kernel untuk menggabungkan seluruh gambar serta peta fitur perantara, menghasilkan berbagai peta fitur. Dari proses konvolusi ini dimaksudkan untuk mempersingkat proses waktu pembelajaran.

(ii) Lapisan Penyatuan. Pooling layer bertugas mereduksi dimensi spasial (width & height) dari input untuk convolutional layer berikutnya. Pooling layer tidak mempengaruhi ukuran dimensi pada gambar. Proses yang dilakukan oleh lapisan ini juga disebut subsampling atau downsampling, pengurangan ukuran menyebabkan hilangnya informasi secara bersamaan dan bermanfaat bagi proses jaringan karena penurunan ukuran menyebabkan lebih sedikit beban komputasi untuk lapisan jaringan yang akan datang dan juga bekerja melawan overfitting.

(iii) Lapisan yang Terhubung Sepenuhnya. Mengikuti beberapa lapisan convolutional dan pooling, Neuron dalam lapisan yang terhubung sepenuhnya memiliki koneksi penuh ke semua aktivasi di lapisan sebelumnya. Aktivasi dari proses ini dapat dihitung dengan perkalian matriks diikuti dengan offset bias. Lapisan yang terhubung sepenuhnya akhirnya mengubah peta fitur 2D menjadi

vektor fitur 1D. Vektor yang diturunkan dapat dimasukkan ke dalam sejumlah kategori tertentu untuk klasifikasi atau dapat dianggap sebagai vektor fitur untuk diproses lebih lanjut (Voulodimos, Doulamis, Doulamis, & Protopapadakis, 2018). Sebagai gambaran, tahapan langkah cara kerja dari metode CNN bisa digambarkan sebagai berikut.



Gambar 2.4. Contoh dari cara kerja metode CNN

2.3.7. Tensorflow

Tensorflow merupakan kerangka dari google yang sangat kuat untuk membangun aplikasi menggunakan Machine Learning. Tensorflow adalah sebuah library open source yang dirilis oleh perusahaan Google. Tensorflow dapat dijalankan dengan menggunakan Central Processing Unit (CPU) dan General Processing Unit (GPU) (Scarpino, 2018). Tensorflow dapat digunakan untuk menangai konputasi numerikal menggunakan data-flow graphs. Tensorflow di kembangkan oleh Google Brain Team yang merupakan organisasi peneliti dibawah google yang meneliti tentang machine learning dan deep neural network. Pertama kali mencapai versi pertamanya Versi 1.0 ke publik pada Februari 2017, dan perkembangannya terus meningkat. Tensorflow sangat populer karena dapat menggunakan sekeluruhan sistem pada komputer, Tepatnya Tensorflow dapat

menggunakan komputasi CPU dan GPU. Dan bahkan termasuk pada perangkat mobile seperti smartphone, dan perangkat pendukung IoT (Internet of Things) seperti Raspbery Pi dan Arduino (Khaeriyah, 2019).

BAB III

METODE PENELITIAN

3.1. Jenis, Sifat, dan Pendekatan Penelitian

Pada tahap ini pendekatan penelitian yang dilakukan pada penelitian ini bersifat kuantitatif dengan jenis penelitian eksperimental, pada penelitian ini dimana akan dilakukan skenario pengujian dalam klasifikasi deteksi penggunaan masker yang menggunakan metode algoritma CNN dengan arsitektur VGG16, ResNet50V2, Xception, dan EfficientNetV2S. Selanjutnya, skenario dari percobaan pengujian tersebut akan dievaluasikan hasil untuk dapat menilai keakuratan dalam melakukan klasifikasi dan deteksi citra dari skenario yang dibuat agar dapat mengetahui fakta apa saja yang didapatkan dari hasil penelitian.

3.2. Metode Pengumpulan Data

Metode pengumpulan data dilakukan dengan cara menggunakan dataset yang didapatkan dari media internet. Hasil dari pengumpulan data didapatkan gambar orang-orang yang menggunakan masker dan tidak menggunakan masker. Dari dataset yang telah dikumpulkan dan diperoleh menampilkan banyak objek hingga diperlukannya *croping* pada objek gambar yang tidak diperlukan dan menggunakan objek gambar berupa wajah orang-orang yang menggunakan masker dan tidak menggunakan masker. Selanjutnya, gambar-gambar tersebut dipisah menjadi 2 folder yaitu folder pertama berisikan gambar orang-orang yang menggunakan masker dan folder kedua berisikan gambar orang-orang yang tidak

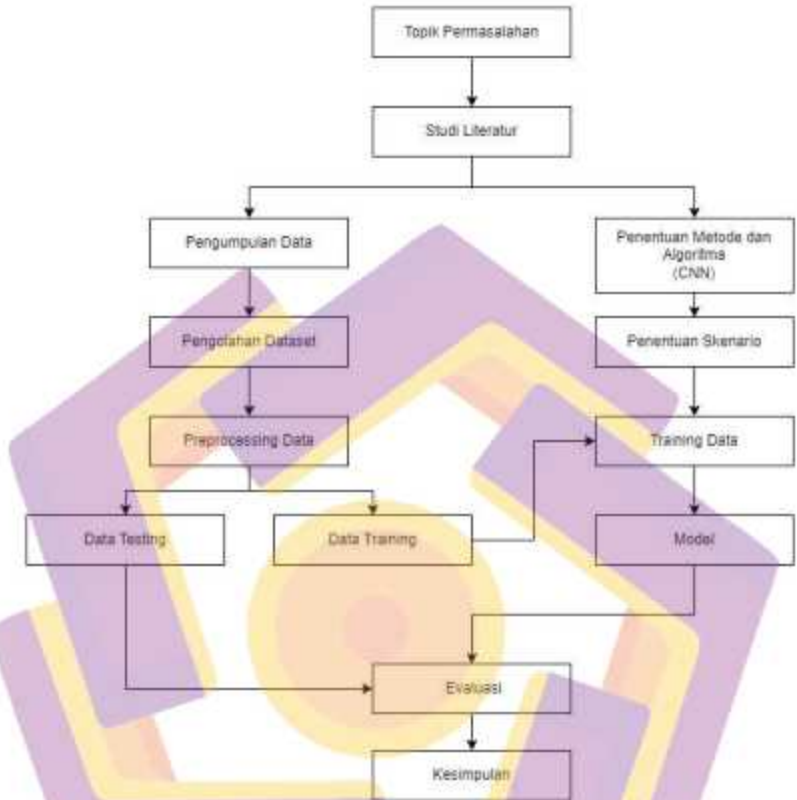
menggunakan masker yang selanjutnya dapat digunakan sebagai dataset. Dataset yang didapatkan tersebut akan diolah hingga berfungi dapat digunakan sebagai pembangunan model pembelajaran mesin.

3.3. Metode Analisis Data

Langkah ini bertujuan untuk memberikan pemahaman dari alur proses dan sistem kerja model yang hendak dibangun menggunakan metode yang sudah ditetapkan. Selain itu juga dapat memberikan pemahaman terhadap langkah-langkah pada proses pembangunan model menggunakan metode algoritma CNN dengan arsitektur VGG16, ResNet50V2, Xception, dan EfficientNetV2S. Dataset yang didapatkan dari pengumpulan data akan dilakukan preprocessing terlebih dahulu seperti melakukan partisi data serta membentuk training image generator sebagai *data augmentation* sebelum tahap ekstraksi fitur dan pembuatan tahap model. Dalam proses percobaan ini menggunakan bahasa pemrograman python serta dengan metode CNN dan model arsitektur VGG16, ResNet50V2, Xception, dan EfficientNetV2S.

3.4. Alur Penelitian

Bagian ini berisi diagram alur langkah-langkah dari penelitian yang akan dilakukan, sebagai berikut.



Gambar 3.1. Alur Penelitian

Alur penelitian dilakukan secara sistematis dan dapat dijelaskan sebagai berikut.

a. Topik Permasalahan

Pada tahap ini topik permasalahan adalah tahap pertama/tahap paling awal dalam memulai penelitian dengan cara melihat permasalahan yang dapat diambil ditengah masyarakat.

b. Studi Literatur

Studi literatur adalah proses mencari informasi terkait hal yang berhubungan dengan permasalahan yang diambil. Studi literatur dapat dilakukan dengan cara membaca jurnal penelitian, paper, buku, artikel dari media internet yang dapat dinilai relevan dengan topik permasalahan yang diambil. Pada tahap ini, proses berikut digunakan sebagai rujukan penelitian agar dapat digunakan untuk pemilihan metode dan model apa yang akan digunakan sebagai solusi dari topik permasalahan yang diambil.

c. Penentuan Metode dan Algoritma

Dari studi literatur yang telah dilakukan sebelumnya, penulis telah menentukan metode dan algoritma yang akan diambil sebagai solusi dalam melakukan penelitian berikut untuk melakukan klasifikasi deteksi penggunaan masker menggunakan metode algoritma CNN dengan arsitektur VGG16, ResNet50V2, Xception, dan EfficientNetV2S.

d. Pengumpulan Dataset

Agar proses penelitian ini dapat berjalan sesuai dengan alur penelitian terutama *machine learning* memerlukan data agar dapat melakukan pembelajaran. Maka, proses pengumpulan dataset diperlukan pada tahap ini. Dataset yang akan digunakan didapatkan yaitu *dataset public* yang didapatkan dari media internet.

e. Pengolahan Dataset

Tahap pengolahan dataset perlu dilakukan dikarenakan dataset yang didapatkan dari media internet dan dataset pribadi pastinya terdiri dari beberapa file yang bervariasi. Maka, data perlu dilakukan pengolahan dengan cara melakukan *croping* objek gambar pada dataset. Selanjutnya dataset tersebut dibagi menjadi 2 folder, folder pertama berisikan gambar orang-orang yang menggunakan masker dan folder kedua berisikan gambar orang-orang yang tidak menggunakan masker.

f. Training Data dan Model

Dataset yang sudah dikumpulkan dan diolah sesuai dengan kebutuhan yang akan digunakan sebagai training data. Tahap selanjutnya yaitu training data yang bertujuan untuk melakukan proses pembuatan model dengan algoritma dan arsitektur yang sudah ditentukan.

g. Evaluasi

Model yang sudah terbentuk dari proses training data selanjutnya dievaluasi dengan menampilkan grafik model hasil dari pelatihan untuk menunjukan kurva tingkat akurasi dari *training accuracy* dan *validation accuracy* yang didapatkan serta menampilkan kurva tingkat error dari *training loss* dan *validation loss*. Kemudian, evaluasi selanjutnya digunakan untuk menampilkan prediksi label menggunakan *confusion matrix* untuk mendapatkan nilai prediksi hasil dari *accuracy, precision, dan recall*.

h. Kesimpulan

Hasil dari penelitian yang sudah dilakukan sesuai alur dari penelitian akan disajikan pada kesimpulan. Hasil dari penelitian akan menghasilkan informasi berupa data/fakta yang dengan menampilkan hasil *output* dari *confusion matrix* terkait dengan algoritma yang ditentukan yaitu algoritma CNN dan arsitektur VGG16, ResNet50V2, Xception, dan EfficientNetV2S.

BAB IV

HASIL PENELITIAN DAN PEMBAHASAN

Pada bab ini dijelaskan tentang implementasi dan analisa dari perancangan pada setiap tahap sesuai dengan alur penelitian. Terdapat beberapa tahap proses dimulai dengan pengumpulan *dataset*, pengolahan *dataset*, menentukan arsitektur, proses pembuatan model, training data dan model hingga pada evaluasi model dari metode CNN.

4.1. Membangun *Dataset*

4.1.1. Pengumpulan *Dataset*

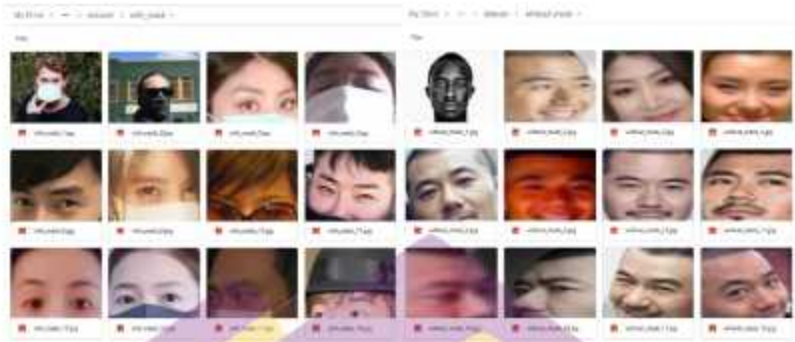
Pada penelitian ini menggunakan *dataset* berupa gambar wajah orang-orang yang menggunakan masker dan tidak menggunakan masker. *Dataset* yang digunakan pada penelitian ini yaitu *dataset* digunakan sebagai data training diperolah dari situs kaggle. Situs kaggle merupakan platform yang menyediakan berbagai jenis *dataset* dan bisa diakses oleh semua pengguna (Suryawan, 2019). *Dataset* yang didapatkan melalui situs kaggle yaitu melalui link sebagai berikut (https://www.kaggle.com/datasets/omkargurav/face-mask-dataset). *Dataset* yang telah didapatkan kemudian dilakukannya proses *croping* pada setiap gambar dengan skala rasio 1:1. Tujuan dari *croping* gambar ini yaitu agar lebih mudah saat diolah pada tahap pembuatan dan pelatihan model CNN. Proses *croping* gambar wajah bermasker dan tidak bermasker dapat dilihat pada gambar 4.1.

Gambar 4.1. *Croping* gambar

Proses *croping* gambar dilakukan pada *dataset* sebagai data training yaitu 2750 data gambar diambil dari kaggle yang dibagi menjadi 1380 gambar wajah bermasker dan 1370 gambar wajah tidak bermasker. Dari kedua jenis gambar wajah bermasker dan tidak bermasker tersebut dibagi menjadi 2 folder sebagai proses pelabelan.

Gambar 4.2. Pelabelan dataset sebagai *data training*

Dataset yang telah dilakukan proses *croping* gambar kemudian disimpan pada *google drive* untuk dilanjutkan proses pengolahan citra menggunakan *google colaboratory*. Berikut sampel gambar *dataset* yang digunakan sebagai *training model* dari masing-masing kelas.



Gambar 4.3. Sampel data training *with_mask* dan *without_mask*

4.1.2. *Preprocessing* Data

Setelah *dataset* yang akan digunakan sebagai *training model* telah dikumpulkan dan dilakukannya proses *cropping*. Tahap berikutnya yaitu melakukan *preprocessing* data yaitu tahap awal untuk melakukan proses pengolahan data yang bertujuan untuk menghindarkan dari data yang menggangu (*noise*) atau data yang tidak konsisten. Bisa juga untuk mempermudah pengolahan data untuk mempermudah proses klasifikasi dalam pembuatan model (Alghifari & Juardi, 2021).

Preprocessing data dimulai dengan diberikannya inisialisasi nilai pada *learning rate*, berapa banyak *epoch* pelatihan dan *batch size* yang pada tahap selanjutnya pemberian nilai tersebut akan digunakan sebagai proses pelatihan model sesuai dengan skenario pengujian. *Learning rate* merupakan parameter *training* yang ditetapkan untuk melakukan perhitungan nilai pada koreksi bobot pada saat waktu proses *training*. Dalam memilih nilai *learning rate* jika nilai yang digunakan terlalu kecil maka proses *training* memerlukan waktu yang lama.

Sebaliknya jika nilai yang digunakan besar proses pembelajaran pada model lebih cepat tetapi menjadi kurang optimal dan proses *training* tidak stabil. *Batch size* merupakan jumlah contoh pelatihan yang digunakan kedalam satu iterasi serta merupakan salah satu *hyperparameter* penting dalam menyesuaikan sistem *machine learning* (Rochmawati, et al., 2021). Dalam menentukan nilai *learning rate* dan *batch size* tentunya akan berdampak pada hasil proses pelatihan model. Selain itu *epoch* merupakan satu siklus dimana algoritma atau model melakukan proses pembelajaran menggunakan *data training* atau istilahnya bisa dikenal sebagai *mini batch*. Dalam penggunaan *mini batch* mesin tidak memproses pembelajaran dari keseluruhan data namun mesin akan belajar dari nilai *batch* yang ditentukan (Wasil, Harianto, & Fathurrahman, 2022).

Pada tahap *preprocessing* berikut juga dilakukan penginputan data gambar yang akan digunakan sebagai *data training* dari *google drive* kedalam *google colaboratory*. Pada tahap *preprocessing* berikut data gambar yang telah diinputkan kedalam *google colaboratory* langsung dilakukannya *resize* yang bertujuan untuk menyamakan ukuran masing-masing gambar pada dataset menjadi 224 x 244 pixel untuk mempermudah proses dalam pembuatan model. Semua *data training* yang diproses kedalam ukuran yang sama kemudian data dan label dikonversikan kedalam variabel kebentuk *array*.

```
# Inisialisasi nilai Initial Learning Rate, berapa banyak Epoch pelatihan, dan Batch Size
INIT_LR = 1e-4
EPOCHS = 25
BS = 30

# Mengambil gambar dari dataset directory, kemudian inisialisasi data dan class gambar
print("Menginput gambar...")
imagePaths = list(paths.list_images('dataset'))
data = []
labels = []

# Melakukan perulangan pada image paths
for imagePath in imagePaths:

    # Mengekstrak class label dari filename
    label = imagePath.split(os.path.sep)[-2]
    # Memuat input gambar (224x224) dan melakukan proses
    image = load_img(imagePath, target_size=(224, 224))
    image = img_to_array(image)
    image = preprocess_input(image)

    # Mengupdate data dan labels lists, berurutan
    data.append(image)
    labels.append(label)

# Mengkonversi data dan label ke dalam NumPy Arrays
data = np.array(data, dtype="float32")
labels = np.array(labels)

# Melakukan one-hot encoding on the labels
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)
print("Input gambar berhasil")
```

Gambar 4.4. *Source code* untuk perintah *preprocessing data*

Hasil akhir dari *preprocessing data* yaitu data gambar sebagai *data training* telah terinput ke dalam *google colaboratory* serta data dan label yang telah terinput tersimpan dalam bentuk *NumPy array* yang bertujuan untuk memudahkan proses komputasi data.

4.1.3. Skenario Percobaan

Agar mendapatkan hasil sesuai dengan yang diharapkan memerlukan beberapa skenario percobaan. Pada skenario percobaan yang akan dibuat

diharapkan dapat digunakan untuk mengukur nilai *accuracy, precision,* dan recall. Selain itu penerapan skenario percobaan dapat digunakan untuk mengukur lama proses waktu pelatihan model. Skenario percobaan dapat dilihat pada tabel berikut.

Tabel 4.1. Skenario percobaan

No	Skenario	Keterangan
1	S1	Arsitektur VGG16
2	S2	Arsitektur ResNet50V2
3	S3	Arsitektur Xception
4	S4	Arsitektur EfficientNetV2S

Ada empat skenario yang ditunjukkan pada tabel 4.1 akan diterapkan pada masing-masing empat skenario yang telah ditulis sebelumnya. Setiap skenario percobaan pada tabel diatas akan ujikan dalam pembelajaran model sebanyak dua kali pelatihan. Pelatihan model pertama menggunakan *epoch* 25 dan *batch size* 30 dan pelatihan kedua menggunakan *epoch 35* dan *batch size* 40. Penentuan nilai *epoch* dan *batch size* merupakan pengembangan dari penelitian (Septhyan, Magdalena, & Pratiwi, 2022) yang telah melakukan penelitian dengan skenario pengujian menentukan dan membandingkan ukuran citra, nilai *epoch,* dan *batch size* untuk mengetahui hasil serta pengaruh dari penentuan masing-masing skenario pengujian dengan nilai tersebut.

4.1.4. Pengelompokan Data

Tahap selanjutnya pengelompokan data yaitu *data testing* dipartisi atau dibagi menjadi dua bagian. *Train set* dan *test set* dengan rasio perbandingan 80% : 20%. Pengambilan perbandingan tersebut diambil berdasarkan dari penelitian

(Wikarta, Suryo, & Effendi, 2020) yang telah melakukan penelitian pengaruh pembagian ukuran *testing* data pada nilai tingkat akurasi. *Train set* yaitu data yang dapat digunakan sebagai data latih untuk melatih model yang akan dibuat. *Test set* yaitu data yang dapat digunakan sebagai evaluasi dan pengujian pada model yang sudah dilatih pada *train set* (Utomo, 2021). Penerapan pengelompokan data diberikan dengan perintah seperti berikut.

```
# Mempartisi data ke dalam pelatihan dan pengujian ( 80% : 20% )
(trainX, testX, trainY, testY) = train_test_split(data, labels,
    test_size=0.20, stratify=labels, random_state=5)
```

Gambar 4.5. *Source code* untuk perintah pengelompokan data

Pemberian perintah *test_size* yaitu digunakan sebagai pemberian nilai berdasar berapa persen rasio yang diinginkan untuk *data test*. Sedangkan pemberian perintah *random_state* yaitu digunakan sebagai pengunci acak pada data. Karena ketika dilakukan pemisahan data saat pelatihan model maka data gambar akan diacak (Asro'i & Februariyanti, 2022).

4.1.5. Pembuatan Image Data Generator dan Data Augmentation

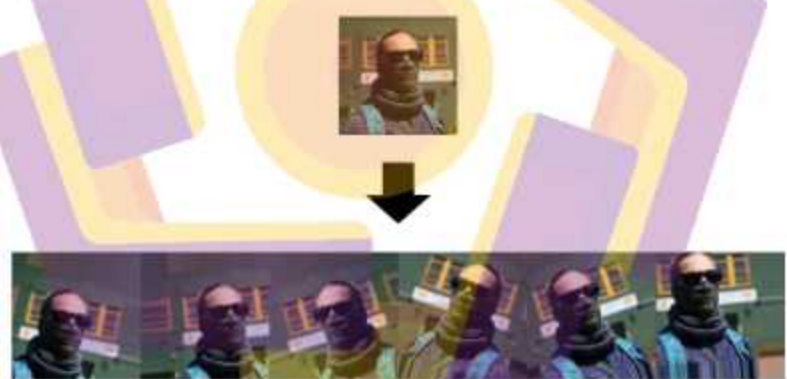
Augmentasi data merupakan proses dalam mengolah data gambar untuk memodifikasi, mengubah serta untuk mendapatkan data tambahan dalam upaya meningkatkan jumlah data saat pelatihan model. Ketika gambar dimodifikasi manusia masih dapat melihat gambar tersebut adalah gambar yang sama namun komputer akan mendeteksi bahwa gambar tersebut berbeda (Mahmud, Adiwijaya, & Faraby, 2019). Hal umum ketika menggunakan augmentasi data yaitu dengan

melakukan modifikasi perputaran gambar dengan nilai zoom tertentu dengan menggunakan *library ImageDataGenerator()* dari *tensorflow*.

```
# Membentuk training image generator untuk data augmentation
aug = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.15,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.15,
    horizontal_flip=True,
    fill_mode="nearest")
```

Gambar 4.6. *Source code* untuk perintah augmentasi data

Berikut merupakan perintah untuk augmentasi data. Setelah mendefinisikan nilai dan perintah dari generatornya maka contoh dari gambar setelah dilakukan augmentasi data seperti berikut.



Gambar 4.7. Contoh hasil dari augmentasi data

4.2. Analisis Data

4.2.1. Convolutional Neural Network

Ketika seluruh kebutuhan *dataset* telah selesai disiapkan maka citra telah siap diproses untuk membuat model pembelajaran menggunakan metode CNN.

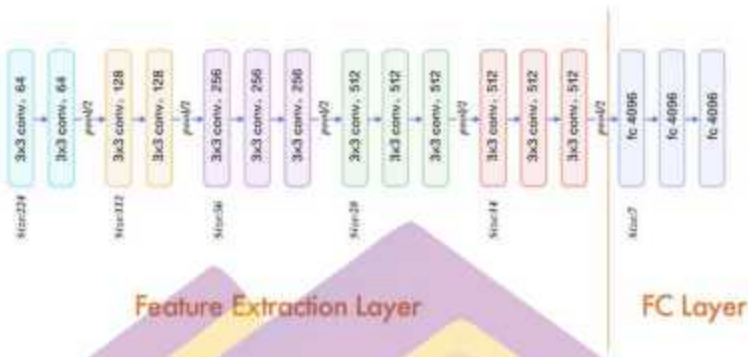
Proses dari klasifikasi model pembelajaran menggunakan arsitektur VGG16, ResNet50V2, Xception, dan EfficientNetV2S agar mendapatkan hasil klasifikasi dengan nilai akurasi terbaik. Berikut merupakan *source code* untuk mengunduh setiap arsitektur.

```
# Arsitektur jaringan VGG16Net
baseModel = tf.keras.applications.VGG16(weights="imagenet", include_top=False,
    input_tensor=Input(shape=(224, 224, 3)))

# Arsitektur jaringan ResNet50V2
baseModel = ResNet50V2(weights="imagenet", include_top=False,
    input_tensor=Input(shape=(224, 224, 3)))

# Arsitektur jaringan Xception
baseModel = Xception(weights="imagenet", include_top=False,
    input_tensor=Input(shape=(224, 224, 3)))

# Arsitektur jaringan EfficientNetV2
baseModel = EfficientNetV2S(weights="imagenet", include_top=False,
    input_tensor=Input(shape=(224, 224, 3)))
```

Gambar 4.8. *Source code* untuk mengunduh arsitektur

4.2.2. Arsitektur VGG16

Arsitektur VGG16 seperti namanya yaitu memiliki jumlah 16 blok layer terdiri dari *feature extraction layer* yang didalamnya terdapat *convolution layer* dan *pooling layer*. Masing-masing bertugas melakukan ekstraksi fitur dan mengecilkan dimensi dari citra. Kemudian lapisan layer terakhir yaitu *fully connected layer* memiliki jumlah 2 *fully connected layer*. Pada pemanggilan arsitektur VGG16 belum memiliki parameter *output*. Maka diperlukan *fine tuning* agar dapat memberikan parameter *output* sesuai dengan jumlah kelasnya. *Fully connected layer* berfungsi sebagai penentuan kelas dari citra berdasarkan dengan tingkat probabilitas pada setiap kelas mana yang paling mendekati.



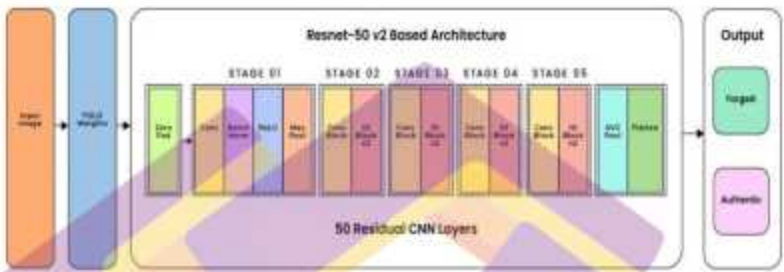
Gambar 4.9. Kerangka Arsitektur VGG16

Pada gambar tersebut merupakan kerangka arsitektur VGG16 yang dikembangkan dari alexnet dimana arsitektur ini menggunakan ukuran dimensi kernel yang sama secara konsisten yaitu 3 x 3. Pada masing – masing susunan konvolusi memiliki perbedaan pada tiap susunan jumlah filter. Jumlah filter 64 terdapat pada susunan konvolusi pertama atau 1 dan 2, selanjutnya jumlah filter 128 tedapat pada susunan 3 dan 4. Selanjutnya pada filter 256 terdapat pada susunan 5, 6, 7. Filter 512 terdapat pada susunan 8, 9, 10, 11, 12, 13. *Max pooling* 2 x 2 terdapat pada susunan setelah konvolusi ke 2, 4, 7, 10 dan 13. Pada *output pooling* yang terdapat pada bagian terakhir dihubungkan ke *fully connected layer* sehingga akan menghubungkan ke tahap klasifikasi untuk menentukan kelas dan labelling (Saputro, Mu'min, Lutfi, & Putri, 2022).

4.2.3. Arsitektur ResNet50V2

Residual Network (ResNet) merupakan jaringan saraf yang telah banyak digunakan pada komputer vision. Arsitektur ResNet50V2 dikembangkan pada

tahun 2016 merupakan versi modifikasi pengembangan dari ResNet50 yang dikembangkan pada tahun 2015. Arsitektur ResNet50V2 diklaim memiliki kinerja lebih baik dari pada ResNet50 dan ResNet101 (Rahimzadeh & Attar, 2020).

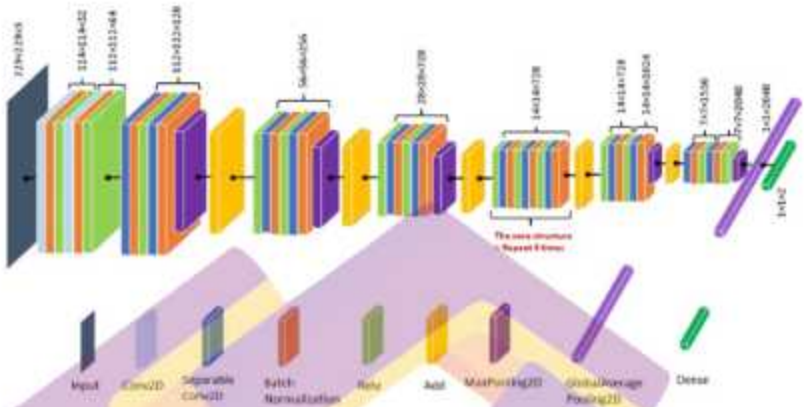


Gambar 4.10. Kerangka Arsitektur ResNet50V2

Pada gambar diatas menunjukkan gambar dari kerangka arsitektur ResNet50V2. Pada arsitektur berikut digunakan sebagai model konvolusi yang memiliki 50 lapisan terdiri dari 5 bagian blok residual. Setiap bagian blok terdiri dari tiga lapisan konvolusi. Selanjutnya pada lapisan akhir terdapat lapisan fungsi aktivasi softmax untuk klasifikasi multi-kelas (Qazi, Zia, & Almorjan, 2022).

4.2.4. Arsitektur Xception

Arsitektur Xception merupakan pengembangan dari arsitektur Inception yang memperluas konsep Inception dengan menggunakan operasi konvolusi yang lebih mendalam dan lebih kompleks.



Gambar 4.11. Kerangka Arsitektur Xception

Pada gambar diatas menunjukkan gambar dari kerangka arsitektur Xception yang memiliki tiga bagian blok utama. Yaitu pada bagian pertama terdapat konvolusi awal yang digunakan untuk mengolah input gambar. Kemudian pada tiga bagian blok utama terdapat blok Entrada, Middle Flow dan Exit Flow (Mehmood, 2021).

4.2.5. Arsitektur EfficientNetV2S

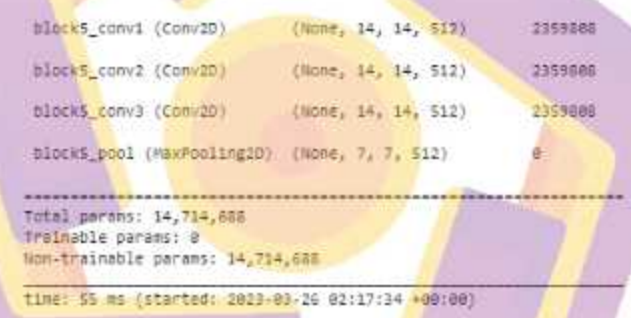
EfficientNet merupakan arsitektur yang dikembangkan pada tahun 2019 oleh google. Sedangkan arsitektur EfficientNet pengembangan V2 merupakan arsitektur yang dikembangkan dari pendahulunya dirilis pada tahun 2021 (Christian, 2021).

Gambar 4.12. Kerangka Arsitektur EfficientNetV2S

Pada gambar berikut merupakan kerangka arsitektur EfficientNetV2S yang memiliki jumlah parameter yang jauh lebih banyak. Pada kerangka arsitektur EfficientNetV2S dimulai dengan lapisan konvolusi awal untuk mengolah input gambar yang memiliki lebih sedikit parameter. Kemudian gambar diolah pada lapisan Fused-MB Conv dan MBconv. Parameter lapisan ini diusulkan dan dibuat lebih banyak karena pada arsitektur berikut difokuskan agar untuk meningkatkan kinerja dari arsitektur jaringan. Arsitektur EfficientNetV2S diklaim dapat menyelesaikan pelatihan model lebih cepat dari pada arsitektur EfficientNet terdahulu (Said, et al., 2022). Terlebih EfficientNetV2S dirancang untuk dapat mengambil manfaat dari transfer learning. Dengan menggunakan bobot yang telah dilatih sebelumnya pada dataset gambar yang besar, EfficientNetV2S dapat menggeneralisasi dan mengekstraksi fitur dari gambar baru dengan efektif.

4.2.6. *Fine Tuning*

Proses ini yaitu melakukan pengambilan bobot dari arsitektur menggunakan *imagenet* selain itu proses ini juga dipergunakan untuk melakukan penyesuaian pada jumlah kelas pada layer konvolusi dengan *data training* yang akan dilatih pada model (Yudianto, 2021). Penerapan *fine tuning* pada penelitian ini yaitu dengan manambahkan *dropout layer* dan *classification layer*. Penambahan tersebut dilakukan dengan membentuk bagian *head* yang akan ditempatkan pada *base model*. Setiap arsitektur yang digunakan pada penelitian ini menggunakan *fine tuning*.



Gambar 4.13. Model VGG16 sebelum diterapkan *fine tuning*

Pada gambar diatas menunjukkan bahwa model dari arsitektur VGG16 sebelum diterapkannya *fine tuning*. Pada arsitektur VGG16 terdiri dari 5 blok *feature extraction layer* yang masing-masing terdiri dari layer konvolusi dan layer pooling. Penerapan *fine tuning* dilakukan menggunakan *source code* seperti berikut.

```
# Membentuk bagian head dari model yang akan ditempatkan pada base model
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)

# menempatkan head model pada base model
model = Model(inputs=baseModel.input, outputs=headModel)
```

Gambar 4.14. *Source code* untuk perintah membentuk bagian *head model*

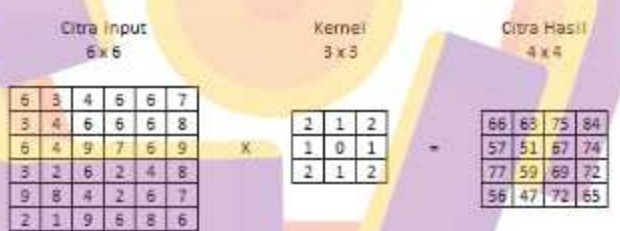
```
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0

average_pooling2d (AverageP  (None, 1, 1, 512)         0
ooling2D)

flatten (Flatten)            (None, 512)               0

dense (Dense)                (None, 128)               65664

dropout (Dropout)            (None, 128)               0

dense_1 (Dense)              (None, 2)                 258

=================================================================
Total params: 14,780,610
Trainable params: 65,922
Non-trainable params: 14,714,688
```

Gambar 4.15. Model VGG16 setelah diterapkan *fine tuning*

Pada gambar 4.15 diatas menunjukkan bahwa model dari arsitektur VGG16 setelah diterapkannya *fine tuning*. Pada gambar tersebut sudah terdapat *average pooling 2d layer*, *flatten layer*, *dense layer* dan *dropout layer*. Terutama pada bagian *dense layer* diterapkan aktivasi *relu* dan *softmax* setelah dilakukannya *fine tuning* memiliki angka 2. Dimana angka tersebut digunakan sesuai dengan jumlah kelas yang akan diproses ke tahap pengujian pada penelitian ini. 2 kelas tersebut terdiri dari kelas bermasker dan tidak bermasker. Penerapan *source code* untuk *fine tuning* tidak hanya pada arsitektur VGG16 saja tetapi juga pada arsitektur lainnya seperti ResNet50V2, Xception, dan EfficientNetV2S menggunakan *source code* yang sama seperti proses *fine tuning* yang dicontohkan pada arsitektur VGG16.

4.2.7. Proses *Convolution*

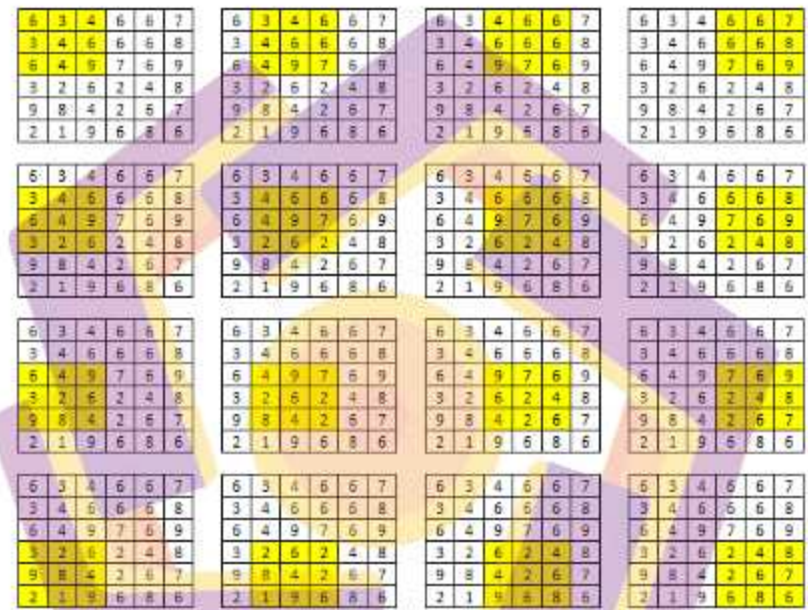
Proses *convolution* atau konvolusi terdapat pada layer konvolusi yang berfungsi dalam melakukan ekstraksi fitur penting terdapat pada citra. Data berupa gambar yang diolah oleh sistem atau mesin yaitu dengan cara mengubah data gambar tersebut menjadi sekumpulan matriks hingga dapat diolah oleh mesin. Proses ini dilakukan menggunakan operasi perhitungan berupa matriks perkalian yang dihitung dari matriks citra dan matriks kernel. Ukuran matriks citra asli yang digunakan pada penelitian ini yaitu memiliki ukuran 224 x 224 piksel. Agar dapat mempermudah dalam penyampaian informasi dan proses dari konvolusi ini lebih mudah untuk dipahami, maka penulis akan menggunakan sebuah citra dengan sampel matriks yang memiliki ukuran 6 x 6 piksel.



Gambar 4.16. Proses Konvolusi

Pada gambar diatas menunjukkan contoh dari operasi perhitungan proses konvolusi menggunakan sampel matriks citra 6 x 6 piksel dan kernel 3 x 3 dengan stride 1. Stride 1 dapat diartikan sebagai gerakan pergeseran 1 matriks ke arah kanan dan seterusnya hingga sampai ujung paling kanan kemudian akan bergeser satu matriks ke arah bawah mulai dari ujung paling kiri dan bergeser 1 matriks terus menerus sampai ke ujung kanan kembali. Proses dari operasi konvolusi tersebut

akan berulang hingga matriks citra selesai dihitung. Angka / nilai yang terdapat pada kolom matriks tersebut direpresentasikan sebagai fitur citra yang dibaca oleh mesin. Pada operasi proses konvolusi nilai tersebut dapat berkisar mulai dari 0 – 255.



Gambar 4.17. Pergerakan dari Proses Konvolusi

Pada gambar diatas operasi dari proses konvolusi menunjukkan pergerakan setiap 1 stride yang berarti pergeseran 1 matriks. Setiap blok matriks 3 x 3 tersebut akan dikalikan dengan matriks kernel 3 x 3. Dalam prakteknya sesuai dengan penelitian yang dilakukan nilai bobot dari kernel akan selalu berubah dalam prosesnya saat dilakukan proses training. Berikut merupakan sampel dari perhitungan proses konvolusi pada pergerakan matriks pertama sampai ke tiga.

```
P1 = (6*2) + (3*1) + (4*2) + (3*1) + (4*0) + (6*1) + (6*2) + (4*1) + (9*2)
   = 66
P2 = (3*2) + (4*1) + (6*2) + (4*1) + (6*0) + (6*1) + (4*2) + (9*1) + (7*2)
   = 63
P3 = (4*2) + (6*1) + (6*2) + (6*1) + (6*0) + (6*1) + (9*2) + (7*1) + (6*2)
   = 75
```

Gambar 4.18. Contoh operasi perhitungan proses konvolusi

Gambar diatas merupakan contoh dari operasi perhitungan perkalian antara matriks citra 6 x 6 dan matriks kernel 3 x 3. Hasil dari operasi proses konvolusi tersebut maka menghasilkan matriks baru dengan ukuran 4 x 4. Ketika proses konvolusi selesai dilakukan maka selanjutnya yaitu melakukan proses *pooling layer*.

4.2.8. Proses *Pooling*

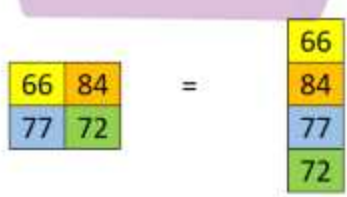
Pada tahap proses *pooling* yaitu untuk mengecilkan dimensi dari citra yang telah diinputkan dari awal. Proses ini terdiri dari 2 jenis yaitu *MaxPooling2D* dan *average_pooling2D*. Pada tahap ini nilai hasil dari citra 4 x 4 dilakukan proses *pooling* dan diperkecil menjadi nilai matriks 2 x 2. Kali ini penulis akan memberikan contoh proses *pooling* menggunakan *max pooling* yaitu dengan memilih nilai tertinggi dalam setiap satu blok dalam matriks yang di bagi menjadi 4 bagian.

Gambar 4.19. Proses *pooling layer*

Pada gambar diatas menunjukkan proses dari *pooling layer* dengan citra hasil proses dari operasi perhitungan konvolusi. Proses *pooling layer* diatas yaitu mengkonversi dari matriks 4 x 4 dengan jumlah stride yaitu 2 menjadi matriks 2 x 2. Pada tahap ini stride 2 yaitu diartikan sebagai pergeseran matriks sebanyak 2 baris. Blok pertama diberikan tanda warna kuning, blok kedua jingga, blok ketiga biru dan blok ke empat hijau. Karena contoh yang diberikan penulis menggunakan teknik *max pooling* yaitu setiap blok diambil jumlah nilai yang terbesar kemudian selanjutnya hasil proses didapatkan matriks ukuran 2 x 2.

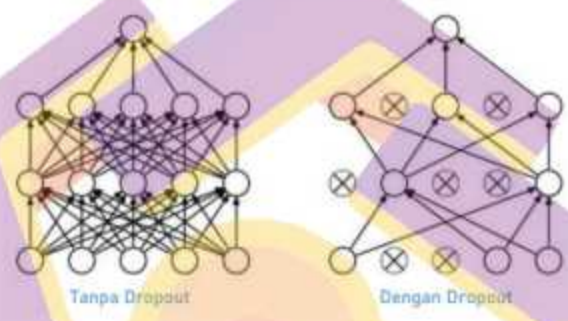
4.2.9. Proses Klasifikasi

Proses klasifikasi yaitu melewati tahap *flatten, dropout,* dan proses *dense*. Citra hasil dari pengolahan sistem/mesin yang berada pada tahap sebelumnya yaitu masih berbentuk matriks multidimensi. Tetapi pada tahap klasifikasi yaitu tidak dapat menerima input berbentuk matriks multidimensi *array*. Maka dari itu diperlukan proses *flatten*. *Flatten* memiliki fungsi melakukan pembentukan ulang pada *feature map* dari matriks multidimensi *array* diolah menjadi vektor. Pada proses *flatten* ini diperlukan sehingga nilai dari input matriks multidimensi *array* dapat digunakan dan diproses pada tahap klasifikasi (Ramba, 2020).



Gambar 4.20. Proses *Flatten*

Setelah melewati tahap proses *flatten*, kemudian selanjutnya proses *dropout* yaitu merupakan teknik untuk melakukan pemilihan secara acak pada beberapa neuron yang dipakai dan tidak dipakai dalam proses *training*. Tujuan dalam penggunaan teknik ini yaitu untuk mempercepat waktu pemrosesan pembelajaran menggunakan metode CNN (Ramba, 2020).



Gambar 4.21. Perbandingan proses tanpa *dropout* dan dengan *dropout*

Pada gambar diatas dijelaskan gambaran dari perbandingan proses pembelajaran menggunakan *neural network* tidak menggunakan *dropout layer* dan menggunakan *dropout layer*. Ketika melewati proses ini maka hasil *output* dari klasifikasi tersebut dapat diolah ke *dense layer* yang memiliki fungsi aktivasi *softmax* untuk menentukan kelas dari citra yang telah diolah sesuai dengan nilai probabilitas yang paling mendekati dengan kelas citra.

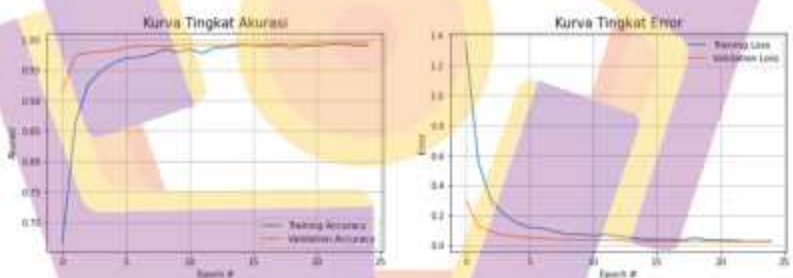
4.3. Hasil *Training Model* dan *Validation Data*

Dataset, skenario percobaan, dan arsitektur telah siap digunakan tahap lanjutannya yaitu pelatihan model yang dilakukan menggunakan *google colaboratory*. Pelatihan model ini diharapkan agar mendapatkan hasil yang sesuai

dengan skenario percobaan yang telah direncanakan pada tahap *preprocessing*. Setiap skenario percobaan model akan dilatih sebanyak 2 kali pelatihan setiap skenario. Model yang telah dilatih kemudian dievaluasikan menggunakan grafik dan *confusion matrix*.

4.3.1. Skenario Pertama

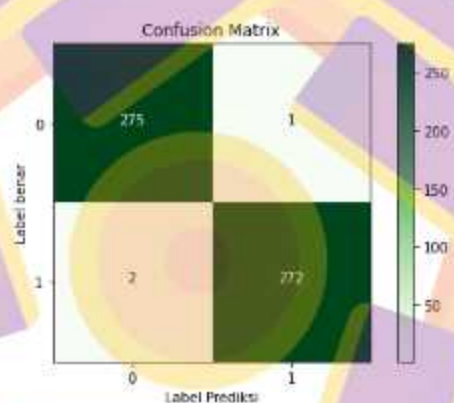
Pada pengujian model berikut skenario pertama yang diterapkan yaitu menggunakan arsitektur VGG16. Model akan dilatih sebanyak dua kali pada pelatihan pertama menggunakan *max epoch* 25 dan *batch size* 30 sedangkan pelatihan kedua menggunakan *max epoch* 35 dan *batch size* 40. Hasil dari proses pelatihan model pertama dapat ditampilkan kedalam grafik seperti berikut.



Gambar 4.22. Grafik tingkat akurasi dan grafik tingkat error pelatihan model 1

Pada gambar 4.22 menampilkan gambar bagian grafik tingkat akurasi dimana *training* dan *validation accuracy* dari *epoch* pertama hingga 25 mendapatkan nilai *training accuracy* 0,6682 mengalami peningkatan mencapai nilai 0,9908. Sedangkan untuk *validation accuracy* mulai dari 0,9145 mengalami peningkatan mencapai nilai 0,9945. Pada gambar 4.22 bagian grafik tingkat error menampilkan grafik nilai *loss* dari *epoch* pertama hingga 25 mendapatkan nilai

training loss 1,3496 mengalami penurunan mencapai nilai 0,0290. Sedangkan untuk *validation loss* mulai dari 0,3026 mengalami penurunan mencapai nilai 0,0253. Waktu proses pelatihan model yang diperlukan 15 menit 1 detik atau total 901 detik. Model yang sudah terbentuk kemudian diuji dan dievaluasikan menggunakan *confusion matrix*. *Confusion matrix* berfungsi untuk mengukur kinerja dari suatu model yang sudah terbentuk dari klasifikasi yang dilakukan oleh sistem (Karsito & Susanti, 2019).



Gambar 4.23. Evaluasi *confusion matrix* pelatihan model 1

Evaluasi yang dilakukan dengan menggunakan 550 data uji dibagi dalam kelas *with_mask* 276 data dan *without_mask* 274 data. Dimana diketahui 275 data dalam kelas *with_mask* diprediksi dengan benar, 1 data diprediksi salah. Selain itu diketahui 272 data dalam kelas *without_mask* diprediksi dengan benar, 2 data diprediksi salah. Dengan melalui proses prediksi berikut maka dapat dihitung jumlah nilai dari *accuracy*, *precision*, dan *recall*.

$$\text{Accuracy} = \frac{275 + 272}{550} = 0{,}9945 \quad (99{,}45\%)$$

$$\text{Precision } (with\_mask) = \frac{275}{275 + 1} = 0{,}9964 \quad (99{,}64\%)$$

$$\text{Precision } (without\_mask) = \frac{272}{272 + 2} = 0{,}9927 \quad (99{,}27\%)$$

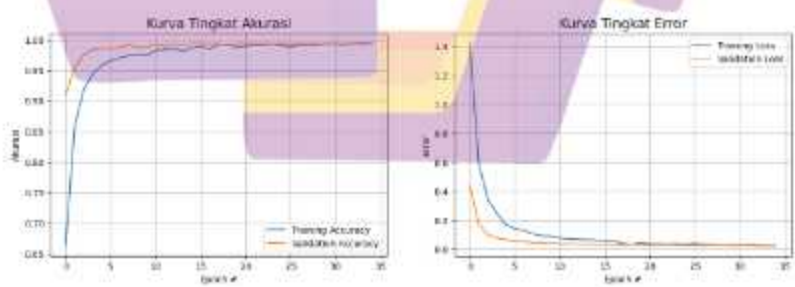
$$\textbf{Precision} = \frac{\textbf{0,9964 + 0,9927}}{\textbf{2}} = \textbf{0,9946} \quad \textbf{(99,46\%)}$$

$$\text{Recall } (with\_mask) = \frac{275}{275 + 2} = 0{,}9928 \quad (99{,}28\%)$$

$$\text{Recall } (without\_mask) = \frac{272}{272 + 1} = 0{,}9963 \quad (99{,}63\%)$$

$$\textbf{Recall} = \frac{\textbf{0,9928 + 0,9963}}{\textbf{2}} = \textbf{0,9946} \quad \textbf{(99,46\%)}$$

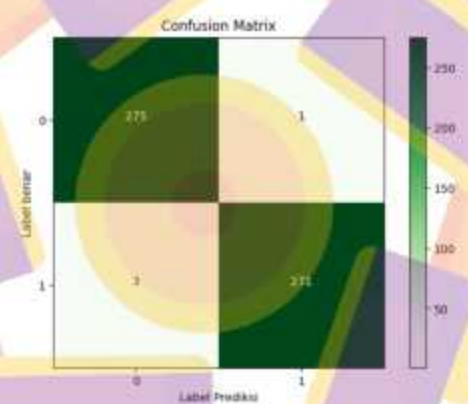
Dari perhitungan diatas, pengujian model yang telah terbentuk mendapatkan hasil nilai *accuracy* 0,9945 (99,45%), nilai *precision* 0,9946 (99,46%), dan nilai *recall* 0,9946 (99,46%). Pada skenario 1 pelatihan model kedua menggunakan *max epoch* 35 dan *batch size* 40. Hasil dari proses pelatihan model kedua dapat ditampilkan kedalam grafik seperti berikut.



Gambar 4. 24. Grafik tingkat akurasi dan grafik tingkat error pelatihan model 2

Pada gambar 4.24 menampilkan gambar bagian grafik tingkat akurasi dimana *training* dan *validation accuracy* dari *epoch* pertama hingga 35

mendapatkan nilai *training accuracy* 0,6623 mengalami peningkatan mencapai nilai 0,9932. Sedangkan untuk *validation accuracy* mulai dari 0,9091 mengalami peningkatan mencapai nilai 0,9927. Pada gambar 4.24 bagian grafik tingkat error menampilkan grafik nilai *loss* dari *epoch* pertama hingga 35 mendapatkan nilai *training loss* 1,4208 mengalami penurunan mencapai nilai 0,0238. Sedangkan untuk *validation loss* mulai dari 0,4404 mengalami penurunan mencapai nilai 0,0233. Waktu proses pelatihan model yang diperlukan 19 menit 46 detik atau total 1186 detik.



Gambar 4.25. Evaluasi *confusion matrix* pelatihan model 2

Evaluasi yang dilakukan dengan menggunakan 550 data uji dibagi dalam kelas *with_mask* 275 data dan *without_mask* 271 data. Dimana diketahui 275 data dalam kelas *with_mask* diprediksi dengan benar, 1 data diprediksi salah. Selain itu diketahui 271 data dalam kelas *without_mask* diprediksi dengan benar, 3 data diprediksi salah. Dengan melalui proses prediksi berikut maka dapat dihitung jumlah nilai dari *accuracy*, *precision*, dan *recall*.

$$\text{Accuracy} = \frac{275 + 271}{} = 0{,}9927 \quad (99{,}27\%)$$

550

$$\text{Precision } (with\_mask) = \frac{275}{275 + 1} = 0{,}9964 \quad (99{,}64\%)$$

$$\text{Precision } (without\_mask) = \frac{271}{271 + 3} = 0{,}9891 \quad (98{,}91\%)$$

$$\textbf{Precision} = \frac{0{,}9964 + 0{,}9891}{2} = \textbf{0,9928} \quad \textbf{(99,28\%)}$$

$$\text{Recall } (with\_mask) = \frac{275}{275 + 3} = 0{,}9892 \quad (98{,}92\%)$$

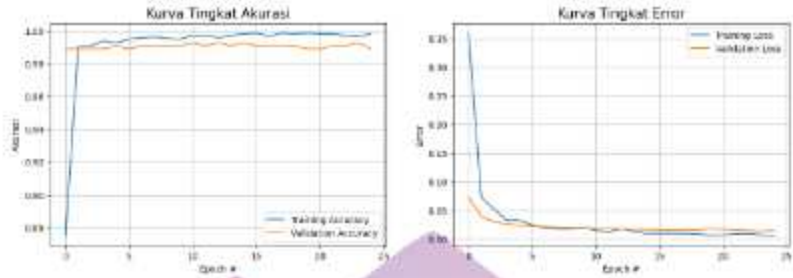
$$\text{Recall } (without\_mask) = \frac{271}{271 + 1} = 0{,}9963 \quad (99{,}63\%)$$

$$\textbf{Recall} = \frac{0{,}9892 + 0{,}9963}{2} = \textbf{0,9928} \quad \textbf{(99,28\%)}$$

Dari perhitungan diatas, pengujian model yang telah terbentuk mendapatkan hasil nilai *accuracy* 0,9927 (99,27%), nilai *precision* 0,9928 (99,28%), dan nilai *recall* 0,9928 (99,28%).

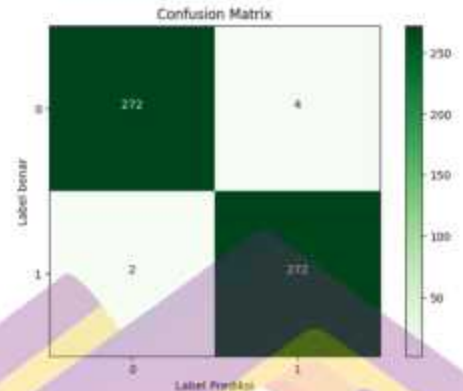
4.3.2. Skenario Kedua

Pada pengujian model berikut skenario kedua yang diterapkan yaitu menggunakan arsitektur ResNet50V2. Model akan dilatih sebanyak dua kali pada pelatihan pertama menggunakan *max epoch* 25 dan *batch size* 30 sedangkan pelatihan kedua menggunakan *max epoch* 35 dan *batch size* 40. Hasil dari proses pelatihan model pertama dapat ditampilkan kedalam grafik seperti berikut.



Gambar 4.26. Grafik tingkat akurasi dan grafik tingkat error pelatihan model 1

Pada gambar 4.26 menampilkan gambar bagian grafik tingkat akurasi dimana *training* dan *validation accuracy* dari *epoch* pertama hingga 25 mendapatkan nilai *training accuracy* 0,8751 mengalami peningkatan mencapai nilai 0,9982. Sedangkan untuk *validation accuracy* mulai dari 0,9891 mengalami perubahan nilai yang tidak terlalu signifikan pada setiap proses pembelajarannya hingga pada akhir pelatihan mendapatkan hasil nilai 0,9891. Pada gambar 4.26 bagian grafik tingkat error menampilkan grafik nilai *loss* dari *epoch* pertama hingga 25 mendapatkan nilai *training loss* 0,3595 mengalami penurunan mencapai nilai 0,0054. Sedangkan untuk *validation loss* mulai dari 0,0735 mengalami penurunan mencapai nilai 0,0147. Waktu proses pelatihan model yang diperlukan 15 menit 29 detik atau 929 detik.



Gambar 4.27. Evaluasi *confusion matrix* pelatihan model 1

Evaluasi yang dilakukan dengan menggunakan 550 data uji dibagi dalam kelas *with_mask* 272 data dan *without_mask* 272 data. Dimana diketahui 272 data dalam kelas *with_mask* diprediksi dengan benar, 4 data diprediksi salah. Selain itu diketahui 272 data dalam kelas *without_mask* diprediksi dengan benar, 2 data diprediksi salah. Dengan melalui proses prediksi berikut maka dapat dihitung jumlah nilai dari *accuracy*, *precision*, dan *recall*.

$$\textbf{Accuracy} = \frac{\mathbf{272 + 272}}{\mathbf{550}} = \mathbf{0{,}9891} \quad \mathbf{(98{,}91\%)}$$

$$\text{Precision } (\textit{with\_mask}) = \frac{272}{272 + 4} = 0{,}9855 \quad (98{,}55\%)$$

$$\text{Precision } (\textit{without\_mask}) = \frac{272}{272 + 2} = 0{,}9927 \quad (99{,}27\%)$$

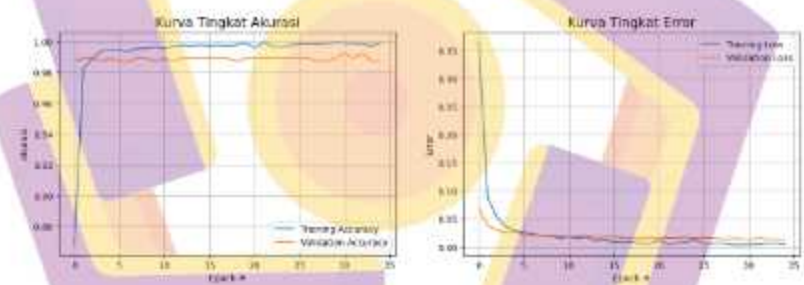
$$\textbf{Precision} = \frac{\mathbf{0{,}9855 + 0{,}9927}}{\mathbf{2}} = \mathbf{0{,}9891} \quad \mathbf{(98{,}91\%)}$$

$$\text{Recall } (\textit{with\_mask}) = \frac{275}{275 + 2} = 0{,}9928 \quad (99{,}28\%)$$

$$\text{Recall } (\textit{without\_mask}) = \frac{272}{272 + 4} = 0{,}9855 \quad (98{,}55\%)$$

$$\textbf{Recall} = \frac{\textbf{0,9928 + 0,9855}}{\textbf{2}} = \textbf{0,9892} \quad \textbf{(98,92\%)}$$

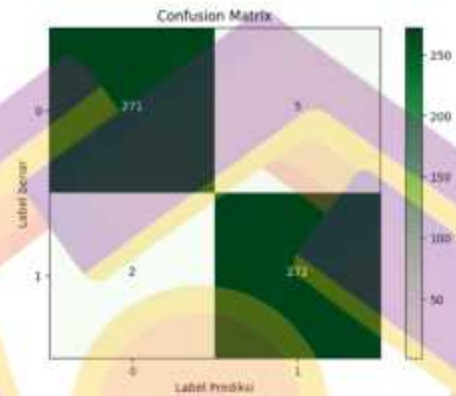
Dari perhitungan diatas, pengujian model yang telah terbentuk mendapatkan hasil nilai *accuracy* 0,9891 (98,91%), nilai *precision* 0,9891 (98,91%), dan nilai *recall* 0,9892 (98,92%). Pada skenario 2 pelatihan model kedua menggunakan *max epoch* 35 dan *batch size* 40. Hasil dari proses pelatihan model kedua dapat ditampilkan kedalam grafik seperti berikut.



Gambar 4.28. Grafik tingkat akurasi dan grafik tingkat error pelatihan model 2

Pada gambar 4.28 menampilkan gambar bagian grafik tingkat akurasi dimana *training* dan *validation accuracy* dari *epoch* pertama hingga 35 mendapatkan nilai *training accuracy* 0,8682 mengalami peningkatan mencapai nilai 0,9991. Sedangkan untuk *validation accuracy* mulai dari 0,9873 mengalami perubahan nilai yang tidak terlalu signifikan pada setiap proses pembelajarannya hingga pada akhir pelatihan mendapatkan hasil nilai 0,9873. Pada gambar 4.28 bagian grafik tingkat error menampilkan grafik nilai *loss* dari *epoch* pertama hingga

35 mendapatkan nilai *training loss* 0,3645 mengalami penurunan mencapai nilai 0,0045. Sedangkan untuk *validation loss* mulai dari 0,0707 mengalami penurunan mencapai nilai 0,0144. Waktu proses pelatihan model yang diperlukan 17 menit 17 detik atau 1037 detik.



Gambar 4.29. Evaluasi *confusion matrix* pelatihan model 2

Evaluasi yang dilakukan dengan menggunakan 550 data uji dibagi dalam kelas *with_mask* 271 data dan *without_mask* 272 data. Dimana diketahui 271 data dalam kelas *with_mask* diprediksi dengan benar, 5 data diprediksi salah. Selain itu diketahui 272 data dalam kelas *without_mask* diprediksi dengan benar, 2 data diprediksi salah. Dengan melalui proses prediksi berikut maka dapat dihitung jumlah nilai dari *accuracy*, *precision*, dan *recall*.

$$\textbf{Accuracy} = \frac{\mathbf{271 + 272}}{\mathbf{550}} = \mathbf{0{,}9873} \quad \mathbf{(98{,}73\%)}$$

$$\text{Precision } (\textit{with\_mask}) = \frac{271}{271 + 5} = 0{,}9819 \quad (98{,}19\%)$$

$$\text{Precision } (\textit{without\_mask}) = \frac{272}{} = 0{,}9927 \quad (99{,}27\%)$$

$$272 + 2$$

$$\textbf{Precision} = \frac{\textbf{0,9819 + 0,9927}}{\textbf{2}} = \textbf{0,9873} \quad \textbf{(98,73\%)}$$

$$\text{Recall } (\textit{with\_mask}) = \frac{271}{271 + 2} = 0{,}9927 \quad (99{,}27\%)$$

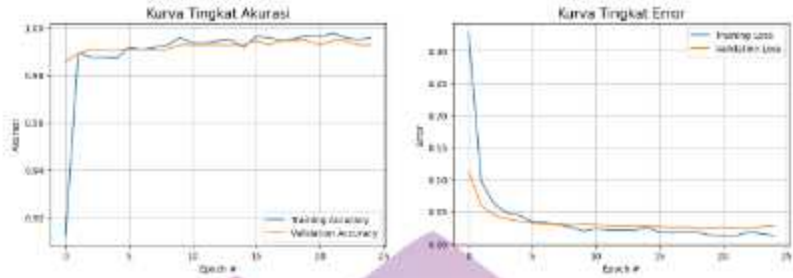
$$\text{Recall } (\textit{without\_mask}) = \frac{272}{272 + 5} = 0{,}9819 \quad (98{,}19\%)$$

$$\textbf{Recall} = \frac{\textbf{0,9927 + 0,9819}}{\textbf{2}} = \textbf{0,9873} \quad \textbf{(98,73\%)}$$

Dari perhitungan diatas, pengujian model yang telah terbentuk mendapatkan hasil nilai *accuracy* 0,9873 (98,73%), nilai *precision* 0,9873 (98,73%), dan nilai *recall* 0,9873 (98,73%).

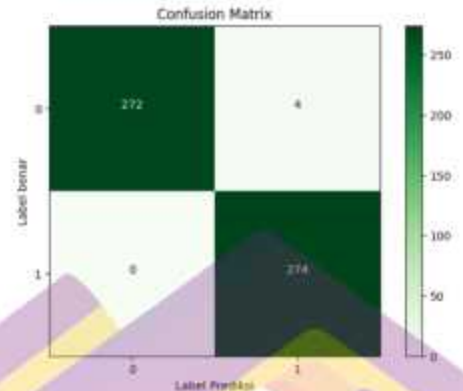
4.3.3. Skenario Ketiga

Pada pengujian model berikut skenario ketiga yang diterapkan yaitu menggunakan arsitektur Xception. Model akan dilatih sebanyak dua kali pada pelatihan pertama menggunakan *max epoch* 25 dan *batch size* 30 sedangkan pelatihan kedua menggunakan *max epoch* 35 dan *batch size* 40. Hasil dari proses pelatihan model pertama dapat ditampilkan kedalam grafik seperti berikut.



Gambar 4.30. Grafik tingkat akurasi dan grafik tingkat error pelatihan model 1

Pada gambar 4.30 menampilkan gambar bagian grafik tingkat akurasi dimana *training* dan *validation accuracy* dari *epoch* pertama hingga 25 mendapatkan nilai *training accuracy* 0,9124 mengalami peningkatan mencapai nilai 0,9959. Sedangkan untuk *validation accuracy* mulai dari 0,9855 mengalami peningkatan hingga nilai 0,9927. Pada gambar 4.30 bagian grafik tingkat error menampilkan grafik nilai *loss* dari *epoch* pertama hingga 25 mendapatkan nilai *training loss* 0,3275 mengalami penurunan mencapai nilai 0,0130. Sedangkan untuk *validation loss* mulai dari 0,1131 mengalami penurunan mencapai nilai 0,0276. Waktu proses pelatihan model yang diperlukan 14 menit 27 detik atau 867 detik.



Gambar 4.31. Evaluasi *confusion matrix* pelatihan model 1

Evaluasi yang dilakukan dengan menggunakan 550 data uji dibagi dalam kelas *with_mask* 272 data dan *without_mask* 274 data. Dimana diketahui 272 data dalam kelas *with_mask* diprediksi dengan benar, 4 data diprediksi salah. Selain itu diketahui 274 data dalam kelas *without_mask* diprediksi dengan benar, 0 data diprediksi salah. Dengan melalui proses prediksi berikut maka dapat dihitung jumlah nilai dari *accuracy*, *precision*, dan *recall*.

$$\textbf{Accuracy} = \frac{\mathbf{272 + 274}}{\mathbf{550}} = \mathbf{0{,}9927}\ \ \mathbf{(99{,}27\%)}$$

$$\text{Precision } (\textit{with\_mask}) = \frac{272}{272 + 4} = 0{,}9855\ \ (98{,}55\%)$$

$$\text{Precision } (\textit{without\_mask}) = \frac{274}{274 + 0} = 1{,}0000\ \ (100{,}00\%)$$

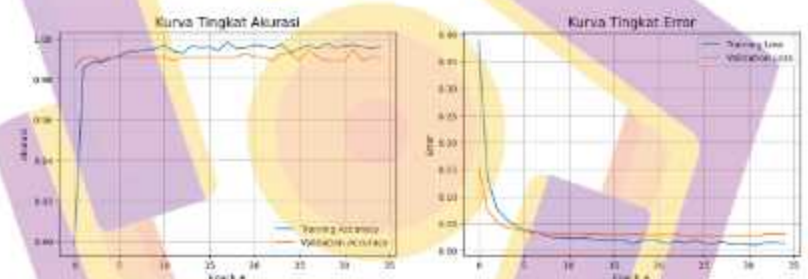
$$\textbf{Precision} = \frac{\mathbf{0{,}9855 + 1{,}0000}}{\mathbf{2}} = \mathbf{0{,}9928}\ \ \mathbf{(99{,}28\%)}$$

$$\text{Recall } (\textit{with\_mask}) = \frac{272}{272 + 0} = 1{,}0000\ \ (100{,}00\%)$$

$$\text{Recall } (\textit{without\_mask}) = \frac{274}{274 + 4} = 0{,}9856 \quad (98{,}56\%)$$

$$\textbf{Recall} = \frac{\textbf{0,9928 + 0,9855}}{\textbf{2}} = \textbf{0,9928} \quad \textbf{(99,28\%)}$$

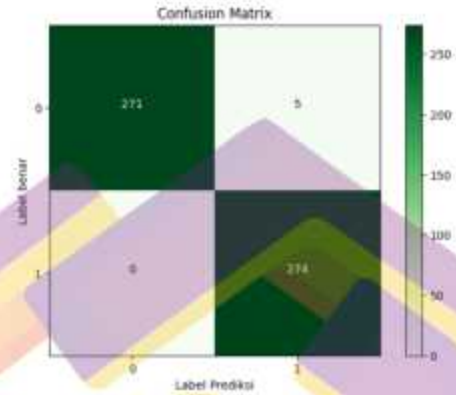
Dari perhitungan diatas, pengujian model yang telah terbentuk mendapatkan hasil nilai *accuracy* 0,9927 (99,27%), nilai *precision* 0,9928 (99,28%), dan nilai *recall* 0,9928 (99,28%). Pada skenario 2 pelatihan model kedua menggunakan *max epoch* 35 dan *batch size* 40. Hasil dari proses pelatihan model kedua dapat ditampilkan kedalam grafik seperti berikut.



Gambar 4.32. Grafik tingkat akurasi dan grafik tingkat error pelatihan model 2

Pada gambar 4.32 menampilkan gambar bagian grafik tingkat akurasi dimana *training* dan *validation accuracy* dari *epoch* pertama hingga 35 mendapatkan nilai *training accuracy* 0,8982 mengalami peningkatan mencapai nilai 0,9964. Sedangkan untuk *validation accuracy* mulai dari 0,9855 mengalami peningkatan hingga nilai 0,9909. Pada gambar 4.32 bagian grafik tingkat error menampilkan grafik nilai *loss* dari *epoch* pertama hingga 35 mendapatkan nilai *training loss* 0,3853 mengalami penurunan mencapai nilai 0,0132. Sedangkan untuk *validation loss* mulai dari 0,1553 mengalami penurunan mencapai nilai

0,0300. Waktu proses pelatihan model yang diperlukan 18 menit 7 detik atau 1097 detik.



Gambar 4.33. Evaluasi *confusion matrix* pelatihan model 2

Evaluasi yang dilakukan dengan menggunakan 550 data uji dibagi dalam kelas *with_mask* 271 data dan *without_mask* 274 data. Dimana diketahui 271 data dalam kelas *with_mask* diprediksi dengan benar, 5 data diprediksi salah. Selain itu diketahui 274 data dalam kelas *without_mask* diprediksi dengan benar, 0 data diprediksi salah. Dengan melalui proses prediksi berikut maka dapat dihitung jumlah nilai dari *accuracy*, *precision*, dan *recall*.

$$\textbf{Accuracy} = \frac{\mathbf{271 + 274}}{\mathbf{550}} = \mathbf{0{,}9909} \quad \mathbf{99{,}09\%}$$

$$\text{Precision } (\textit{with\_mask}) = \frac{271}{271 + 5} = 0{,}9819 \quad 98{,}19\%$$

$$\text{Precision } (\textit{without\_mask}) = \frac{274}{274 + 0} = 1{,}0000 \quad 100{,}00\%$$

$$\textbf{Precision} = \frac{\mathbf{0{,}9819 + 1{,}0000}}{\mathbf{2}} = \mathbf{0{,}9910} \quad \mathbf{99{,}10\%}$$

$$\text{Recall } (with\_mask) = \frac{271}{271 + 0} = 1{,}0000 \quad 100{,}00\%$$

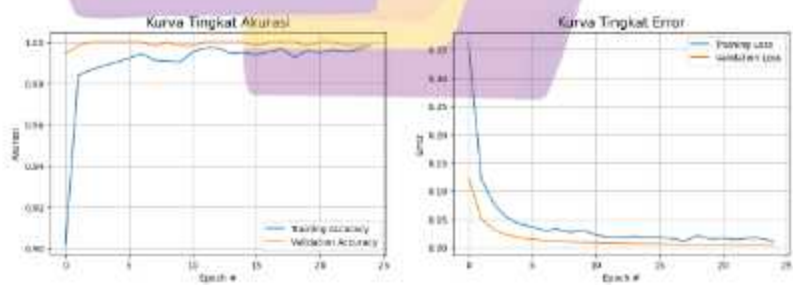
$$\text{Recall } (without\_mask) = \frac{274}{274 + 5} = 0{,}9821 \quad 98{,}21\%$$

$$\textbf{Recall} = \frac{\textbf{1,0000 + 0,9821}}{\textbf{2}} = \textbf{0,9911} \quad \textbf{99,11\%}$$

Dari perhitungan diatas, pengujian model yang telah terbentuk mendapatkan hasil nilai *accuracy* 0,9909 (99,09%), nilai *precision* 0,9910 (99,10%), dan nilai *recall* 0,9911 (99,11%).

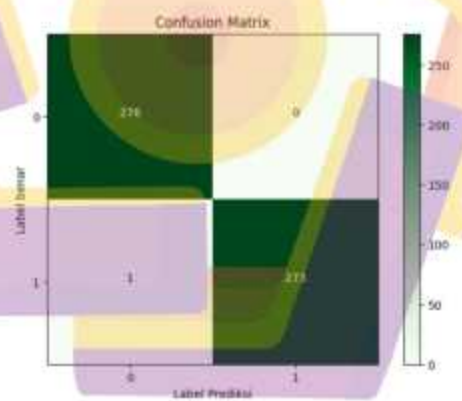
4.3.4. Skenario Keempat

Pada pengujian model berikut skenario ketiga yang diterapkan yaitu menggunakan arsitektur EfficientNetV2S. Model akan dilatih sebanyak dua kali pada pelatihan pertama menggunakan *max epoch* 25 dan *batch size* 30 sedangkan pelatihan kedua menggunakan *max epoch* 35 dan *batch size* 40. Hasil dari proses pelatihan model pertama dapat ditampilkan kedalam grafik seperti berikut.



Gambar 4.34. Grafik tingkat akurasi dan grafik tingkat error pelatihan model 1

Pada gambar 4.34 menampilkan gambar bagian grafik tingkat akurasi dimana *training* dan *validation accuracy* dari *epoch* pertama hingga 25 mendapatkan nilai *training accuracy* 0,9018 mengalami peningkatan mencapai nilai 0,9986. Sedangkan untuk *validation accuracy* mulai dari 0,9945 mengalami perubahan nilai naik turun pada setiap proses pembelajarannya hingga pada akhir pelatihan mendapatkan hasil nilai 0,9982. Pada gambar 4.34 bagian grafik tingkat error menampilkan grafik nilai *loss* dari *epoch* pertama hingga 25 mendapatkan nilai *training loss* 0,3631 mengalami penurunan mencapai nilai 0,0100. Sedangkan untuk *validation loss* mulai dari 0,1235 mengalami penurunan mencapai nilai 0,0049. Waktu proses pelatihan model yang diperlukan 16 menit 15 detik atau 975 detik.



Gambar 4.35. Evaluasi *confusion matrix* pelatihan model 1

Evaluasi yang dilakukan dengan menggunakan 550 data uji dibagi dalam kelas *with_mask* 276 data dan *without_mask* 273 data. Dimana diketahui 276 data dalam kelas *with_mask* diprediksi dengan benar, 0 data diprediksi salah. Selain itu diketahui 273 data dalam kelas *without_mask* diprediksi dengan benar, 1 data

diprediksi salah. Dengan melalui proses prediksi berikut maka dapat dihitung jumlah nilai dari *accuracy*, *precision*, dan *recall*.

$$\textbf{Accuracy} = \frac{\mathbf{276 + 273}}{\mathbf{550}} = \mathbf{0{,}9982} \quad \mathbf{(99{,}82\%)}$$

$$\text{Precision } (\textit{with\_mask}) = \frac{276}{276 + 0} = 1{,}0000 \quad (100{,}00\%)$$

$$\text{Precision } (\textit{without\_mask}) = \frac{273}{273 + 1} = 0{,}9964 \quad (99{,}64\%)$$

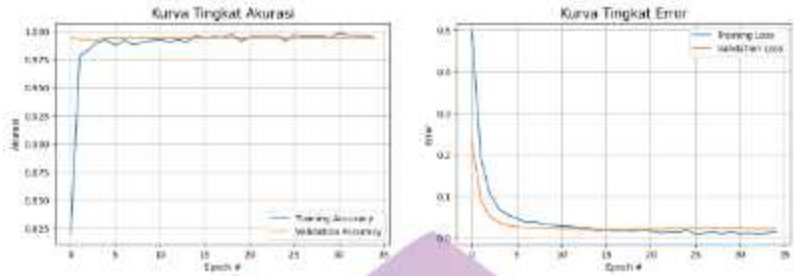
$$\textbf{Precision} = \frac{\mathbf{1{,}0000 + 0{,}9964}}{\mathbf{2}} = \mathbf{0{,}9982} \quad \mathbf{(99{,}82\%)}$$

$$\text{Recall } (\textit{with\_mask}) = \frac{276}{276 + 1} = 0{,}9964 \quad (99{,}64\%)$$

$$\text{Recall } (\textit{without\_mask}) = \frac{273}{273 + 0} = 1{,}0000 \quad (100{,}00\%)$$

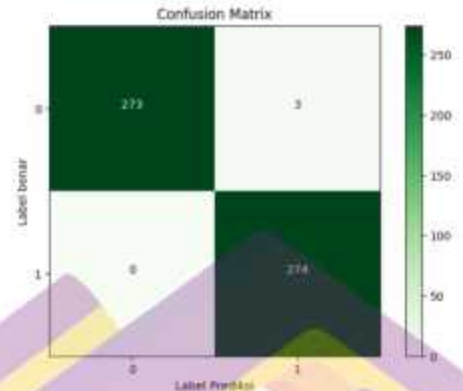
$$\textbf{Recall} = \frac{\mathbf{0{,}9964 + 1{,}0000}}{\mathbf{2}} = \mathbf{0{,}9982} \quad \mathbf{(99{,}82\%)}$$

Dari perhitungan diatas, pengujian model yang telah terbentuk mendapatkan hasil nilai *accuracy* 0,9982 (99,82%), nilai *precision* 0,9982 (99,82%), dan nilai *recall* 0,9982 (99,82%). Pada skenario 2 pelatihan model kedua menggunakan *max epoch* 35 dan *batch size* 40. Hasil dari proses pelatihan model kedua dapat ditampilkan kedalam grafik seperti berikut.



Gambar 4.36. Grafik tingkat akurasi dan grafik tingkat error pelatihan model 2

Pada gambar 4.36 menampilkan gambar bagian grafik tingkat akurasi dimana *training* dan *validation accuracy* dari *epoch* pertama hingga 35 mendapatkan nilai *training accuracy* 0,8200 mengalami peningkatan mencapai nilai 0,9945. Sedangkan untuk *validation accuracy* mulai dari 0,9945 mengalami perubahan nilai yang tidak terlalu signifikan pada setiap proses pembelajarannya hingga pada akhir pelatihan mendapatkan hasil nilai 0,9945. Pada gambar 4.36 bagian grafik tingkat error menampilkan grafik nilai *loss* dari *epoch* pertama hingga 35 mendapatkan nilai *training loss* 0,4922 mengalami penurunan mencapai nilai 0,0148. Sedangkan untuk *validation loss* mulai dari 0,2307 mengalami penurunan mencapai nilai 0,0222. Waktu proses pelatihan model yang diperlukan 18 menit 25 detik atau 1105 detik.



Gambar 4.37. Evaluasi *confusion matrix* pelatihan model 2

Evaluasi yang dilakukan dengan menggunakan 550 data uji dibagi dalam kelas *with_mask* 273 data dan *without_mask* 274 data. Dimana diketahui 273 data dalam kelas *with_mask* diprediksi dengan benar, 3 data diprediksi salah. Selain itu diketahui 274 data dalam kelas *without_mask* diprediksi dengan benar, 0 data diprediksi salah. Dengan melalui proses prediksi berikut maka dapat dihitung jumlah nilai dari *accuracy*, *precision*, dan *recall*.

$$\textbf{Accuracy} = \frac{\mathbf{273 + 274}}{\mathbf{550}} = \mathbf{0{,}9945}\ \ \mathbf{(99{,}45\%)}$$

$$\text{Precision } (\textit{with\_mask}) = \frac{273}{273 + 3} = 0{,}9891\ \ (98{,}91\%)$$

$$\text{Precision } (\textit{without\_mask}) = \frac{274}{274 + 0} = 1{,}0000\ \ (100{,}00\%)$$

$$\textbf{Precision} = \frac{\mathbf{0{,}9891 + 1{,}0000}}{\mathbf{2}} = \mathbf{0{,}9946}\ \ \mathbf{(99{,}46\%)}$$

$$\text{Recall } (\textit{with\_mask}) = \frac{273}{273 + 0} = 1{,}0000\ \ (100{,}00\%)$$

$$\text{Recall } (\textit{without\_mask}) = \frac{274}{274 + 3} = 0{,}9892 \quad (98{,}92\%)$$

$$\textbf{Recall} = \frac{\mathbf{1{,}0000 + 0{,}9892}}{\mathbf{2}} = \mathbf{0{,}9946} \quad \mathbf{(99{,}46\%)}$$

Dari perhitungan diatas, pengujian model yang telah terbentuk mendapatkan hasil nilai *accuracy* 0,9945 (99,45%), nilai *precision* 0,9946 (99,46%), dan nilai *recall* 0,9946 (99,46%).

4.4. Analisis Evaluasi dan Hasil Penelitian

Dimulai dari membangun dataset dan dilakukannya preprocessing data kemudian menentukan skenario yang digunakan dan pelatihan model yang dilakukan dengan menggunakan masing-masing skenario percobaan didapatkan hasil yang dapat dievaluasikan lebih lanjut. Model yang telah dilatih merupakan hasil dari pengetahuan dari pembelajaran citra dari gambar wajah bermasker dan tidak bermasker. Skenario pertama menggunakan arsitektur VGG16, skenario kedua arsitektur ResNet50V2, skenario ketiga Xception, dan skenario keempat EfficientNetV2S. Evaluasi dari hasil pelatihan model dapat ditampilkan dalam tabel seperti berikut.

Tabel 4.2. Tabel Evaluasi Hasil Skenario Percobaan

Epoch 25 & Batch Size 30								
Skenario	Training		Validation		Testing			Time (s)
	Acc	Loss	Acc	Loss	Acc	Prec	Rec	
S1	0,991	0,0290	0,9945	0,0253	0,9945	0,9945	0,9946	901
S2	0,9982	0,0054	0,9891	0,0147	0,9891	0,9891	0,9891	929
S3	0,9959	0,0130	0,9927	0,0276	0,9927	0,9928	0,9928	867
S4	0,9986	0,0100	0,9982	0,0049	0,9982	0,9982	0,9982	975
Epoch 35 & Batch Size 40								
Skenario	Training		Validation		Testing			Time (s)
	Acc	Loss	Acc	Loss	Acc	Prec	Rec	
S1	0,9932	0,0238	0,9927	0,4404	0,9927	0,9927	0,9928	1186
S2	0,9991	0,0045	0,9873	0,0144	0,9873	0,9873	0,9873	1037
S3	0,9964	0,0132	0,9909	0,0300	0,9909	0,9909	0,9910	1097
S4	0,9945	0,0148	0,9945	0,0222	0,9945	0,9946	0,9946	1105

Pada tabel 4.2 menunjukkan evaluasi hasil dari semua skenario percobaan yang telah dijalankan. Dari hasil skenario yang telah dilakukan didapatkan nilai *accuracy* tertinggi dari keempat skenario yaitu terdapat pada skenario keempat yaitu arsitektur EfficientNetV2S menggunakan *epoch* 25 & *batch size* 30 yang memiliki hasil output akurasi yaitu 0,9982. Namun pada *epoch* 35 & *batch size* 40 akurasi menurun pada nilai 0,9945. Kemudian jika dilihat pada nilai *validation loss* skenario keempat EfficientNetV2S menggunakan *epoch* 25 & *batch size* 30 masih mendapatkan nilai terbaik karena menghasilkan output nilai terendah daripada arsitektur lainnya yaitu dengan nilai 0,0049. Namun pada *epoch* 35 & *batch size* 40 mendapatkan nilai *validation loss* pada skenario keempat mendapatkan nilai 0,0222. Nilai tersebut masih diatas skenario percobaan kedua yaitu arsitektur ResNet50V2 dengan nilai *validation loss* mencapai 0,0144. Sedangkan dari segi waktu komputasi pada percobaan *epoch* 25 & *batch size* 30 didapatkan arsitektur

paling ringan pada percobaan ketiga menggunakan arsitektur Xception dengan waktu komputasi mencapai 14 menit 27 detik atau 867 detik. Selain itu pada percobaan *epoch* 35 & *batch size* 40 didapatkan arsitektur paling ringan yaitu pada skenario percobaan kedua menggunakan arsitektur ResNet50V2 dengan waktu komputasi mencapai 17 menit 17 detik atau 1037 detik. Untuk melihat perbandingan lebih jelas dari keempat skenario percobaan maka penulis membuat beberapa perbandingan dalam bentuk grafik.

4.4.1. Perbandingan Hasil *Training* dan *Validation Loss*

Dalam penelitian ini dilakukan perbandingan pada nilai *loss / error* pada saat pelatihan model *training* dan *testing* data. Berikut terdapat dua grafik dari masing-masing percobaan.

Gambar 4.38. Grafik *training* & *validation loss epoch* 25 *&* *BS* 30

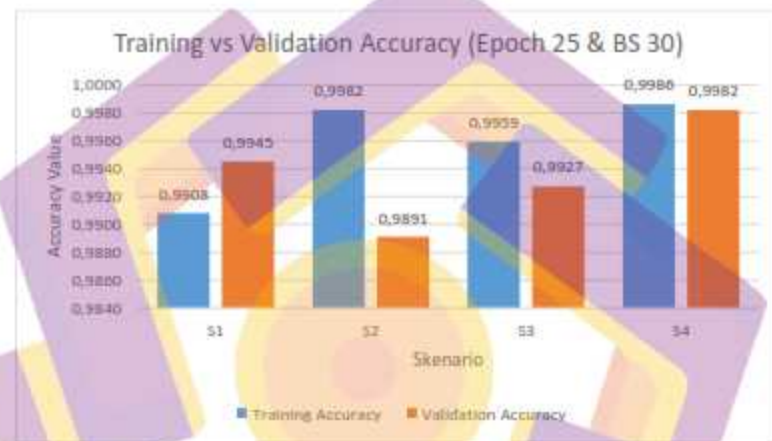
Gambar 4.39. Grafik *training* & *validation loss epoch* 35 & *BS* 40

Pada gambar 4.38 memperlihatkan grafik hasil perbandingan proses dari *training* dan *validation loss* pada setiap skenario dengan *epoch* 25 & *batch size* 30 terlihat perbedaan keempat skenario memiliki rata-rata nilai yang tidak jauh berbeda. Hal tersebut mengindikasikan bahwa performa setiap model baik dan termasuk pada kondisi *good fit*. Pada skenario pertama dan keempat lebih baik dari pada skenario kedua dan ketiga karena memiliki nilai *validation loss* lebih rendah dari pada *training loss*. Hal tersebut terlihat bahwa pada skenario tesebut model dapat membaca data baru yang diujikan dengan lebih baik.

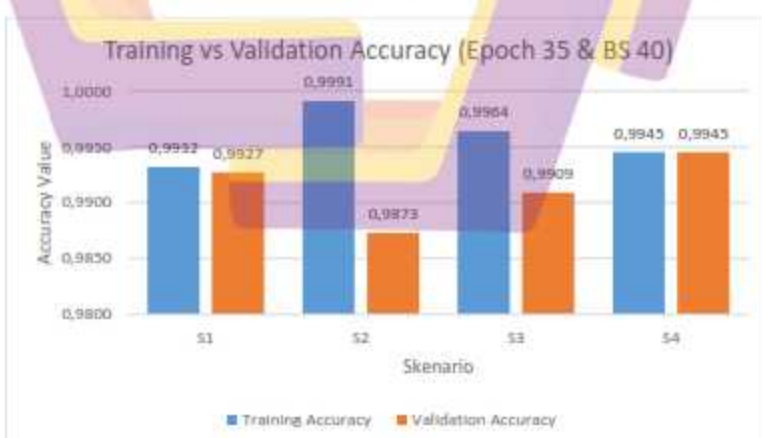
Sedangkan pada gambar 4.39 memperlihatkan grafik hasil perbandingan proses dari *training* dan *validation loss* pada setiap skenario pada *epoch* 35 & *batch size* 40 terlihat sama memiliki rata-rata nilai yang tidak jauh berbeda. Namun pada skenario pertama terlihat lebih baik dari pada skenario kedua, ketiga, dan keempat karena memiliki nilai *validation loss* lebih rendah dari pada *training loss*.

4.4.2. Perbandingan Hasil *Training* dan *Validation Accuracy*

Dalam penelitian ini dilakukan perbandingan pada nilai *accuracy*. Nilai *accuracy* dapat menggambarkan performa bagus atau tidaknya suatu model klasifikasi yang kedepanya ketika digunakan untuk memprediksi data/objek baru. Berikut terdapat dua grafik dari masing-masing percobaan.



Gambar 4.40. Grafik *training* & *validation accuracy epoch* 25 *&* *BS* 30



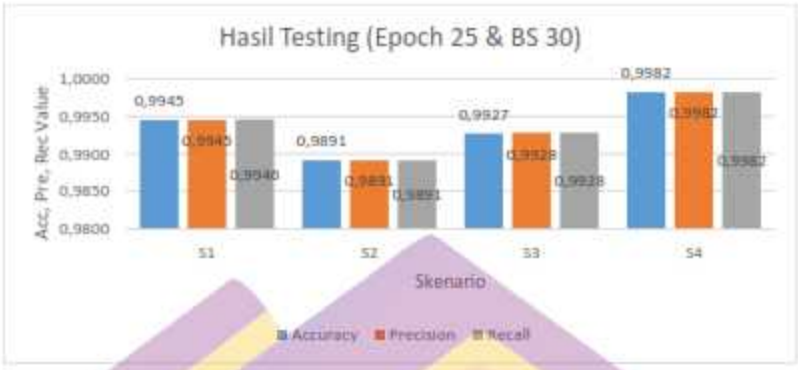
Gambar 4.41. Grafik *training* & *validation accuracy epoch* 35 *&* *BS* 40

Pada gambar 4.40 memperlihatkan grafik hasil perbandingan proses dari *training* dan *validation* pada setiap skenario dengan *epoch* 25 & *batch size* 30 terlihat perbedaan skenario pertama memiliki nilai *validation accuracy* lebih baik dari pada *training accuracy*. Hal tersebut berbanding terbalik pada skenario kedua dan ketiga yang memiliki nilai *training accuracy* lebih tinggi dari pada *validation accuracy* serta pada skenario kedua terdapat *gap* antara kedua nilai tersebut. Namun pada skenario keempat terdapat jarak nilai yang hampir mirip. Hal ini mengindikasikan bahwa model termasuk dalam kategori baik.

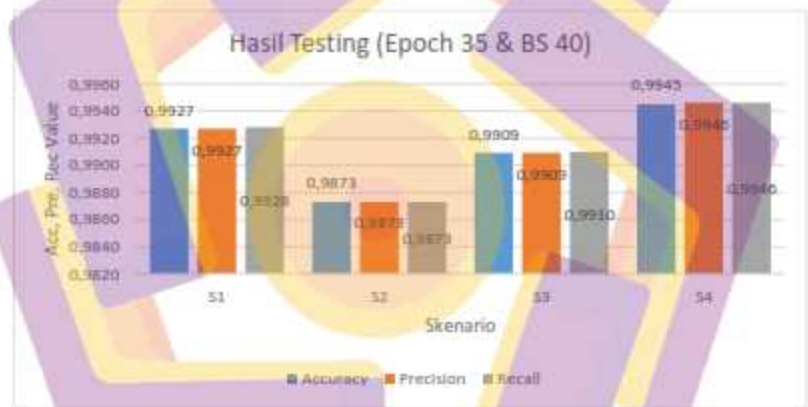
Kemudian pada gambar 4.41 memperlihatkan grafik hasil perbandingan proses dari *training* dan *validation accuracy* setiap skenario pada *epoch* 35 & *batch size* 40 terlihat keempat skenario memiliki nilai *validation accuracy* yang lebih rendah dari pada percobaan pertama yang menggunakan *epoch* 25 & *batch size* 40.

4.4.3. Perbandingan Hasil *Testing*

Dalam percobaan ini dilakukan perbandingan hasil pelatihan model yang telah dibuat berdasarkan dengan nilai *accuracy*, *precision*, dan *recall* dari keempat skenario percobaan dan masing-masing dua kali pelatihan model sesuai dengan *epoch* & *batch size* yang telah ditentukan.



Gambar 4.42. Hasil *testing*



Gambar 4.43. Hasil *testing*

Pada gambar 4.42 memperlihatkan nilai dari *accuracy, precision,* dan *recall* antara keempat skenario dari grafik terlihat bahwa skenario keempat mendapatkan nilai *accuracy, precision,* dan *recall* tertinggi dari pada skenario lainnya dan skenario kedua mendapatkan nilai *accuracy, precision,* dan *recall* terendah. Hal tersebut mengindikasikan hasil *testing* yang sama pada gambar 4.43 dari semua skenario percobaan. Skenario keempat masih mendapatkan nilai *accuracy, precision,* dan *recall* tertinggi dari pada skenario lainnya walaupun mengalami

penurunan pada pelatihan model kedua dengan *epoch* 35 & *batch size* 40. Maka dari keempat skenario percobaan yang dilakukan dan model diuji sebanyak dua kali. Dari kedua gambar grafik 4.42 dan 4.43 tersebut skenario keempat yang mendapatkan nilai *accuracy, precision,* dan *recall* tertinggi.

4.4.4. Perbandingan Waktu Komputasi

Dalam penelitian ini dilakukan pencatatan terhadap waktu komputasi yang dibutuhkan dalam proses pembelajaran model untuk setiap skenario percobaan. Berikut merupakan grafik perbandingan waktu komputasi pada setiap skenario percobaan dan masing-masing dua kali pelatihan model sesuai dengan *epoch* & *batch size* yang telah ditentukan.

Gambar 4.44. Perbandingan waktu komputasi

Pada gambar 4.44 memperlihatkan perbandingan waktu komputasi dari keempat skenario percobaan dengan *epoch* 25 & *batch size* 30. Dari grafik tersebut terlihat bahwa skenario ketiga yaitu arsitektur Xception memiliki waktu komputasi

terkecil, diikuti dengan skenario pertama arsitektur VGG16, skenario kedua arsitektur ResNet50V2, dan skenario keempat arsitektur EfficientNetV2S. Sedangkan skenario percobaan dengan *epoch* 35 & *batch size* 40 skenario kedua menjadi yang terkecil diikuti oleh skenario ketiga, keempat dan skenario pertama.

4.4.5. Perbandingan dan Evaluasi Hasil Pada Penelitian Sebelumnya

Pada beberapa pembahasan terhadap hasil percobaan diatas relevan dengan penelitian yang telah dilakukan sebelumnya, seperti penelitian yang telah dilakukan oleh (Agarwal, Itondia, & Mishra, 2023) dengan melakukan perbandingan 6 arsitektur CNN yang terdiri dari VGG16, VGG19, ResNet50, ResNet101, ResNet152, Xception. Hasil yang diperoleh dari dataset 4653 citra yang digunakan untuk pelatihan model didapatkan hasil akurasi dengan arsitektur Xception mendapatkan nilai 97,74%, VGG16 dengan nilai 99,14%, VGG19 dengan nilai 98,92%, ResNet50 dengan nilai 99,03%, ResNet101 dengan nilai 99,57%, dan akurasi tertinggi didapatkan oleh arsitektur ResNet152 dengan nilai 99,78%. Sedangkan pada penelitian ini menggunakan dataset dengan jumlah 2750 data dibagi 1380 gambar wajah bermasker dan 1370 gambar wajah tidak bermasker dengan arsitektur yang sama yaitu VGG16 mendapatkan akurasi sebesar 99,45% dan arsitektur yang lebih baru ResNet50V2 mendapatkan akurasi sebesar 98,91% keduanya dilakukan skenario percobaan dengan jumlah *epoch* 25 & *batch size* 30. Pada bagian arsitektur ResNet50V2 mendapatkan nilai lebih rendah dari pada penelitian sebelumnya. Namun pada arsitektur EfficientNetV2S mendapatkan nilai akurasi 99.82% lebih tinggi dari penelitian sebelumnya.

Selanjutnya pada penelitian yang dilakukan oleh (Ramadhan, et al., 2023) menggunakan arsitektur VGG11, ResNet50, InceptionV3, EfficientNetB4, dan YOLO. Hasil nilai akurasi tertinggi terdapat pada arsitektur EfficientNetB4 mendapatkan nilai tertinggi 95,77% dengan waktu 52 menit selanjutnya arsitektur dengan segi efektifitas waktu ResNet50 merupakan arsitektur tercepat dalam proses pelatihan model dengan waktu 25 menit atau 1500 detik. Sedangkan pada penelitian ini, didapatkan arsitektur EfficientNetV2S percobaan yang dievaluasikan memiliki akurasi tertinggi dengan nilai 0,9982 atau 99,82% dengan menggunakan *epoch* 25 dan *batch size* 30 memakan waktu komputasi 975 detik. Selanjutnya arsitektur paling ringan terdapat pada Xception dengan waktu komputasi 867 detik.

Kemudian pada penelitian yang dilakukan oleh (Naufal & Kusuma, 2021) dengan melakukan perbandingan 4 arsitektur CNN termasuk didalamnya yaitu VGG16 dan Xception. Menggunakan dataset dari kaggle Larxel dengan jumlah data gambar 3725 wajah bermasker dan 3828 wajah tidak bermasker. Hasil dari penelitian ini dari arsitektur Xception mendapatkan nilai akurasi terbaik dengan validasi akurasi dan waktu komputasi 0.988 dan 18.274 detik. VGG16 mendapatkan validasi akurasi 0,582 dan 24705 detik. Sedangkan pada penelitian ini menggunakan dataset yang lebih sedikit dengan jumlah 1380 gambar wajah bermasker dan 1370 gambar wajah tidak bermasker dengan arsitektur yang sama Xception mendapatkan akurasi sebesar 0,9959 dan waktu komputasi 867 detik. Selain itu arsitektur VGG16 mendapatkan akurasi sebesar 0,9945 dan waktu komputasi 901 detik.

4.4.6. Pengujian Tambahan

Skenario pengujian tambahan pada penelitian berikut merupakan masukan dari salah satu dosen penguji untuk mengetahui nilai signifikansi hasil penelitian berikut dengan penelitian sebelumnya menggunakan Anova dikarenakan terdapat jarak nilai peningkatan akurasi yang tipis. Hasil pengujian yang telah peneliti lakukan dapat dilihat pada tabel 4.3 dan 4.4.

Tabel 4.3. Tabel pengujian tambahan

Referensi Penelitian	VGG	ResNet	Xception	EfficientNet
Nalda K.N (2023)	0,9945	0,9891	0,9959	0,9982
Agarwal et al. (2023)	0,9914	0,9978	0,9774	-
Ramadhan et al. (2023)	0,8438	0,8441	-	0,9577
Naufal et al. (2021)	0,5820	-	0,9880	-
Nilai Rata-rata	0,8529	0,9437	0,9871	0,9780

Tabel 4.4. Tabel pengujian Anova

Anova: Single Factor

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
VGG	4	3,4117	0,8529	0,0376
ResNet	3	2,8310	0,9437	0,0075
Xception	3	2,9613	0,9871	0,0001
EfficientNet	2	1,9559	0,9780	0,0008

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0,0387	3	0,0129	0,8025	0,5267	4,0662
Within Groups	0,1286	8	0,0161			
Total	0,1673	11				

Seperti yang terlihat pada tabel 4.3 dan 4.4 peneliti melakukan pengujian tambahan menggunakan Anova dengan membandingkan hasil akurasi setiap arsitektur yang digunakan. Hasil dari pengujian yang dilakukan dapat dilihat pada nilai *P-value* dengan nilai 0,5267 dimana pada pengujian ini jika nilai *P-value* <0,05 maka terdapat perbedaan yang signifikan. Sedangkan jika nilai >0,05 maka tidak ada perbedaan yang signifikan pada hasil penelitian berikut dengan penelitian sebelumnya. Sedangkan hasil pengujian terdapat *output* dengan nilai *P-value* 0,5267 maka dapat ditarik kesimpulan bahwa peningkatan akurasi yang terdapat pada penelitian ini tidak terdapat nilai yang signifikan dengan hasil akurasi penelitian sebelumnya.

BAB V

PENUTUP

5.1. Kesimpulan

Berdasarkan dengan hasil skenario percobaan yang telah dilakukan terhadap keempat arsitektur dengan masing-masing dua kali pelatihan model dapat disimpulkan sebagai berikut.

1. Dari hasil percobaan menggunakan 4 arsitektur algoritma CNN dengan arsitektur EfficientNetV2S mendapatkan nilai kinerja tertinggi dengan *accuracy* 99,82%, *precision* 99,82%, dan *recall* 99,82%. Jika dibandingkan dengan arsitektur VGG16 dengan nilai kinerja *accuracy* 99,45%, *precision* 99,45%, dan *recall* 99,46%. Xception dengan nilai kinerja *accuracy* 99,27%, *precision* 99,28%, dan *recall* 99,28%. Sedangkan ResNet50V2 dengan nilai kinerja *accuracy* 98,91%, *precision* 98,91%, dan *recall* 98,91%.
2. Pada penelitian ini penggunaan arsitektur lebih baru seperti EfficientNetV2S dapat memberikan hasil lebih baik dari segi kinerja nilai *accuracy* dibandingkan dengan arsitektur lainnya. Namun dari segi waktu komputasi arsitektur EfficientNetV2S masih kurang efektif dibandingkan dengan arsitektur lainnya.
3. Penggunaan arsitektur lebih baru seperti EfficientNetV2S dapat memberikan performa hasil lebih baik dikarenakan arsitektur EfficientNetV2S telah dirancang dengan efisiensi parameter lebih baik.

Dalam arsitektur ini peningkatan skala resolusi dan strategi pemodelan pada arsitektur membantu dalam meningkatkan akurasi pengenalan gambar. Dengan mempertahankan keseimbangan antara keefisienan dan akurasi, EfficientNetV2S dapat memberikan performa yang baik dalam tugas klasifikasi gambar. EfficientNetV2S dapat mengambil manfaat dari transfer learning. Dengan menggunakan bobot yang telah dilatih sebelumnya pada dataset gambar yang besar, EfficientNetV2S dapat menggeneralisasi dan mengekstraksi fitur dari gambar baru dengan efektif.

DAFTAR PUSTAKA

PUSTAKA BUKU

Andrejevic, M., & Selwyn, N. (2019). Learning, Media and Technology. In *Facial recognition technology in schools: critical questions and concerns* (pp. 115-128).

Christin, S., Hervet, É., & Lecomte, N. (2019). Applications for deep learning in ecology. Dalam A. Ellison, B. O'Hara, & N. C. Lecomte, *Methods in Ecology and Evolution* (hal. 1632-1644). BRITISH ECOLOGICAL SOCIETY.

Deepan, P., & Sudha, L. (2020). Object Classification of Remote Sensing Image Using Deep Convolutional Neural Network. Dalam D. Peter, A. H. Alavi, B. Javadi, & S. L. Fernandes, *The Cognitive Approach in Cloud Computing and Internet of Things Technologies for Surveillance Tracking Systems* (hal. 107-120). ACADEMIC PRESS.

Mahesh, B. (2019). *Machine Learning Algorithms - A Review.* International Journal of Science and Research (IJSR).

Scarpino, M. (2018). *TensorFlow For Dummies.* New Jersey: John Wiley & Sons.

Sinha, A. (2016). Hadoop in the Cloud to Analyze Climate Datasets. Dalam T. C. Vance, N. Merati, C. Yang, & M. Yuan, *Cloud Computing in Ocean and Atmospheric Sciences* (hal. 245-276). ACADEMIC PRESS.

PUSTAKA MAJALAH, JURNAL ILMIAH ATAU PROSIDING

Agarwal, C., Itondia, P., & Mishra, A. (2023). A novel DCNN-ELM hybrid framework for face mask detection. *Intelligent Systems with Applications*.

Agustiani, D. (2019). *Implementasi Machine Learning dan Computer Vision pada Pengembangan Sistem Otomasi Klasifikasi dan Perhitungan Kendaraan*.

Arrofiqoh, E. N., & Harintaka. (2018). *IMPLEMENTASI METODE CONVOLUTIONAL NEURAL NETWORK UNTUK KLASIFIKASI TANAMAN PADA CITRA RESOLUSI TINGGI*.

Asro'i, A., & Februariyanti, H. (2022). Data Splitting. *Analisis Sentimen Pengguna Twitter terhadap Perpanjangan PPKM Menggunakan Metode K-Nearest Neighbor*, 20.

Bi, Q., Goodman, K. E., Kaminsky, J., & Lessler, J. (2019). What is Machine Learning? A Primer for the Epidemiologist. *American Journal of Epidemiology*, 2222–2239.

Budiman, B., Lubis, C., & Perdana, N. J. (2021). *Pendeteksian Penggunaan Masker Wajah Dengan Metode Convolutional Neural Network*.

Christian, H. D. (2021). Pre-Trained Model EfficientNet. *Klasifikasi Status Hidrasi Berdasarkan Warna Urine Menggunakan CNN pada Aplikasi Berbasis Web*, 23.

Firdaus, R. (2020). Computer Vision. *Rancang Bangun Sistem Navigasi Multirotor Berbasis Waypoint Dan Computer Vision*.

Fonda, H. (2020). *KLASIFIKASI BATIK RIAU DENGAN MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORKS (CNN)*.

Halim, M. R. (2022). *TEKNIK AUGMENTASI DAN MODIFIKASI ARSITEKTUR XCEPTION DENSE LAYER PADA CITRA RETINA UNTUK KLASIFIKASI PENYAKIT GLAUCOMA*.

Harjoseputro, Y. (2018). BAB I. PENDAHULUAN. *CONVOLUTIONAL NEURAL NETWORK (CNN) UNTUK PENGKLASIFIKASIAN AKSARA JAWA*, 1-2.

Iglesias, F., Zseby, T., Ferreira, D., & Zimek, A. (2019). MDCGen: Multidimensional Dataset Generator for Clustering. *Journal of Classification*, 599-618.

Karsito, & Susanti, S. (2019). Confusion Matrix. *PENGAJUAN KREDIT RUMAH DENGAN ALGORITMA NAÏVE BAYES DI PERUMAHAN AZZURA RESIDENCIA*, 44.

Khaeriyah, R. (2019). *IMPLEMENTASI METODE CONVOLUTIONAL NEURAL NETWORK MENGGUNAKAN TENSORFLOW DALAM MENDETEKSI SEBUAH OBJEK*.

Kocacinar, B., Tas, B., Akbulut, F. P., Catal, C., & Mishra, D. (2022). *A Real-Time CNN-Based Lightweight Mobile Masked Face Recognition System*.

Kurniadi, D., Sugiyono, A., & Wardaya, L. A. (2021). *Pattern Recognition of Human Face With Photos Using KNN Algorithm*, 17-25.

Liu, X., Deng, Z., & Yang, Y. (2018). *Recent progress in semantic image segmentation*. Diambil kembali dari Springer Link.

Ma, J., Jiang, X., Fan, A., Jiang, J., & Yan, J. (2021). Image Matching from Handcrafted to Deep Features: A Survey. *International Journal of Computer Vision*, 23–79.

Mahmud, K. H., Adiwijaya, & Faraby, S. A. (2019). Augmentasi Data. *Klasifikasi Citra Multi-Kelas Menggunakan Convolutional Neural Network*, 2127.

Mehmood, A. (2021). CNN Architecture. *Efficient Anomaly Detection in Crowd Videos Using Pre-Trained 2D Convolutional Neural Networks*, 138289-138290.

Mittal, M., Verma, A., Kaur, I., Kaur, B., Sharma, M., Goyal, L. M., . . . Kim, T.-H. (2019). *An Efficient Edge Detection Approach to Provide Better Edge Connectivity for Image Analysis*, 33240 - 33255.

Muharram, R. F. (2021). *IMPLEMENTASI ARTIFICIAL INTELLIGENCE UNTUK DETEKSI MASKER SECARA REALTIME DENGAN TENSORFLOW DAN SSD MOBILENET BERBASIS PYTHON*.

Munantri, N. Z., Sofyan, H., & F, M. Y. (2019). *APLIKASI PENGOLAHAN CITRA DIGITAL UNTUK IDENTIFIKASI UMUR POHON*.

Naufal, M. F., & Kusuma, S. F. (2021). *PENDETEKSI CITRA MASKER WAJAH MENGGUNAKAN CNN DAN TRANSFER LEARNING*.

Permana, D. A. (2021). *PENDETEKSI WAJAH BERMASKER MENGGUNAKAN METODE FASTER R-CNN*.

Putri, T. S., Fikih, M. A., & Setyawan, N. (2020). *FACE MASK DETECTION COVID-19 USING CONVOLUTIONAL NEURAL NETWORK (CNN)*.

Qazi, E. U., Zia, T., & Almorjan, A. (2022). Proposed System Architecture. *Deep Learning-Based Digital Image Forgery Detection System*, 7.

Rahim, A. (2021). *ANALISIS KOMPARASI CONVOLUTIONAL NEURAL NETWORK (CNN) DENGAN SUPPORT VECTOR MACHINE (SVM) UNTUK KLASIFIKASI CITRA PENGGUNA MASKER*.

Rahimzadeh, M., & Attar, A. (2020). Neural networks. *A modified deep convolutional neural network for detecting COVID-19 and pneumonia from chest X-ray images based on the concatenation of Xception and ResNet50V2*, 3.

Rahman, & Afifana, F. (2020). *KLASIFIKASI INVASIVE DUCTAL CARCINOMA MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK*.

Ramadhan, M. V., Muchtar, K., Nurdin, Y., Oktiana, M., Fitria, M., Maulina, N., . . . Pardamean, B. (2023). Comparative analysis of deep learning models for detecting face mask. *Procedia Computer Science*, 48-56.

Ramba, L. S. (2020). Classification. *Perancangan Sistem Home Automation Dengan Kendali Perintah Suara Menggunakan Deep Learning Convolutional Neural Network (Dl-Cnn)*, 13.

Rochmawati, N., Hidayati, H. B., Yamasari, Y., Tjahyaningtijas, H. P., Yustanti, W., & Prihanto, A. (2021). Pendahuluan. *Analisa Learning rate dan Batch size Pada Klasifikasi Covid Menggunakan Deep learning dengan Optimizer Adam*, 44.

Said, A., Shoukat, S., Shehzad, K., Ahmad, I., Esmawi, A. A., Amin, A. H., & Tag-Eldin, E. (2022). EFFICIENTNET V2S. *A Deep Learning-Based Approach for the Diagnosis of Acute Lymphoblastic Leukemia*, 7.

Saputro, A., Mu'min, S., Lutfi, M., & Putri, H. (2022). VGGNet. *DEEP TRANSFER LEARNING DENGAN MODEL ARSITEKTUR VGG16 UNTUK KLASIFIKASI JENIS VARIETAS TANAMAN LENGKENG BERDASARKAN CITRA DAUN*, 612.

Shamrat, F. J., Billah, M. M., Islam, M. S., Chakraborty, S., Jubair, M. A., & Ranjan, R. (2021). *Face Mask Detection using Convolutional Neural Network (CNN) to reduce the spread of Covid-19*.

Suryawan, I. G. (2019). BAB I. *PEMBANGUNAN APLIKASI ALAT BANTU PROSES ANOTASI MENGGUNAKAN PROGRESSIVE WEB APPS*, 1.

Tan, M., & Le, Q. V. (2021). *EfficientNetV2: Smaller Models and Faster Training*.

Utomo, B. A. (2021). Split Data. *Klasifikasi COVID-19 Berdasarkan Foto Rontgen Dada Menggunakan EfficientNet*, 34.

Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep Learning for Computer Vision: A Brief Review. *Computational Intelligence and Neuroscience*.

Wasil, M., Harianto, & Fathurrahman. (2022). Pendahuluan. *Pengaruh Epoch pada Akurasi menggunakan Convolutional Neural Network untuk Klasifikasi fashion dan Furniture*, 54.

Wikarta, A., Suryo, I. B., & Effendi, M. K. (2020). Hasil dan Pembahasan. *Analisa Pengaruh Ukuran Testing Data dan Data Augmentation pada Tingkat Akurasi Deteksi Pemakaian Masker oleh Pengemudi Kendaraan menggunakan Deep Learning*, 20-24.

Wu, X., Sahoo, D., & C.H.Hoi, S. (2020). *Recent advances in deep learning for object detection*, 39-64.

Yudianto, M. R. (2021). *KLASIFIKASI CITRA WAYANG PUNAKAWAN MENGGUNAKAN METODE GAUSSIAN FILTER DAN ALGORITMA CONVOLUTIONAL NEURAL NETWORK*.

Zhang, H., Tang, J., Wu, P., Li, H., & Zeng, N. (2023). A novel attention-based enhancement framework for face mask detection in complicated scenarios. *Signal Processing: Image Communication*.

PUSTAKA ELEKTRONIK

Dharmadi, R. (2018, 05 09). *CNN: Beyond Image Classification*. Diambil kembali dari Medium: https://medium.com/nodeflux/cnn-beyond-image-classification-3b9b0af021a9

Dharmadi, R. (2018, 06 27). *Convolutional Neural Net untuk Deteksi Objek*. Diambil kembali dari https://medium.com/@richad_/convolutional-neural-net-untuk-deteksi-objek-f14d72f11ba6

Nisa. (2022, 04 04). *Mengenal Apa Itu Society 5.0 dan Contoh Penerapannya dalam Berbagai Bidang*. Diambil kembali dari Inmarketing.id: https://inmarketing.id/society-5-0-adalah.html

Nursyafitri, G. D. (2022, 04 12). *Kenali Penggunaan Computer Vision dalam Data Science*. Diambil kembali dari DQLab.id: https://dqlab.id/kenali-penggunaan-computer-vision-dalam-data-science

Raharja, A. (2022, 09 29). *Dataset Adalah: Pengertian, Tipe, Perbedaan dengan Database, dan 10 Web Penyedia*. Diambil kembali dari EKRUT Media: https://www.ekrut.com/media/dataset-adalah

Salim, A. (2020, 04 02). *Object Detection (Case: Plat Detection)*. Diambil kembali dari Medium: https://medium.com/bisa-ai/object-detection-case-plat-detection-7cb5f53682ae

Universitas Medan Area. (2022, 01 11). *Apa Itu Era Society 5.0 dan Apa Perbedaannya dengan Era Industri 4.0?* Diambil kembali dari Biro Administrasi Registrasi Kemahasiswaan dan Informasi - Universitas Medan Area: https://barki.uma.ac.id/2022/01/11/apa-itu-era-society-5-0-dan-apa-perbedaannya-dengan-era-industri-4-0/