

**DETEKSI PENGGUNAAN MASKER MENGGUNAKAN METODE
CONVOLUTIONAL NEURAL NETWORK (CNN) PADA STUDI KASUS
COVID-19
SKRIPSI**

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
Program Studi Informatika



Oleh:

George Jusen
185314072

**Fakultas Sains dan Teknologi
Universitas Sanata Dharma
Yogyakarta
2023**

HALAMAN PERSETUJUAN

SKRIPSI

**“DETEKSI PENGGUNAAN MASKER MENGGUNAKAN METODE
CONVOLUTIONAL NEURAL NETWORK (CNN)
PADA STUDI KASUS COVID-19”**

Disusun oleh :

George Jusen

NIM : 185314072

Yogyakarta, 2023

Dosen Pembimbing,

Ir. Kartono Pinaryanto S.T., M.Cs

HALAMAN PENGESAHAN

DETEKSI PENGGUNAAN MASKER MENGGUNAKAN METODE *CONVOLUTIONAL NEURAL NETWORK* (CNN)

PADA STUDI KASUS COVID-19

Dipersiapkan dan ditulis oleh :

GEORGE JUSEN

NIM : 185314072

Telah dipertahankan di depan panitia dan penguji

Pada tanggal, 2023

Dan dinyatakan telah memenuhi syarat.

Susunan Panitia Penguji

Nama Lengkap	Tanda Tangan
Ketua : Rosalia Arum Kumalasanti, M.T.
Sekretaris : Ir. Kartono Pinaryanto, S.T., M.Cs
Anggota : Cyprianus Kuntoro Adi, S.J. M.A., M.Sc., Ph.D.

Yogyakarta, 2023

Fakultas Sains dan Teknologi

Universitas Sanata Dharma

Dekan,

Ir. Drs. Haris Sriwindono, M.Kom., Ph.D.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Coronavirus Disease (COVID-19) merupakan penyakit menular yang disebabkan oleh virus *Severe Acute Respiratory Syndrome Coronavirus 2* (SARS-CoV-2). Terdapat banyak jenis virus corona yang telah terdeteksi, seperti *Middle East Respiratory Syndrome* (MERS), dan varian baru COVID-19, termasuk *Omricon* yang pertama kali dilaporkan di Afrika Selatan pada November 2021. COVID-19 menyebar dengan sangat cepat melalui tetesan udara dan tetesan air liur saat seseorang yang terinfeksi batuk, bersin, berbicara, bernyanyi, atau berteriak. Oleh karena itu, sangat penting untuk mempraktikkan tindakan pencegahan seperti mencuci tangan secara teratur, menghindari kerumunan, memakai masker, dan menjaga jarak sosial guna mengurangi risiko penyebaran virus. Diketahui bahwa virus COVID-19 dapat menyebar dengan cepat. Ketika terinfeksi, seseorang akan mengalami beberapa gejala umum, termasuk gangguan pernapasan akut seperti demam, batuk, dan sesak napas. Kasus pertama COVID-19 dilaporkan terkait dengan penemuan di pasar ikan Wuhan, China pada akhir Desember 2019, yang melibatkan 27 orang yang mengalami gejala demam dan infeksi saluran pernapasan (*Pedoman Pencegahan Dan Pengendalian CORONAVIRUS DISEASE (COVID-19) Revisi Ke-5 - Protokol / Covid19.Go.Id*, n.d., p. 19)



Gambar 1. Berita Covid-19 muncul pertama kali di Wuhan

Pada Gambar 1 dalam berita tersebut Virus Corona atau COVID-19 pertama kali ditemukan di Wuhan, China pada akhir 2019. Kasus ini diduga berkaitan dengan pasar hewan Huanan di Wuhan yang menjual berbagai jenis daging binatang, termasuk yang tidak biasa dikonsumsi, misal ular, kelelawar, dan berbagai jenis tikus¹². Virus Corona atau COVID-19 diduga dibawa kelelawar dan hewan lain yang dimakan manusia hingga terjadi penularan. *Coronavirus* sebetulnya tidak asing dalam dunia kesehatan hewan, tapi hanya beberapa jenis yang mampu menginfeksi manusia hingga menjadi penyakit radang paru. Sebelum COVID-19 mewabah, dunia sempat heboh dengan SARS dan MERS, yang juga berkaitan dengan virus Corona.

Pada Desember 2019, pneumonia atau radang paru-paru misterius mulai menyebar di Wuhan, China. Pada awalnya, virus ini dianggap sebagai pneumonia biasa, tetapi kemudian diketahui bahwa virus ini sangat menular dan dapat menyebabkan kematian. Virus ini kemudian menyebar ke negara-negara lain, dan pada 11 Maret 2020, Organisasi Kesehatan Dunia (WHO) mengumumkan bahwa COVID-19 adalah pandemi global. Sejak itu, virus ini telah menyebar ke seluruh dunia dan mempengaruhi berbagai aspek kehidupan, seperti ekonomi, kesehatan mental, dan kehidupan sosial.

Salah satu keharusan dalam menerapkan protokol kesehatan adalah dengan menggunakan masker ketika berada di luar rumah. Penggunaan masker merupakan cara untuk melindungi diri dari penyebaran virus dan mencegah penularan penyakit, sehingga setiap orang, baik yang sehat maupun bergejala, wajib menggunakan masker.

Bagi seseorang yang mengalami gejala infeksi pernapasan seperti batuk atau bersin, diduga kuat bahwa orang tersebut terinfeksi COVID-19 dan oleh karena itu disarankan bagi petugas kesehatan untuk menggunakan masker bedah. Sayangnya, masih banyak orang yang mengabaikan penggunaan masker saat memasuki tempat-tempat yang ramai. Oleh karena itu, diperlukan sebuah terobosan untuk mendeteksi penggunaan masker di lingkungan kampus atau tempat umum lainnya sehingga penggunaan masker dapat lebih terkontrol dan efektif.

Machine Learning adalah cabang ilmu kecerdasan buatan yang berkembang pesat dalam tahun terakhir, yang bertujuan untuk memungkinkan mesin dapat melakukan pekerjaannya dengan terampil dan mampu melakukan tugas – tugas tertentu secara otomatis melalui pembelajaran dari data yang diberikan. Pada *Machine Learning* terdapat tiga tipe yaitu *Supervised Learning*, *Unsupervised Learning* dan *Deep Learning*.

Deep Learning adalah suatu teknologi kecerdasan buatan yang memiliki serangkaian metode yang menggunakan multi-layer neural network untuk dapat secara otomatis mempelajari representasi data. Dalam hal ini, *Deep Learning* dapat dilatih untuk melakukan tugas seperti mendeteksi dan mengklasifikasikan objek. Terdapat beberapa algoritma yang dapat digunakan untuk melakukan pelatihan model *Deep Learning*, misalnya menggunakan arsitektur *MobileNet*, *VGGNet*, dan sebagainya untuk klasifikasi objek sedangkan untuk deteksi objek dapat menggunakan *YOLO*, *SSD ResNet*, *MTCNN*, dan sebagainya. Salah satu model *Deep Learning* yang populer adalah *Convolutional Neural Network* (CNN).

Dalam penyusunan penelitian tugas akhir “Deteksi Penggunaan Masker Menggunakan Metode *Convolutional Neural Network* (CNN) Pada Studi Kasus COVID-19” terdapat beberapa penelitian terdahulu yang berhubungan mengenai membahas klasifikasi obyek dengan citra

menggunakan *machine learning*. Penelitian tersebut dapat dilihat sebagai berikut:

- a. Menurut Dharmaputra, A., Cahyanti, M., Septian, M. R. D., & Swedia, E. R. (2021), dalam penelitian yang menggunakan jaringan saraf tiruan Mobilenetv2 berbasis Android secara real-time, berhasil dihasilkan sistem yang mampu mengenali apakah seseorang menggunakan masker atau tidak dengan tingkat akurasi mencapai 90%.
- b. Dalam penelitian yang dilakukan oleh Ahmad, F. L., Nugroho, A., & Suni, A. F. (2021) dengan menerapkan metode Haar Cascade, ditemukan bahwa sistem berbasis waktu nyata berhasil mengidentifikasi apakah seseorang sedang memakai masker atau tidak dengan tingkat ketepatan tertinggi mencapai 93,33% pada jarak 40 cm dan dalam kondisi pencahayaan yang tinggi.
- c. Hasil penelitian yang dilaporkan oleh Hapsari, Y. dan rekan-rekannya pada tahun 2022 menggunakan algoritma *Viola and Jones* menunjukkan bahwa sistem mampu mendeteksi keberadaan mulut dan hidung pada wajah yang menghadap ke depan dalam rentang jarak 50 cm hingga 100 cm, meskipun dalam kondisi pencahayaan yang rendah. Hal ini mengindikasikan apakah seseorang memakai masker (ketika mulut dan hidung terdeteksi) atau tidak (ketika tidak terdeteksi mulut dan hidung).

Namun, sistem tidak efektif dalam mendeteksi situasi di mana wajah miring atau sedang memalingkan pandangan.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, peneliti merumuskan masalah dalam penelitian ini sebagai berikut :

- a. Bagaimana implementasi algoritma *Convolutional Neural Network* dalam mendeteksi penggunaan masker.
- b. Berapa besar akurasi yang dihasilkan dari prediksi menggunakan *Convolutional Neural Network*

1.3 Batasan Masalah

Agar penelitian tugas akhir mendapatkan hasil yang optimal maka permasalahan dibatas sebagai berikut :

- a. Metode *Deep Learning* yang digunakan untuk pendeteksi masker terdiri dari satu komponen yaitu pendeteksi muka menggunakan *MTCNN*, serta algoritma klasifikasi wajah dengan *VGG16Net*
- b. Program dibuat untuk membedakan pengguna menggunakan masker atau tidak menggunakan masker dan bersifat tunggal.
- c. Program pendeteksi masker menggunakan bahasa pemrograman *Python* dengan Platform *Google Colaboratory*.
- d. Membangun model *Deep Learning* citra pendeteksi masker menggunakan *Framework Keras* dan *Tensorflow*.

1.4 Tujuan dan Manfaat Penelitian

Dalam Penelitian tugas akhir ini terdapat beberapa tujuan yaitu sebagai berikut.

- a. Mengembangkan sebuah sistem deteksi masker menggunakan *Convolutional Neural Network*
- b. Sistem ini diharapkan dapat secara akurat dan otomatis mendeteksi apakah seseorang menggunakan masker atau tidak menggunakan masker
- c. Untuk mendeteksi mana yang menggunakan masker atau tidak menggunakan masker

Manfaat yang diperoleh dari penelitian ini adalah:

- a. Bagi peneliti:
Mampu menerapkan ilmu yang didapatkan pada bangku perkuliahan untuk menyelesaikan tugas akhir.
- b. Bagi masyarakat:
Membantu mempercepat masyarakat dalam mendeteksi penggunaan masker di dalam suatu instansi seperti rumah sakit dan lain sebagainya
- c. Bagi Universitas:
Menambah kontribusi dalam ilmu informatika serta dapat menjadi referensi bagi mahasiswa lain yang sedang melakukan penelitian mengenai *Convolutional Neural Network*.

1.5 Sistematika Penelitian

Penelitian ini disusun secara sistematis yang tersusun dari beberapa bab diantaranya sebagai berikut:

1. BAB I PENDAHULUAN

Pada bagian ini menjelaskan tentang latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

2. BAB II DASAR TEORI

Pada bagian ini menjelaskan tentang tinjauan pustaka dan landasan teori yang mendasari penelitian ini.

3. BAB III METODE PENELITIAN

Pada bagian ini menjelaskan tentang jenis penelitian, prosedur penelitian, dan diagram alir penelitian.

4. BAB IV HASIL DAN PEMBAHASAN

Pada bagian ini menjelaskan tentang hasil penelitian dan pembahasan dari data yang telah diolah.

5. BAB V PENUTUP

Pada bagian ini berisikan kesimpulan yang diperoleh dari hasil penelitian dan pembahasan serta saran yang diberikan untuk mencapai hasil yang lebih baik.

BAB II

LANDASAN TEORI

2.1 Coronavirus Disease 2019 (COVID-19)

Coronavirus Disease 2019 (COVID-19) adalah penyakit menular yang disebabkan oleh jenis virus *Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2)*. Ini merupakan penemuan virus corona jenis baru yang belum pernah diidentifikasi sebelumnya pada manusia sebelum mulainya wabah di Wuhan, Tiongkok, bulan Desember 2019. Demam, batuk kering, dan rasa lelah merupakan gejala COVID-19 yang paling umum. Beberapa pasien mungkin juga mengalami gejala lain yang jarang seperti rasa nyeri dan sakit, hidung tersumbat, sakit kepala, konjungtivitis, sakit tenggorokan, diare, kehilangan indera rasa atau penciuman, ruam pada kulit, atau perubahan warna jari tangan atau kaki. Meskipun gejala-gejala ini muncul secara bertahap dan cenderung bersifat ringan, beberapa orang mungkin hanya mengalami gejala ringan meskipun telah terinfeksi. (*Pedoman Pencegahan Dan Pengendalian CORONAVIRUS DISEASE (COVID-19) Revisi Ke-5 - Protokol / Covid19.Go.Id, n.d.*)

2.1.1 Pencegahan dan Pengendalian COVID-19

Berdasarkan dalam buku pedoman pencegahan dan pengendalian COVID-19 (*Pedoman Pencegahan Dan Pengendalian CORONAVIRUS DISEASE (COVID-19) Revisi Ke-5 - Protokol / Covid19.Go.Id, n.d., p. 16*), Pemerintah Indonesia merilis panduan untuk mencegah dan mengendalikan

penyebaran Covid-19. Virus dapat menyebar melalui droplet dari satu orang ke orang lain melalui hidung, mulut, atau mata, sehingga dapat terjadi di mana saja dan kapan saja. Oleh karena itu, beberapa tindakan dapat dilakukan untuk mencegah penyebaran virus Covid-19 sebagai berikut :

- a. Menjaga kebersihan tangan dengan mencuci tangan menggunakan sabun dan air mengalir selama 40-60 detik, kemudian bilas dan keringkan dengan handuk bersih atau kertas sekali pakai. Jika tidak memungkinkan untuk mencuci tangan, alternatif lainnya adalah menggunakan pembersih tangan berbasis alkohol (*Handsanitizer*) minimal 20-30 detik.
- b. Menghindari penyebaran *droplet* dari bersin atau batuk, disarankan untuk menutup mulut dan hidung menggunakan tisu atau bagian dalam lengan atas. Kemudian, tisu yang digunakan harus dibuang ke tempat sampah yang tertutup dan tangan dicuci dengan sabun dan air mengalir atau menggunakan *handsanitizer*.
- c. Menjaga jarak minimal 1 meter dengan orang lain agar terhindar terkena *droplet* dari orang-orang yang batuk atau bersin.
- d. Menggunakan masker kain bila berada di luar rumah, setelah 4 jam dipakai, dan cuci hingga bersih setelah dipakai.
- e. Disarankan menggunakan masker jika ingin berobat ke fasyankes.

2.1.2 Pentingnya Penggunaan Masker

(*Pedoman Pencegahan Dan Pengendalian CORONAVIRUS DISEASE (COVID-19) Revisi Ke-5 - Protokol / Covid19.Go.Id*, n.d., p. 110) Seperti yang diketahui, virus *corona* dapat menyebar melalui *droplet* atau percikan saat seseorang berbicara, batuk, bersin, dan lain sebagainya. Oleh karena itu, masker digunakan untuk melindungi diri dari *droplet* yang dikeluarkan oleh orang lain agar tidak masuk ke dalam hidung dan mulut kita, begitu juga sebaliknya. Ada tiga jenis masker yang direkomendasikan untuk digunakan oleh masyarakat agar dapat membantu memutuskan penyebaran virus *corona*, antara lain sebagai berikut :

a. Masker kain

Sesuai anjuran dalam pedoman Kementerian Kesehatan RI, masyarakat disarankan menggunakan masker kain ketika bepergian keluar rumah, misalnya seperti sedang bekerja atau keperluan lain. Masker kain dapat meminimalisir untuk mencegah penularan virus *corona*.

b. Masker bedah

Masker ini sering di jumpai di tenaga medis saat sedang bertugas karena memiliki tiga fungsi utama agar pencegahan lebih efektif dan hanya sekali pakai.

c. Masker N95

Harga masker ini lebih tinggi dibanding masker bedah karena kinerjanya yang sangat efektif dalam mencegah penyebaran virus corona. Selain dapat menghalangi percikan air liur, masker ini juga mampu menghambat partikel

kecil di udara yang bisa membawa virus. Meski demikian, masker ini tidak disarankan untuk digunakan sehari-hari dan disarankan hanya untuk petugas medis yang terlibat secara langsung dengan pasien Covid-19.

2.2 Citra Digital

Dalam sebuah citra digital terdiri dari kumpulan piksel-piksel, setiap piksel dalam citra memiliki nilai intensitas yang mempresentasikan tingkat kecerahan atau warna pada lokasi piksel tersebut dan memiliki 2 dimensi $f(x,y)$ yang dimana x dan y merupakan tingkat kecerahan suatu citra pada suatu titik (Gonzalez, 2008)

Dalam setiap titik citra dapat di tuliskan sebagai berikut:

$$0 < f(x,y) < \infty$$

Citra digital juga bisa di representasikan dalam sebuah *matrix* $M \times N$, yang dimana M berupa jumlah baris dan N merupakan jumlah kolom citra. Nilai yang dapat dalam matriks disebut sebagai piksel yang mempresentasikan cahaya atau warna citra tersebut.

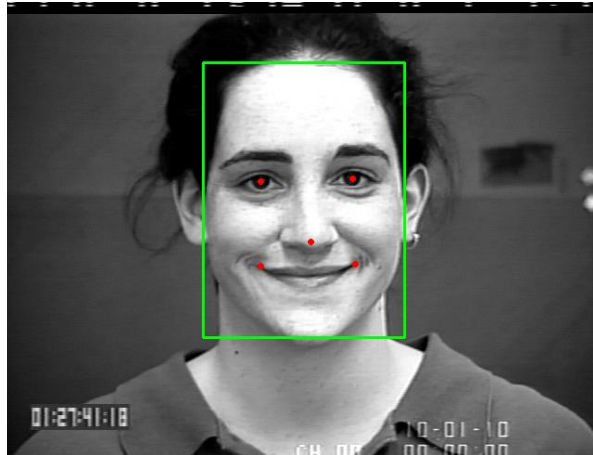
2.3 Multi-Task Cascaded Convolutional Networks

MTCNN (Multi-Task Cascaded Convolutional Networks) adalah sebuah model *deep learning* untuk pengenalan wajah dan deteksi *landmark* pada wajah yang dikembangkan oleh Zhang et al. dalam jurnal "*Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks*".

Model *MTCNN* terdiri dari tiga tahap yang saling terkait: tahap deteksi wajah (*face detection*), tahap pencarian *landmark* (*landmark localization*), dan tahap penapisan wajah palsu (*face false positive filtering*). Setiap tahap menggunakan jaringan konvolusi yang terpisah untuk menghasilkan output yang semakin tepat dalam mengenali wajah.

Tahap pertama (deteksi wajah) menggunakan jaringan konvolusi untuk mencari kotak pembatas (*bounding box*) pada wajah dalam Gambar. Tahap kedua (pencarian *landmark*) menggunakan jaringan konvolusi untuk menentukan posisi *landmark* pada wajah, seperti mata, hidung, dan mulut. Tahap ketiga (penapisan wajah palsu) kemudian menggunakan sebuah jaringan konvolusi lagi untuk memfilter wajah yang tidak relevan atau palsu.

Dengan menggabungkan tiga tahap ini, *MTCNN* dapat mengenali wajah dengan akurasi yang tinggi dan dapat mengatasi beberapa masalah seperti variasi posisi dan ukuran wajah dalam Gambar. Model ini dapat digunakan dalam berbagai aplikasi seperti pengenalan wajah, deteksi emosi, pengenalan gender, dan masih banyak lagi.



Gambar 2. Deteksi dengan *MTCNN*

2.4 Sistem Pengenalan Wajah

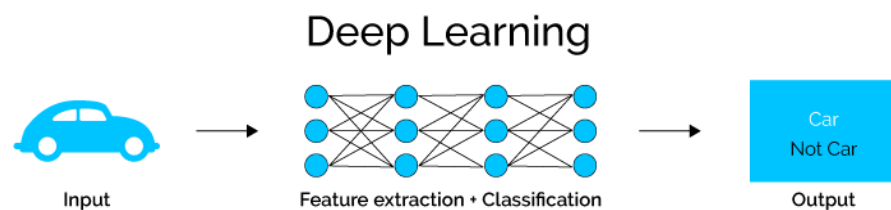
Pengenalan wajah adalah teknologi komputer untuk mengenali dan menentukan lokasi wajah, ukuran wajah, deteksi fitur wajah, dan pengabaian citra latar, selanjutnya dilakukan mengidentifikasi citra wajah. Pengenalan wajah ini dibagi menjadi dua yaitu bagian dikenali dan tidak dikenali, dalam proses pengenalan wajah, citra dapat diambil dari jarak jauh tanpa menyentuh orang yang sedang diidentifikasi. (Bridga. 2016)

Ada banyak model yang dapat digunakan untuk pengenalan wajah terutama dalam penelitian ini akan menggunakan *Multi-Task Cascaded Convolutional Networks (MTCNN)*.

2.5 Deep Learning

Mengatakan *Deep Learning* adalah salah satu metode pembelajaran mesin yang menggunakan jaringan saraf dengan banyak layer untuk memproses data dan menghasilkan output yang akurat. Jaringan saraf ini terdiri dari banyak neuron yang saling terhubung dan bekerja secara

bersama-sama untuk mengekstraksi fitur dari data yang kompleks. Teknik ini telah digunakan dalam berbagai aplikasi, seperti pengenalan wajah, pengenalan suara, pemrosesan bahasa alami, dan lain-lain (Hao et al., 2016).



Gambar 3. Struktur arsitektur *deep learning*

Menurut (Neapolitan & Jiang, 2018) Dalam konteks pembangunan model, ada dua istilah yang krusial, yaitu pelatihan dan pengujian. Pelatihan atau yang dikenal dengan istilah *training*, merupakan proses pembentukan model, sedangkan pengujian atau *testing*, adalah proses untuk mengevaluasi kinerja model yang telah dibangun. Data set yang digunakan dalam pelatihan biasanya terdiri dari kumpulan data, yang dapat berupa sampel data dalam bidang statistika atau citra. Secara umum, data set dapat dikelompokkan ke dalam tiga jenis yang saling tidak beririsan antara lain sebagai berikut:

a. *Training Set*

Training set merupakan perkumpulan banyaknya data yang akan digunakan pelatihan dalam membangun sebuah model.

b. *Development set*

Development set atau biasa disebut dengan *validation set* merupakan perkumpulan yang digunakan untuk melakukan *tuning* pada mesin yang sudah dilatih menggunakan *training set*, maka akan dilakukan validasi menggunakan *Validation set*. Sehingga akan memudahkan dalam proses generalisasi dan model mampu mengenali pola secara generik.

c. *Testing set*

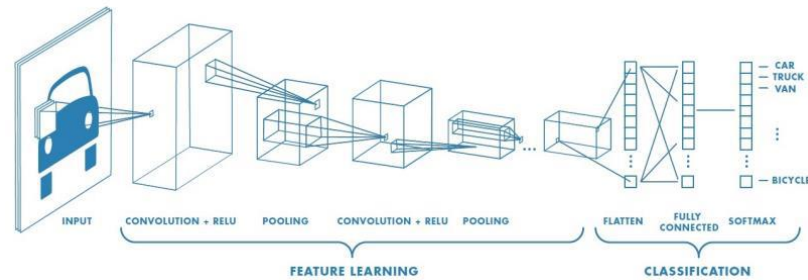
Testing set merupakan kumpulan data yang digunakan untuk menguji model setelah proses pelatihan selesai seberapa akurat pada data yang pernah dilihat sebelumnya.

Pembagian *dataset* dapat bervariasi tergantung pada situasi yang ada. Biasanya, pembagian *dataset* (*training: validation: testing*) adalah (80%: 10%: 10%) atau (90%: 5%: 5%). Namun, jika *dataset* yang digunakan kecil, maka *validation set* mungkin tidak diperlukan sehingga *dataset* hanya dibagi menjadi *training set* dan *testing set* saja. Jenis pembagian ini disebut sebagai *closed testing*. Rasio yang dapat digunakan untuk pembagian *dataset* (*training, testing*) meliputi (90%: 10%), (80%: 20%), (75%: 25%), atau bahkan (50%: 50%).

2.6 Convolutional Neural Network (CNN)

(Putra, 2016) *Convolutional Neural Network* (CNN) adalah pengembangan dari *Multilayer Perceptron* (MLP) yang didesain untuk mengelola data dua dimensi, seperti citra teks, potongan suara dan sebagainya yang berupa data. CNN dapat mengenali dan mengambil gambar

input, menetapkan bobot dan bias yang dapat dipelajari untuk berbagai beberapa objek.

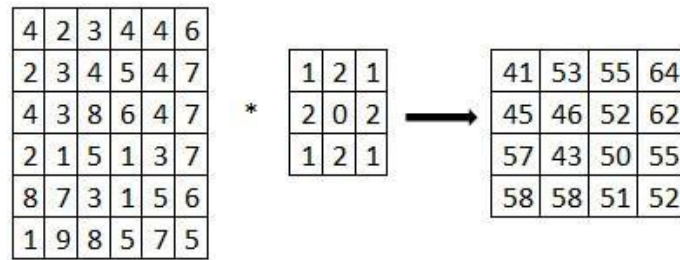


Gambar 4. Ilustrasi Arsitektur *Convolutional Neural Network*

2.6.1 Convolutional Layer

(Zufar et al., 2016) mengatakan bahwa *Convolutional Layer* merupakan lapisan yang pertama kali menerima masukan citra yang langsung pada arsitektur. Operasi pada lapisan ini sama dengan operasi konvolusi yaitu melakukan operasi kombinasi *linier filter* terhadap daerah lokal. Setiap filter yang akan mengalami operasi “dot” dari data awal dan nilai filter dalam tahapan awal filter setiap *citra* yang di *input* dengan matrix 6x6 lalu dimasukkan akan dikali dengan setiap kernel 3x3 untuk proses konvolusi. Citra input akan dibagi menjadi matriks 3x3, dan setiap matriks tersebut akan dikalikan dengan kernel yang tersedia. Hasil perkalian dari setiap matriks 3x3 dan kernel akan dijumlahkan, dan output dari penjumlahan tersebut akan ditempatkan di *cell* pada citra hasil baru.

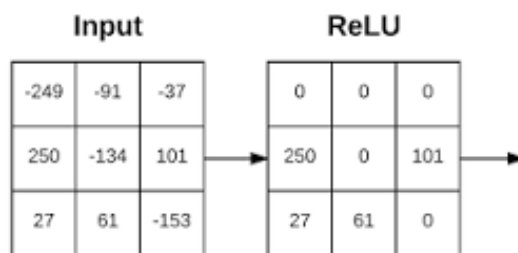
Proses ini ditunjukkan dalam Gambar 1.3, yang merupakan contoh dari layer konvolusi.



Gambar 5. Proses *convolutional layer*

2.6.2 Rectified Linear Unit

Aktivasi ReLu (*Rectified Linear Unit*) adalah fungsi aktivasi yang paling umum digunakan pada layer CNN dan jaringan saraf dalam pemrosesan Gambar. Fungsi $f(x)=\max(0,x)$ ini menghasilkan nilai nol untuk semua nilai input yang negatif, dan menghasilkan nilai input itu sendiri untuk nilai input yang positif. Dengan kata lain, jika nilai input kurang dari atau sama dengan nol, maka nilai output akan nol, sedangkan jika nilai input lebih besar dari nol, maka nilai output akan sama dengan nilai input (Agarap, 2018).

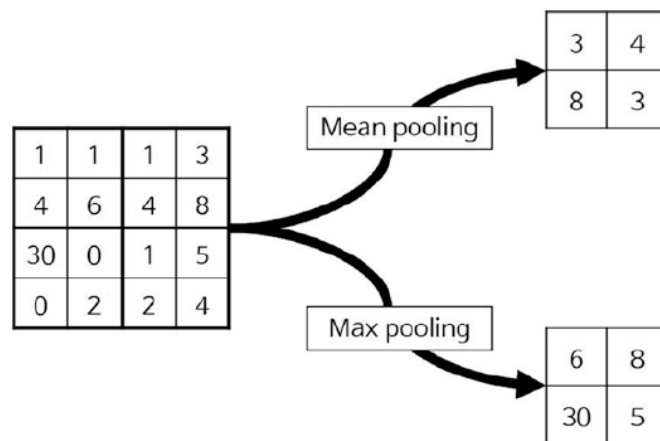


Gambar 6. Contoh Penerapan Aktivasi *ReLU*

2.6.3 Pooling Layer

Pooling Layer merupakan salah satu tahapan proses dari *convolutional neural network* setelah dari tahap aktivasi biasa disebut

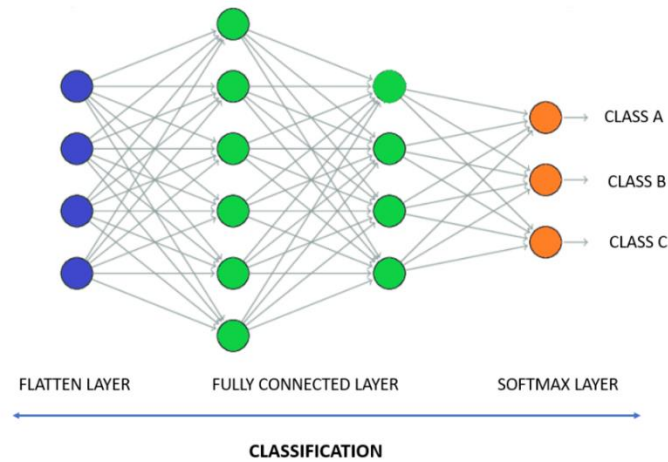
subsampling atau *downsampling* yang bertujuan untuk mengurangi dimensi dari *feature map* tanpa mengurangi informasi penting di dalamnya. Langkah awal pada layer *pooling* adalah menentukan ukuran *downsampling* yang akan diterapkan pada *feature map*. Setelah itu, dilakukan proses *pooling* pada *feature map* tersebut. Tujuan dari layer *pooling* adalah mengurangi dimensi pada *feature map* sehingga dapat mempercepat proses komputasi karena jumlah parameter yang perlu diperbarui menjadi lebih sedikit (Gholamalinezhad & Khosravi, 2020). Proses *pooling* pada *feature map* dapat dijelaskan melalui Gambar 7 berikut.



Gambar 7. Matriks *feature map* 4x4 dengan proses *pooling* 2x2

2.6.4 Fully Connected Layer

Layer ini adalah layer akhir yang menerima hasil dari layer *pooling* sebagai input dan mengubah data dalam bentuk matriks x-dimensi menjadi *matriks linear* atau matriks 1 dimensi agar klasifikasi menjadi lebih mudah untuk dilakukan (Basha et al., 2019).

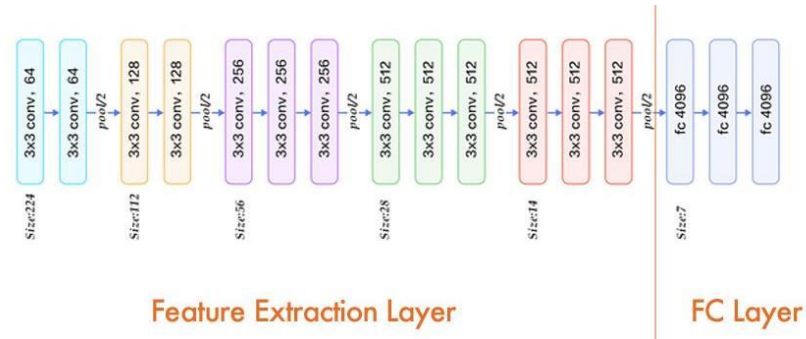


Gambar 8. *Fully Connected Layer*

2.6.5 Arsitektur Jaringan VGG16Net

Arsitektur Jaringan *VGG16Net* merupakan salah satu arsitektur jaringan *convolutional neural network* (CNN) yang dikembangkan oleh *Geometry Group* (VGG) dari *University of Oxford*. Arsitektur ini terdiri dari 16 *layer convolutional* dan layer-layer *fully connected*, sehingga memungkinkan *VGG16Net* untuk mengenali pola-pola kompleks dalam citra.

VGG16Net menggunakan konvolusi 3x3 dengan *stride* 1 dan *padding* sama untuk semua *layer convolutional*-nya, dan *max pooling* 2x2 dengan *stride* 2 setelah setiap dua *layer convolutional*. *VGG16Net* juga menggunakan *dropout* dan *regularization* untuk menghindari *overfitting*. *VGG16Net* memiliki kinerja yang sangat baik dalam pengenalan objek pada data set ImageNet, dengan menghasilkan tingkat akurasi yang lebih tinggi dibandingkan dengan arsitektur jaringan CNN lainnya.



Gambar 9. Arsitektur jaringan *VGG16Net*

Pada Gambar 9 *VGG16Net* yang disusulkan oleh Simoyan dan Zisserman dari *Universitas Oxford* dalam tulisan “*Very Deep Convolutional Networks for Large-Scale Image Recognition*” Arsitektur model VGG16 berhasil mencapai tingkat akurasi pengujian sebesar 92,7% pada data set ImageNet yang terdiri dari lebih dari 14 juta Gambar dengan 1000 kelas. Model VGG16 termasuk salah satu model yang terkenal dan berhasil diajukan ke dalam kompetisi ILSVRC-2014.(Simonyan & Zisserman, 2014)

2.7 Adaptive Moment Estimation

Adam adalah sebuah algoritma optimasi gradien stokastik yang sangat populer di bidang *Deep learning*. *Adam* adalah singkatan dari “*Adaptive Moment Estimation*”. *Adam* sering digunakan untuk memperbaharui parameter model dalam pelatihan jaringan syarat yang beberapa kasus dapat memberikan konvergensi yang lebih cepat dan akurat dibanding algoritme optimisasi gradien stokastik lainnya (Kingma & Ba, 2014)

2.8 Root Mean Square Propagation

Algoritma *RMSprop* merupakan salah satu terobosan dalam pengembangan metode pelatihan untuk Jaringan Saraf Tiruan (JST). Disebut sebagai metode tingkat pembelajaran adaptif, *RMSprop* didesain dengan inspirasi dari konsep penurunan gradien dan teknik *Resilient Back Propagation* (RProp). Metode ini ditempatkan dalam konteks metode tingkat pembelajaran adaptif yang kian populer akhir-akhir ini, mengingat kemajuan ini merupakan hasil pengembangan dari beberapa pendekatan lain dalam dunia JST.

Integrasi konsep rata-rata pada *mini-batch*, efisiensi, dan gradien pada *mini-batch* berurutan adalah karakteristik utama dari pendekatan ini. Tujuannya adalah untuk mencapai tingkat konvergensi yang lebih cepat daripada pendekatan optimasi dasar, meskipun masih lebih rendah dibandingkan dengan pendekatan optimasi lanjutan seperti algoritma *Adam*.

Salah satu penerapan praktis dari *RMSprop* adalah dalam teknologi stokastik untuk penurunan *gradien mini-batch*. *RMSprop* dirancang khusus untuk mengatasi kendala yang muncul pada algoritma *Resilient Back Propagation* (RProp), yang kurang efektif ketika diterapkan pada *mini-batch*. Dengan menggunakan tanda gradien dari algoritma *RProp*, *RMSprop* dapat meningkatkan *efisiensi mini-batch* dan rata-rata di atas *mini-batch*, yang pada gilirannya memungkinkan penggabungan gradien yang lebih

akurat. (*RMSprop* - *Cornell University Computational Optimization Open Textbook - Optimization Wiki*, n.d.)

2.9 Stochastic Gradient Descent

Algoritma *Stochastic Gradient Descent* (*SGD*) merupakan metode iteratif yang umumnya digunakan dalam pembelajaran mesin dengan tujuan mengoptimalkan penurunan gradien. Pendekatan ini melibatkan inisialisasi bobot oleh pengguna dan proses pembaharuan vektor bobot menggunakan satu titik data pada setiap langkah iteratif. Penurunan gradien terjadi secara bertahap seiring penyelesaian perhitungan kesalahan, yang bertujuan untuk meningkatkan konvergensi (Needell et al., 2016).

Metode ini berusaha mencari penurunan gradien paling tajam untuk mengurangi jumlah iterasi dan waktu yang dibutuhkan dalam mencari titik data, khususnya pada jumlah data yang besar.

Penurunan gradien stokastik diimplementasikan dalam jaringan saraf untuk mengurangi waktu komputasi sambil meningkatkan kompleksitas dan kinerja, terutama pada masalah dengan skala besar. Algoritma *SGD* merupakan modifikasi dari algoritma *batch gradient descent*, di mana gradien dihitung hanya untuk satu contoh pelatihan pada setiap iterasi (Nugroho et al., n.d.).

Pendekatan ini memiliki berbagai aplikasi dalam bidang pembelajaran mesin, geofisika, *least mean squares* (LMS), dan bidang lainnya.

Dengan demikian, *SGD* menjadi relevan dalam konteks pembelajaran mesin dan aplikasinya dapat ditemui dalam berbagai domain, memberikan kontribusi pada efisiensi komputasi dan peningkatan kinerja terutama dalam menangani masalah skala besar..

2.10 Keras

Keras merupakan *library* jaringan syaraf tiruan yang sangat populer dan tingkat tinggi yang dituliskan dalam bahasa *python* untuk membangun dan melatih sebuah model jaringan syaraf (*Neural Network*). *Keras* menyediakan antarmuka yang mudah digunakan untuk membangun model jaringan saraf yang kompleks dengan berbagai jenis lapisan (*layer*), seperti lapisan konvolusi, rekursif, dan *pooling*. *Keras* juga dapat bekerja dengan berbagai *backend* jaringan saraf seperti *TensorFlow*, *Theano*, dan *Microsoft Cognitive Toolkit* (CNTK). (Chollet, F., et al. 2015)

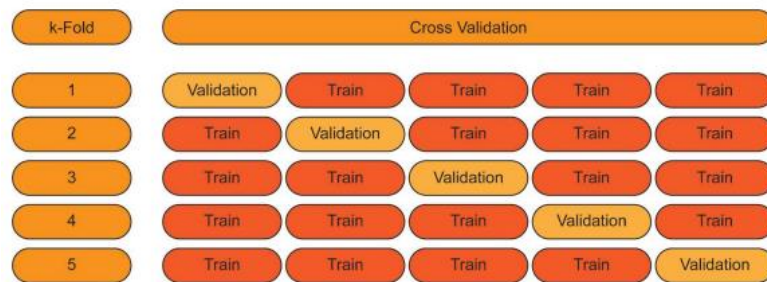
2.11 TensorFlow

TensorFlow merupakan *library open-source* untuk melakukan komputasi numerik dalam data yang berbentuk tensor. *Library* ini sangat banyak digunakan dalam pembelajaran mesin dan kecerdasan buatan dalam hal membangun sebuah model jaringan syaraf yang kompleks. *Tensorflow* menawarkan berbagai variasi *toolkit* yang memungkinkan untuk membangun model pada tingkat abstraksi yang kita inginkan (Nurfita & Ariyanto, 2018)

2.12 K-Fold Cross Validation

Validasi silang *K-Fold* (*K-Fold Cross Validation*) merupakan suatu

teknik esensial dalam *machine learning* yang bertujuan untuk menjamin bahwa model yang dikembangkan mampu beroperasi dengan optimal pada data yang belum pernah dikenal sebelumnya. Pendekatan ini dilakukan dengan membagi *dataset* menjadi 'K' bagian atau '*fold*' yang memiliki ukuran yang serupa, lalu menggunakan 'K-1' *fold* untuk proses pelatihan model dan *fold* yang tersisa untuk menguji performa model. Proses ini diulang sebanyak 'K' kali, memastikan bahwa setiap *fold* memiliki kesempatan sebagai data pengujian dalam ilustrasi seperti Gambar 1 (Widodo et al., 2022)



Gambar 10. Ilustrasi Proses 5-fold Cross Validation

2.13 Confusion Matrix

Pengukuran performa *confusion matrix* adalah salah satu teknik evaluasi model klasifikasi yang digunakan untuk mengukur sejauh mana kinerja model dalam memprediksi label kelas dari suatu data. *Confusion matrix* atau disebut juga dengan *contingency table*, adalah tabel matriks yang menunjukkan jumlah prediksi yang benar atau salah oleh model (Yun, 2021)

Dalam menggunakan kinerja Confusion Matrix terdapat 4 istilah sebagai hasil proses klasifikasi, istilah tersebut adalah :

- a. *True Positive (tp)*
- b. *False Positive (fp)*
- c. *True Negative (tn)*
- d. *False Negative (fn)*

Pada jenis klasifikasi yang hanya memiliki 2 keluaran kelas seperti *WithMask* dan *WithoutMask* dapat disajikan pada tabel berikut:

Tabel 1. *Confusion Matrix*

kelas	<i>withmask</i>	<i>withoutmask</i>
<i>withmask</i>	<i>true positive (tp)</i>	<i>false positive (fp)</i>
<i>withoutmask</i>	<i>false negative (fn)</i>	<i>true negative (tn)</i>

Sehingga untuk mencari akurasi dalam suatu model berdasarkan tabel *confusion matrix* adalah sebagai berikut:

$$Akurasi = \frac{TP + TN}{TP + FP + TN + FN} \times 100\%$$

Akurasi *confusion matrix* juga dapat digunakan untuk menghitung presisi dan secara matematis dapat dirumuskan sebagai berikut:

$$Presisi = \frac{TP}{TP + FP}$$

Recall, secara definisi, mengukur seberapa banyak *True Positive* (TP) yang berhasil diidentifikasi oleh model, dibandingkan dengan total

jumlah data yang sebenarnya positif. Secara matematis, *recall* dapat dituliskan dengan rumus sebagai berikut:

$$Recall = \frac{TP}{TP + FN}$$

F1-Score adalah rata-rata harmonis antara presisi dan *recall*. Secara matematis, *F1-Score* dapat diungkapkan dengan rumus sebagai berikut:

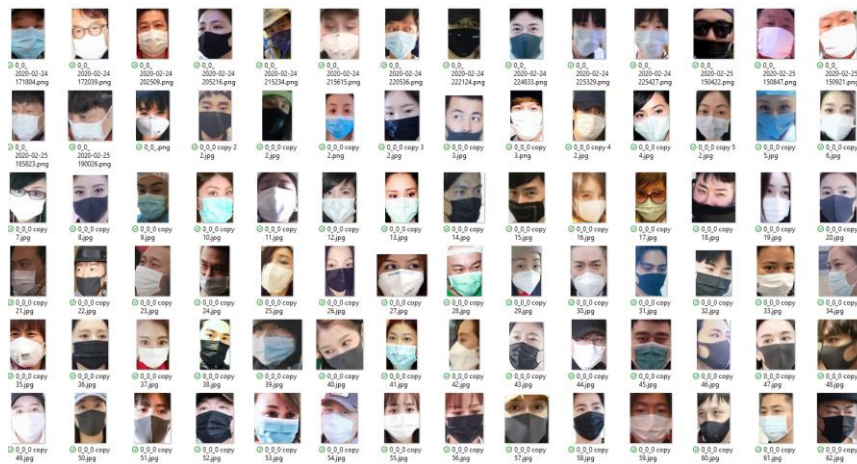
$$\frac{1}{F1} = \frac{1}{2} \left(\frac{1}{presisi} + \frac{1}{recall} \right)$$

BAB III

METODE PENELITIAN

3.1 Deskripsi Data

Data yang digunakan dalam penelitian ini diperoleh dari pengumpulan data citra orang yang menggunakan masker dan tidak menggunakan masker, yang merupakan data citra digunakan untuk melatih *machine learning*. Data ini memiliki jumlah 1.600 data citra yang terdiri dari 800 tanpa masker dan 800 menggunakan masker. Gambar 11 dan 12 merupakan isi dari *dataset* yang digunakan.



Gambar 11. Data penggunaan masker



Gambar 12. Data penggunaan tanpa masker

3.2 Kebutuhan Perangkat *Hardware* dan *Software*

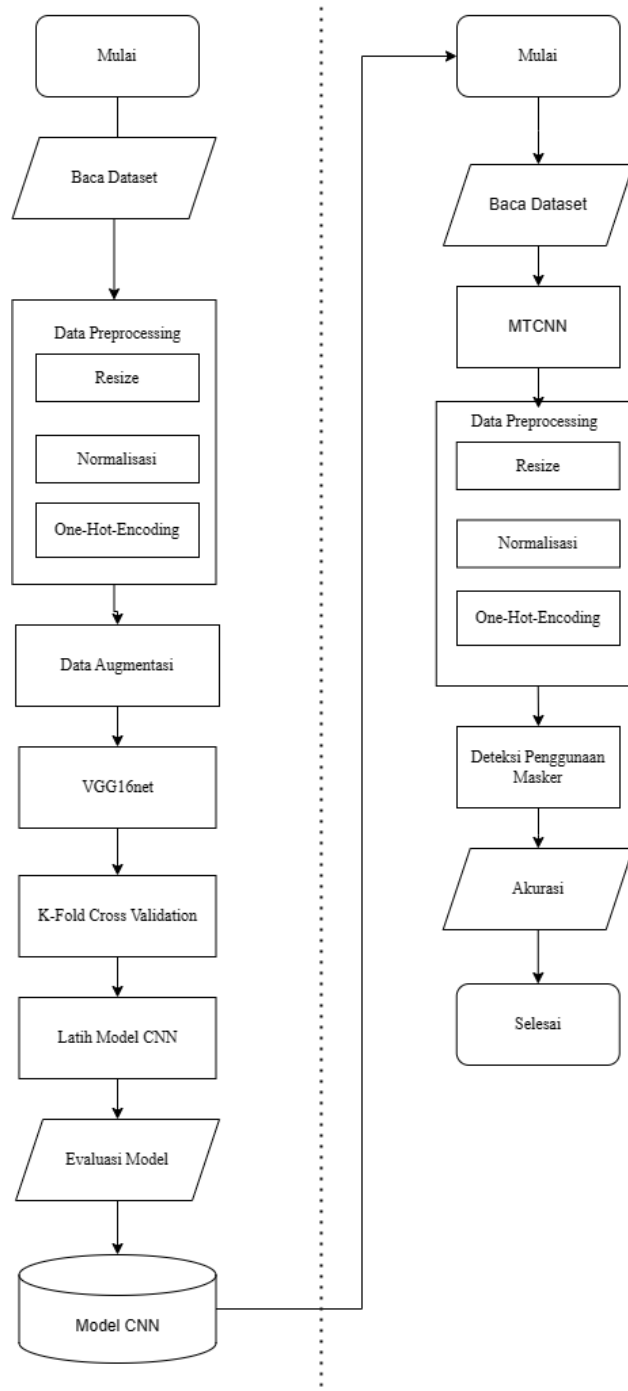
1) Spesifikasi Hardware

- a. Processor AMD Ryzen 5 3550H 4 Core 8 Threads
- b. RAM 16 GB DDR4 Dual-Channel 2400 Mhz
- c. Nvidia GTX 1050 3GB

2) Spesifikasi Software

- a. Sistem Operasi Windows 10 Home 64—bit
- b. Google Collab dan Visual Studio Code

3.3 Perancangan Sistem



Gambar 13. Diagram Perancangan Sistem

Berdasarkan Gambar 13 diagram perancangan sistem dibagi menjadi 6 tahap yaitu *Preprocessing*, *Data Augmentation*, arsitektur

VGG16net, *K-Fold Cross Validation*, Latih Model CNN, dan evaluasi model CNN. Berikut penjelasan masing-masing tahapan tersebut.

3.3.1 Preprocessing

Pada proses *Preprocessing* ini bertujuan untuk mempersiapkan data sebelum dilakukan *Data Augmentation*. Tahap ini bertujuan untuk melakukan normalisasi, transformasi, dan pengubahan ukuran citra sebagai untuk Gambar agar sesuai dengan syarat yang diperlukan oleh arsitektur *VGG16Net* sehingga lebih mudah untuk diproses oleh model *deep learning*. Berikut adalah tahapan *preprocessing* yang akan dilakukan.

3.3.1.1 Rezise

Rezise merupakan proses untuk mengubah ukuran citra menjadi dimensi tertentu yang diperlukan oleh arsitektur jaringan *VGG16Net*. Pada penelitian ini ukuran citra akan di ubah ukurannya menjadi 224 x 224 piksel. Hal ini dilakukan dengan tujuan terdapat beberapa data citra yang ukuran pikselnya berbeda-beda sehingga kemungkinan bisa menimbulkan hasil akurasi berkurang atau rendah.

3.3.1.2 Normalisasi

Tahap normalisasi ini dilakukan untuk memperoleh rentang piksel yang seragam dan mempermudah proses pelatihan untuk *VGG16Net*. Normalisasi dilakukan dengan membagi setiap piksel dengan standar deviasi dari setiap saluran warna. Hal ini bertujuan menghasilkan piksel dengan rata-rata nol dan standar deviasi 1. Hal ini untuk mengontrol variasi piksel dan mempermudah konvergen selama pelatihan

Dalam tahap *Preprocessing*, penelitian ini menggunakan *framework* yang sudah di sediakan oleh *framework* Keras untuk melakukan proses *preprocessing* dalam arsitektur *VGG16Net*. Salah satu fungsi yang digunakan yaitu “*preprocess_input()*” yang tersedia dalam modul “*keras.application.vgg16*”

3.3.1.3 One-Hot-Encoding

Dalam konteks deteksi penggunaan masker dan bukan masker, *one-hot encoding* digunakan sebagai teknik untuk mengubah label kelas menjadi representasi biner yang lebih mudah diolah oleh model. Proses ini dimulai dengan penggunaan *LabelBinarizer*, yang mengonversi label kelas seperti “*mask*” dan “*no_mask*” menjadi vektor biner, di mana elemen vektor yang sesuai dengan kelas tertentu akan memiliki nilai 1, sedangkan elemen lainnya memiliki nilai 0. Misalnya, label “*mask*” akan direpresentasikan sebagai [1, 0], dan label “*no_mask*” akan direpresentasikan sebagai [0, 1].

3.3.2 Data Augmentation

proses *data augmentation* untuk membuat variasi baru dalam *dataset*. Terdapat beberapa parameter variasi data augmentasi yang meliputi rotasi, pergeseran, *cropping*, *zooming*, dan beberapa variasi lainnya yang bertujuan membantu meningkatkan performa model pada *deep learning*.

3.3.3 VGG16net

Memuat VGG16net yang merupakan implementasi dari *transfer learning* menggunakan arsitektur VGG16 dalam *TensorFlow*. Pertama, model dasar VGG16 diinisialisasi dengan menggunakan bobot *pre-trained* dari *dataset imagenet*, dan lapisan teratas (*fully connected layers*) diabaikan (*include_top=False*). Selain itu, *input layer model* diatur sesuai dengan dimensi gambar yang diharapkan (224x224 piksel dengan 3 saluran warna). Setelah itu, lapisan-lapisan VGG16 dinonaktifkan agar tidak dapat di-train ulang (*baseModel.trainable = False*). Selanjutnya, ringkasan dari model tersebut dicetak untuk memberikan gambaran mengenai arsitektur dan jumlah parameter yang terlibat. Dengan langkah ini, model VGG16 siap digunakan sebagai ekstraktor fitur dalam tugas *transfer learning* tanpa melibatkan pelatihan ulang pada lapisan-lapisan internalnya.

3.3.4 K-Fold Cross Validation

K-Fold Cross Validation digunakan untuk menguji dan mengevaluasi performa model secara lebih konsisten dan reliabel. *K-Fold Cross Validation* membagi *dataset* menjadi *k subset* atau "*fold*" yang saling bergantian sebagai data pelatihan dan pengujian. Dalam setiap iterasi *loop*, model dikompilasi dan dilatih menggunakan data pelatihan dari satu *fold* dan diuji pada *fold* yang berbeda. Ini membantu mengurangi risiko *overfitting* atau *underfitting* yang mungkin terjadi pada satu pembagian *dataset* tertentu. Selama pelatihan, beberapa metrik evaluasi seperti akurasi, presisi, recall, dan F1-score dihitung untuk setiap *fold*.

Seluruh proses *K-Fold Cross Validation* memungkinkan kita untuk mendapatkan gambaran yang lebih akurat tentang sejauh mana model dapat melakukan generalisasi pada data yang berbeda, serta memungkinkan pengukuran performa yang lebih stabil dan representatif.

Untuk memahami lebih lanjut tentang cara kerja validasi silang *K-Fold*, dalam penelitian ini dapat mengilustrasikannya pada beberapa contoh:

Dalam *3-Fold Cross Validation*, *dataset* dengan 1600 data dibagi menjadi 3 bagian yang setara, dengan masing-masing bagian berisi sekitar 533 data. Pada setiap iterasi, dua bagian (sekitar 1067 data) digunakan untuk proses 'belajar', sementara satu bagian (sekitar 533 data) digunakan untuk menguji sejauh mana model telah memahami data.

Pada *5-Fold Cross Validation*, 1600 data dibagi menjadi 5 bagian yang serupa, dengan masing-masing bagian berisi sekitar 320 data. Dalam setiap putaran, empat bagian (1280 data) diambil sebagai data 'belajar', dan satu bagian (320 data) digunakan sebagai data 'menguji' untuk mengevaluasi kemampuan model.

Sementara pada *7-Fold Cross Validation*, *dataset* dengan 1600 data dibagi menjadi 7 bagian yang setara, dengan masing-masing bagian berisi sekitar 229 data. Pada setiap iterasi, enam bagian (sekitar 1371 data) digunakan untuk *training*, sementara satu bagian (sekitar 229 data) digunakan untuk *validation* kemampuan model yang telah dihasilkan. Pendekatan ini memberikan Gambaran menyeluruh tentang sejauh mana

model dapat generalisasi terhadap data yang belum pernah dilihat sebelumnya.

3.3.5 Latih Model CNN

Dalam tahap ini akan dilakukan pelatihan dengan *data training* yang telah diaugmentasi. Pada penelitian ini dilakukan dengan menggunakan metode *convolutional neural network* (CNN).

3.3.6 Evaluasi Model

Setelah tahap latih model CNN, akan dilakukan evaluasi performa dalam model pada data testing. Dalam hal ini model akan diukur menggunakan *confusion matrix* yang meliputi *akurasi*, *presisi*, *recall* dan *F1-score*.

3.3.7 Model CNN

Setelah model dinyatakan memenuhi kriteria performa *confusion matrix*, model dapat disimpan untuk digunakan pada tahap untuk mendeteksi masker pada citra baru.

3.4 Skenario Pengujian

Dalam penelitian ini dari jumlah *dataset* akan dibagi menjadi dua percobaan dari *dataset* awal berjumlah 1.600 (800 tanpa masker dan 800 menggunakan masker)

3.4.1 Skenario *K-Fold Cross Validation*

Skenario pengujian ini dilakukan seleksi *k-fold*, dalam pengujian ini akan dilakukan percobaan menggunakan tiga jenis *k-fold* yaitu *3-fold*, *5-fold*, dan *7-fold*, dari 3 percobaan tersebut akan dicari hasil akurasi yang optimal.

3.4.2 Skenario Fungsi Pelatihan CNN

Dalam Skenario ini fungsi pelatihan pada *Convolutional Neural Network* akan menggunakan percobaan 3 jenis fungsi aktivasi yaitu *Adaptive Moment Estimation*, *Stochastic Gradient Descent*, dan *Root Mean Square Propagation* untuk membuat sebuah model CNN yang dibuat panjang iterasi atau *epochs* sebesar 30, 50, dan 100 iterasi.

3.4.3 Skenario Uji Data Tunggal

Dalam Skenario uji data tunggal untuk menguji seberapa baik model yang sudah dilatih untuk mengetahui seberapa akurat model tersebut mendeteksi data citra yang baru.

BAB IV

HASIL DAN ANALISIS

4.1 Data

Data yang digunakan adalah data yang diperoleh dari dari laman <https://www.kaggle.com/datasets/omkargurav/face-mask-dataset>. Data tersebut memiliki dua kelas yaitu *with_mask* dan *without mask* untuk melatih *machine learning*.

4.1.1 Preprocessing

Dalam tahap ini peneliti melakukan tahap mengelola data citra agar data tersebut dimasukkan ke pelatihan mesin dalam *Tensorflow* untuk membuat model dari CNN.

4.1.2 Membuat Variabel Parameter

Pada tahap ini akan mendefinisikan variabel konstanta sebagai berikut :

Tabel 2. Parameter Skenario Pengujian

No.	Parameter	Value
1	<i>Epoch</i>	30,50,100
2	<i>Target Size</i>	224 x 224
3	<i>Batch Size</i>	32
4	<i>Learning Rate</i>	0,0001

INIT_LR adalah *learning rate* awal yang akan digunakan dalam algoritma pelatihan. *Learning rate* mengontrol seberapa besar langkah yang diambil saat mengoptimasi model selama pelatihan.

EPOCHS adalah jumlah *epoch* (iterasi) yang akan digunakan dalam pelatihan model. Setiap *epoch* mewakili satu kali siklus melalui seluruh *dataset* pelatihan.

BS (Batch Size) adalah ukuran *batch* yang akan digunakan selama pelatihan. Pelatihan *deep learning* sering dilakukan dalam *batch*, di mana model diperbarui setelah melihat sejumlah data.

4.1.3 Implementasi Memuat Data Citra

Proses selanjutnya yaitu memuat seluruh data citra yang akan dilakukan untuk pelatihan model yang diinginkan

```
# Mengambil gambar dari dataset directory, kemudian inisialisasi data dan class gambar
print("Menginput gambar...")
imagePaths = list(paths.list_images("/content/Skripsi/dataset")) # Dataset Penelitian
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 14. *Code Blok* untuk memuat data citra.

Pada Gambar 14 seluruh blok kode ini bertanggung jawab untuk memproses Gambar dari direktori "*dataset*", mengambil label kelas dari jalur Gambar, melakukan *preprocessing* Gambar sesuai dengan model yang akan digunakan, dan mengubah label menjadi format yang sesuai untuk

pelatihan jaringan saraf. Data Gambar yang telah diolah dan label yang telah di-*one-hot-encoded* siap digunakan untuk melatih model.

4.1.4 Data Augmentation

```
3 aug = ImageDataGenerator(
4     rotation_range=20,
5     zoom_range=0.15,
6     width_shift_range=0.2,
7     height_shift_range=0.2,
8     shear_range=0.15,
9     horizontal_flip=True,
10    fill_mode="nearest")
```

Gambar 15. Implementasi *Data Augmentation*

Pada Gambar 16 menggunakan objek *aug* ini, dapat menghasilkan variasi dari data pelatihan saat model dilatih, meningkatkan kemampuan umum model dalam mengenali objek yang berbeda-beda dalam berbagai kondisi.

4.2 Membuat Model Jaringan CNN Dengan arsitektur *VGG16Net*

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

Python

Gambar 16. Membangun *base model VGG16Net*

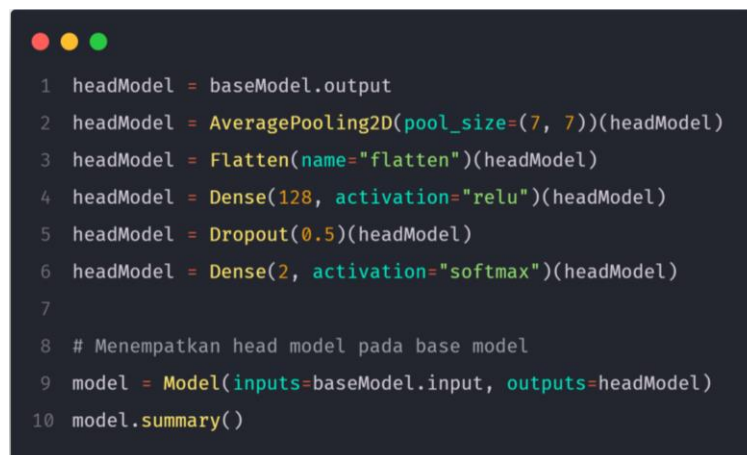
Pada Gambar 16 bertujuan untuk membangun model dasar (*base model*) dengan menggunakan arsitektur *VGG16* yang telah dilatih sebelumnya pada *dataset ImageNet*. *tf.keras.applications.VGG16* merupakan fungsi yang digunakan untuk memanggil model *VGG16* dari

TensorFlow. weights="imagenet" Ini mengindikasikan bahwa kita ingin menggunakan bobot yang telah dilatih pada *dataset ImageNet*. *include_top=False* Ini menghilangkan lapisan *fully connected (top layers)* yang terhubung langsung dengan *output*, sehingga model dapat digunakan untuk ekstraksi fitur.

input_tensor=Input(shape=(224, 224, 3)) merupakan mendefinisikan input untuk model, yaitu Gambar dengan ukuran 224x224 piksel dan 3 channel warna (RGB).

4.3 Implementasi Tahap Pembuatan Model

Dalam tahap ini peneliti langsung membuat proses pembuatan model CNN untuk melakukan pelatihan



```

1 headModel = baseModel.output
2 headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
3 headModel = Flatten(name="flatten")(headModel)
4 headModel = Dense(128, activation="relu")(headModel)
5 headModel = Dropout(0.5)(headModel)
6 headModel = Dense(2, activation="softmax")(headModel)
7
8 # Menempatkan head model pada base model
9 model = Model(inputs=baseModel.input, outputs=headModel)
10 model.summary()

```

Gambar 17. *Code Blok* Pembuatan Model

Pada Gambar 17 *Code Blok* tersebut digunakan untuk membangun model *deep learning* dengan dua lapisan *output*. Model ini terdiri dari dua bagian utama, yaitu:

- 1) *Base model*, yang merupakan model yang sudah ada sebelumnya. Dalam kasus ini, base model adalah model VGG16, yang merupakan model *convolutional neural network* yang telah terbukti efektif untuk berbagai tugas pengenalan objek.
- 2) *Head model*, yang merupakan model tambahan yang ditambahkan di atas base model untuk menghasilkan output akhir. Head model ini terdiri dari beberapa lapisan *neural network*, yang berfungsi untuk menganalisis fitur-fitur yang dihasilkan oleh *base model* dan menghasilkan prediksi kelas.

Kode tersebut pertama-tama mengambil output dari base model, yaitu lapisan pool5. Kemudian, output tersebut diproses oleh beberapa lapisan neural network pada head model, yaitu:

- 1) *AveragePooling2D*, yang berfungsi untuk mengurangi ukuran output.
- 2) *Flatten*, yang berfungsi untuk mengubah output menjadi vektor.
- 3) *Dense*, yang berfungsi untuk menerapkan fungsi aktivasi relu pada vektor tersebut.
- 4) *Dropout*, yang berfungsi untuk mencegah *overfitting*.
- 5) *Dense*, yang berfungsi untuk menerapkan fungsi aktivasi *softmax* pada vektor tersebut.

Fungsi aktivasi *relu* berfungsi untuk memotong nilai-nilai negatif dan menjaga nilai-nilai positif. Fungsi aktivasi *softmax* berfungsi untuk mendistribusikan probabilitas pada kelas-kelas yang berbeda.

Pada akhir kode, model yang sudah dibangun tersebut diringkas menggunakan fungsi “*summary()*”. Fungsi ini akan menampilkan informasi tentang model, seperti jumlah parameter, ukuran model, dan arsitektur model.

4.4 Implementasi *K-Fold Cross Validation*

```
from sklearn.model_selection import KFold
kf = KFold(n_splits=n_fold, shuffle=True)
```

Gambar 18. Inisialisasi *K-Fold Cross Validation*

Pada Gambar 19 akan implementasi *K-Fold Cross Validation* yang dimana $kf = KFold(n_splits=n_fold, shuffle=True)$ Membuat objek *K-Fold* dengan jumlah lipatan (*folds*) sebanyak n_fold dan mengaktifkan pengacakan data.

4.5 Implementasi Tahap Pelatihan Model

```
# Perulangan pada seluruh base model
fold = 1
for train_index, test_index in kf.split(data):
    print("Mengompilasi model...")
    print("fold", fold)

    train_data, train_labels = data[train_index], labels[train_index]
    test_data, test_labels = data[test_index], labels[test_index]

    opt = tf.keras.optimizers.legacy.RMSprop(learning_rate=INIT_LR, decay=INIT_LR / EPOCHS)

    model.compile(loss="binary_crossentropy", optimizer=opt, metrics=["accuracy"])

    print("Training head model...")

    start_time = time.time() # Catat waktu awal

    history = model.fit(
        aug.flow(train_data, train_labels, batch_size=BS),
        steps_per_epoch=len(train_data) // BS,
        validation_data=(test_data, test_labels),
        validation_steps=len(test_data) // BS,
        epochs=EPOCHS)

    end_time = time.time() # Catat waktu akhir
    training_time = end_time - start_time
    fold_training_times.append(training_time)
```

Gambar 19. Tahap Pelatihan Model

Pelatihan model CNN dilakukan pada *subset* pelatihan menggunakan fungsi *model.fit()*. Waktu awal dan akhir pelatihan direkam

untuk setiap *fold*, dan hasil evaluasi model (seperti *accuracy*, *presisi*, *recall*, dan *F1-score*) juga dicatat melalui variabel *history*. Langkah-langkah ini diulang sebanyak *n_fold* kali, sesuai dengan konfigurasi *K-Fold* yang ditentukan sebelumnya.

Hasil waktu pelatihan untuk setiap *fold* dicatat dalam *fold_training_times*. Dengan demikian, tahap implementasi pelatihan model pada *K-Fold Cross Validation* ini memastikan bahwa model *CNN* dievaluasi secara menyeluruh pada berbagai *subset* data, memberikan gambaran yang lebih komprehensif tentang kinerja model dan mengurangi risiko *overfitting* atau *underfitting* pada suatu pembagian *dataset* tertentu.

4.6 Implementasi *Confusion Matrix*

```
from sklearn.metrics import confusion_matrix
# Hitung confusion matrix
conf_matrix = confusion_matrix(test_labels.argmax(axis=1), predictions.argmax(axis=1))

# Tampilkan confusion matrix
print(f"Confusion Matrix for Fold {fold}:\n{conf_matrix}")

# Atau, Anda juga dapat menggunakan library seaborn untuk tampilan yang lebih baik
import seaborn as sns
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=lb.classes_, yticklabels=lb.classes_)
plt.title(f'Confusion Matrix - Fold {fold}')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

Gambar 20. Implementasi *Confusion Matrix*

Pada Gambar 21 tersebut, *confusion matrix* dihitung untuk mengevaluasi hasil klasifikasi model pada setiap *fold* dalam *K-Fold Cross Validation*. Proses ini melibatkan perbandingan antara label sebenarnya (*test_labels*) dan prediksi model (*predictions*). Hasil *confusion matrix* kemudian dicetak ke konsol untuk memberikan informasi tentang performa klasifikasi pada *fold* tertentu. Selain itu, *confusion matrix* juga ditampilkan

dalam bentuk *heatmap* menggunakan *Seaborn*, memberikan visualisasi yang lebih jelas terkait distribusi kelas yang benar dan prediksi yang berhasil. Hal ini memungkinkan analisis yang lebih mendalam terhadap sejauh mana model mampu mengklasifikasikan data pada setiap kelas, memfasilitasi pemahaman yang lebih baik terkait kekuatan dan kelemahan model pada *K-Fold Cross Validation*.

4.7 Implementasi menyimpan dan konversi model

```
# Convert model to TensorFlow Lite format
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Save the TensorFlow Lite model to a file
tflite_model_filename = 'model.tflite'
with open(tflite_model_filename, 'wb') as f:
    f.write(tflite_model)

print(f"TensorFlow Lite model saved to {tflite_model_filename}")
```

Gambar 21. Menyimpan dan konversi model

Pada Gambar 22 kode tersebut bertujuan untuk mengonversi model *TensorFlow* dari format Keras ke format *TensorFlow Lite (TFLite)*, sebuah langkah penting untuk *deploy* model pada perangkat dengan sumber daya terbatas seperti perangkat *mobile*. Pertama, objek konverter di mana model merupakan model Keras yang telah dilatih sebelumnya. Selanjutnya, model tersebut diubah menjadi format *TFLite* menggunakan metode `convert()` dari objek konverter, dan hasilnya disimpan dalam variabel `tflite_model`. Selanjutnya, model *TensorFlow Lite* disimpan ke dalam file dengan nama `'model.tflite'` melalui proses penulisan data biner. Pesan informasi terakhir

mencetak lokasi penyimpanan model *TensorFlow Lite*. Dengan langkah ini, model yang lebih efisien dalam penggunaan sumber daya dapat digunakan pada perangkat dengan keterbatasan komputasi.

4.8 Implementasi Pengujian Model dengan *MTCNN*

Dalam tahap ini, dilakukan implementasi pengujian model deteksi masker menggunakan *MTCNN* (*Multi-task Cascaded Convolutional Networks*). Detektor *MTCNN* kemudian digunakan untuk mendeteksi wajah pada citra, dan hasil deteksi disimpan dalam variabel *faces*

Melalui iterasi pada hasil deteksi, koordinat dan dimensi kotak pembatas wajah diambil untuk mengekstrak *region of interest (ROI)* wajah. *ROI* wajah kemudian diubah ukurannya menjadi 224x224, diubah menjadi *format array*, dan diproses sesuai dengan *preprocessing* yang diterapkan pada model. Setelah itu, model digunakan untuk memprediksi apakah wajah tersebut menggunakan masker atau tidak.

Hasil prediksi kemudian digunakan untuk menentukan label ("Bermasker" atau "Tidak Bermasker") dan warna yang sesuai (hijau untuk bermasker dan merah untuk tidak bermasker).

Probabilitas hasil deteksi juga ditampilkan sebagai bagian dari label. Hasil akhir ditampilkan pada citra asli dengan penambahan label dan kotak pembatas berwarna.

```

▶ detector = MTCNN()
image = cv2.imread(
    'face-mask-detection/example_img/ex04.jpg', cv2.COLOR_BGR2RGB)
faces = detector.detect_faces(image)
for result in faces:
    x, y, w, h = result['box']
    x1, y1 = x + w, y + h

    # Ekstrak ROI wajah, konversikan dari BGR ke pemesanan saluran RGB,
    # dan mengubah ukurannya menjadi 224x224, dan lalu pre-proses
    face = image[y:y1, x:x1]
    face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
    face = cv2.resize(face, (224, 224))
    face = img_to_array(face)
    face = preprocess_input(face)
    face = np.expand_dims(face, axis=0)

    # Membaca wajah dengan model
    (mask, withoutMask) = model.predict(face)[0]

    # Menggunakan masker hijau, tidak bermasker merah
    label = "Bermasker" if mask > withoutMask else "Tidak Bermasker"
    color = (0, 255, 0) if label == "Bermasker" else (0, 0, 255)

    # Probabilitas hasil deteksi
    label = "{: {:.2f}%}".format(label, max(mask, withoutMask) * 100)

    # Menampilkan hasil dengan label dan kotak
    cv2.putText(image, label, (x, y - 10),
        cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
    cv2.rectangle(image, (x, y), (x1, y1), color, 2)

# Menampilkan output
cv2.imshow(" Hasil", image)
cv2.waitKey(0)

```

Gambar 4.1 Implementasi Pengujian Model dengan *MTCNN*

Dengan implementasi pada Gambar 4.11 ini, model deteksi masker diuji pada citra yang memiliki wajah, dan hasilnya ditampilkan secara visual dengan penambahan label dan kotak pembatas pada setiap wajah yang terdeteksi.

4.9 Hasil Pengujian Skenario Training

Pada semua hasil penelitian pengujian ini akan menggunakan konfigurasi sebagai berikut:

Tabel 3. Parameter Untuk Pengujian Skenario

No.	Parameter	Value
1	<i>Epoch</i>	30, 50, dan 100
2	<i>Target Size</i>	224 x 224 RGB
3	<i>Batch Size</i>	32
4	<i>Learning Rate</i>	0,0001

Tabel 3 berisi parameter-parameter yang digunakan dalam pengujian skenario. Parameter pertama adalah "*Epoch*" dengan nilai 30,50, dan 100 yang mengacu pada jumlah iterasi saat melatih model.

Parameter kedua adalah "*Target Size*" dengan nilai 224 x 224 RGB, yang menunjukkan dimensi target Gambar yang digunakan dalam proses pelatihan.

Parameter ketiga adalah "*Batch Size*" dengan nilai 32, yang menunjukkan jumlah sampel yang diproses dalam satu iterasi pelatihan. Parameter terakhir adalah "*Learning Rate*" dengan nilai 0,0001, yang mengindikasikan seberapa besar perubahan bobot model terjadi pada setiap langkah pembelajaran.

Tabel 4. Parameter Data Augmentasi

Parameter	Penjelasan
<i>rotation_range=20</i>	Rentang rotasi dalam derajat untuk memutar Gambar.
<i>zoom_range=0.15</i>	Rentang level zoom-in dan zoom-out pada Gambar.
<i>width_shift_range=0.2</i>	Rentang pergeseran horizontal Gambar.
<i>height_shift_range=0.2</i>	Rentang pergeseran vertikal Gambar.
<i>shear_range=0.15</i>	Rentang pergeseran sudut pemotongan.
<i>horizontal_flip=True</i>	Kemungkinan untuk memutar Gambar secara horizontal.
<i>fill_mode="nearest"</i>	Cara mengisi piksel yang kosong setelah augmentasi (dalam hal ini, menggunakan piksel terdekat).

Tabel 5. Arsitektur Jaringan *VGG16Net*

Layer (type)	Output Shape	Param
<i>input_1 (InputLayer)</i>	(None, 224, 224, 3)	0
<i>block1_conv1 (Conv2D)</i>	(None, 224, 224, 64)	1792
<i>block1_conv2 (Conv2D)</i>	(None, 224, 224, 64)	36928
<i>block1_pool (MaxPooling2D)</i>	(None, 112, 112, 64)	0
<i>block2_conv1 (Conv2D)</i>	(None, 112, 112, 128)	73856
<i>block2_conv2 (Conv2D)</i>	(None, 112, 112, 128)	147584
<i>block2_pool (MaxPooling2D)</i>	(None, 56, 56, 128)	0
<i>block3_conv1 (Conv2D)</i>	(None, 56, 56, 256)	295168
<i>block3_conv2 (Conv2D)</i>	(None, 56, 56, 256)	590080
<i>block3_conv3 (Conv2D)</i>	(None, 56, 56, 256)	590080
<i>block3_pool (MaxPooling2D)</i>	(None, 28, 28, 256)	0
<i>block4_conv1 (Conv2D)</i>	(None, 28, 28, 512)	1180160
<i>block4_conv2 (Conv2D)</i>	(None, 28, 28, 512)	2359808
<i>block4_conv3 (Conv2D)</i>	(None, 28, 28, 512)	2359808
<i>block4_pool (MaxPooling2D)</i>	(None, 14, 14, 512)	0
<i>block5_conv1 (Conv2D)</i>	(None, 14, 14, 512)	2359808
<i>block5_conv2 (Conv2D)</i>	(None, 14, 14, 512)	2359808
<i>block5_conv3 (Conv2D)</i>	(None, 14, 14, 512)	2359808

Layer (type)	Output Shape	Param
<i>block5_pool</i> (<i>MaxPooling2D</i>)	(None, 7, 7, 512)	0
<i>Total params:</i>		14,714,688
<i>Trainable params:</i>		0
<i>Non-trainable params:</i>		147,44,688

Tabel 5 adalah tabel yang menggambarkan arsitektur jaringan *VGG16Net*. Arsitektur ini terdiri dari beberapa layer yang dirinci dalam tabel. Setiap layer memiliki jenis (*type*), bentuk *output* (*Output Shape*), dan jumlah parameter yang digunakan (*Param*).

Jaringan *VGG16Net* terdiri dari beberapa *layer Conv2D* yang digunakan untuk melakukan konvolusi pada *input* Gambar. Setiap layer *Conv2D* memiliki output dengan dimensi yang berbeda-beda. Selain itu, terdapat juga layer *MaxPooling2D* yang berfungsi untuk mengurangi dimensi *spatialis* dari *output Conv2D*.

Arsitektur jaringan *VGG16Net* terdiri dari 5 blok. Setiap blok terdiri dari beberapa layer *Conv2D* dan diakhiri dengan *layer MaxPooling2D*. Dimensi *output* dari setiap blok secara bertahap mengecil seiring dengan meningkatnya tingkat kompleksitas fitur yang dihasilkan.

Total parameter dalam jaringan ini adalah 14.714.688. Parameter ini merupakan jumlah total bobot yang dapat diubah oleh algoritma pelatihan. Dalam kasus ini, semua parameter dalam jaringan ini ditetapkan sebagai *non-trainable*, yang berarti nilai bobotnya tetap tidak berubah selama pelatihan.

Tabel 6. Susunan Pengujian Model *Convolutional Neural Network* .

Layer (type)	Output Shape	Param
<i>input_1</i> (<i>InputLayer</i>)	(None, 224, 224, 3)	0
<i>block1_conv1</i> (<i>Conv2D</i>)	(None, 224, 224, 64)	1792
<i>block1_conv2</i> (<i>Conv2D</i>)	(None, 224, 224, 64)	36928
<i>block1_pool</i> (<i>MaxPooling2D</i>)	(None, 112, 112, 64)	0
<i>block2_conv1</i> (<i>Conv2D</i>)	(None, 112, 112, 128)	73856
<i>block2_conv2</i> (<i>Conv2D</i>)	(None, 112, 112, 128)	147584
<i>block2_pool</i> (<i>MaxPooling2D</i>)	(None, 56, 56, 128)	0
<i>block3_conv1</i> (<i>Conv2D</i>)	(None, 56, 56, 256)	295168
<i>block3_conv2</i> (<i>Conv2D</i>)	(None, 56, 56, 256)	590080
<i>block3_conv3</i> (<i>Conv2D</i>)	(None, 56, 56, 256)	590080
<i>block3_pool</i> (<i>MaxPooling2D</i>)	(None, 28, 28, 256)	0
<i>block4_conv1</i> (<i>Conv2D</i>)	(None, 28, 28, 512)	1180160
<i>block4_conv2</i> (<i>Conv2D</i>)	(None, 28, 28, 512)	2359808
<i>block4_conv3</i> (<i>Conv2D</i>)	(None, 28, 28, 512)	2359808
<i>block4_pool</i> (<i>MaxPooling2D</i>)	(None, 14, 14, 512)	0
<i>block5_conv1</i> (<i>Conv2D</i>)	(None, 14, 14, 512)	2359808
<i>block5_conv2</i> (<i>Conv2D</i>)	(None, 14, 14, 512)	2359808
<i>block5_conv3</i> (<i>Conv2D</i>)	(None, 14, 14, 512)	2359808
<i>block5_pool</i> (<i>MaxPooling2D</i>)	(None, 7, 7, 512)	0
<i>average_pooling2d</i> (<i>AveragePooling2D</i>)	(None, 1, 1, 512)	0
<i>flatten</i> (<i>Flatten</i>)	(None, 512)	0
<i>dense</i> (<i>Dense</i>)	(None, 128)	65664
<i>dropout</i> (<i>Dropout</i>)	(None, 128)	0
<i>dense_1</i> (<i>Dense</i>)	(None, 2)	258
<i>Total params:</i>		14,780,610
<i>Trainable params</i>		65,922
<i>Non-trainable params:</i>		14,714,688

Tabel 6 menampilkan susunan serta parameter-parameter dari Model *Convolutional Neural Network* yang diuji. Model ini terdiri dari beberapa

jenis layer yang terhubung secara berurutan. *Layer input* awal (*input_1*) memiliki dimensi $\text{None} \times 224 \times 224 \times 3$, dengan total parameter sebesar 0.

Selanjutnya, terdapat beberapa layer konvolusi (*Conv2D*) dan *layer pooling* (*MaxPooling2D*) yang membentuk bagian-bagian blok (*block*) dari model ini.

Setiap blok terdiri dari beberapa layer *conv2D* yang memiliki dimensi yang berbeda, dengan parameter yang berbeda pula. Blok-blok ini secara berturut-turut meningkatkan kompleksitas pemrosesan dan mengekstraksi fitur-fitur dari *input* Gambar.

Pada akhirnya, terdapat beberapa layer lain seperti *average_pooling2d* yang melakukan proses *pooling* rata-rata serta layer *flatten* yang mengubah *output* menjadi dimensi 1D. Terakhir, terdapat dua layer *dense* (*Dense*) yang berfungsi sebagai layer terhubung penuh (*fully connected*) dengan *output layer* terakhir berdimensi 2 untuk melakukan klasifikasi.

Total parameter dari model ini sebesar 14,780,610, dengan 65,922 parameter yang dapat diubah (*trainable*) dan sisanya 14,714,688 parameter tidak dapat diubah (*non-trainable*). Informasi ini relevan untuk memahami struktur dan kompleksitas model *Convolutional Neural Network* yang diuji dalam konteks penelitian ini.

Dalam konfigurasi tersebut melibatkan skenario dengan menggunakan metode *K-fold Cross Validation* yang terdiri dari 3 *fold*, 5 *fold*, dan 7 *fold* dengan menggunakan 3 *optimizer* yaitu *Adam*, *SGD*, dan *RMSprop*. Lalu

Ketika mendapatkan model yang terbaik maka akan diujikan dengan uji data tunggal yang melibatkan data baru citra masker untuk melihat seberapa akurat model tersebut untuk mendeteksi masker.

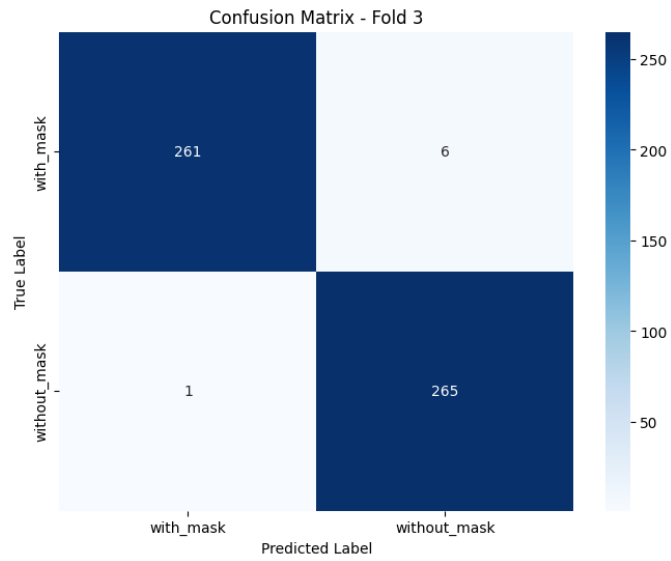
4.9.1 Hasil *Adaptive Moment Estimation*

4.9.1.1 Hasil Skenario 30 Epochs

Tabel 7. Hasil *Adam* 3 fold 30 epochs

<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	603.12	0.9831	0.9831	0.9828	0.9840
2	582.73	0.9869	0.9868	0.9872	0.9865
3	589.55	0.9869	0.9869	0.9870	0.9869
<i>Avg</i>	591.47	98.95	98.95	98.93	98.93

Hasil skenario 30 Epochs dengan metode Adam 3 fold menunjukkan performa yang sangat baik. Rata-rata waktu pelatihan adalah 591.47 detik dengan akurasi, F1 Score, Precision, dan Recall masing-masing sekitar 98.95%. Ini menunjukkan bahwa model memiliki kinerja yang konsisten dan efisien dalam mengklasifikasikan data dengan tepat.



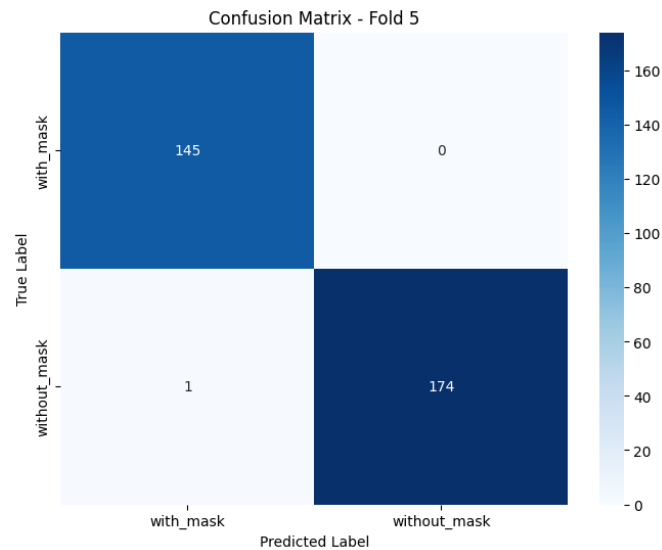
Gambar 22. Hasil *Confusion Matrix Adam 3 Fold 30 epochs*

Nilai-nilai pada gambar 23 dalam matriks adalah: TP (True Positive) = 261, FN (False Negative) = 6, FP (False Positive) = 1, TN (True Negative) = 265. Ini menunjukkan bahwa model memiliki kinerja yang baik dalam mengklasifikasikan data dengan tepat.

Tabel 8. Hasil Adam 5 fold 30 epochs

<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	614.11	0.9844	0.9843	0.9852	0.9840
2	585.44	0.9781	0.9781	0.9789	0.9783
3	587.56	0.9906	0.9906	0.9902	0.9912
4	574.32	0.9937	0.9937	0.9937	0.9937
5	570.84	0.9969	0.9968	0.9966	0.9971
<i>Avg</i>	<i>578.45</i>	<i>99.74</i>	<i>99.74</i>	<i>99.74</i>	<i>99.74</i>

Hasil skenario 30 Epochs dengan metode Adam 5 fold menunjukkan performa yang sangat baik. Rata-rata waktu pelatihan adalah 578.45 detik dengan akurasi, F1 Score, Precision, dan Recall masing-masing sekitar 99.74%. Ini menunjukkan bahwa model memiliki kinerja yang konsisten dan efisien dalam mengklasifikasikan data dengan tepat.

**Gambar 23.** Hasil Confusion Matrix Adam 5 Fold 30 Epochs

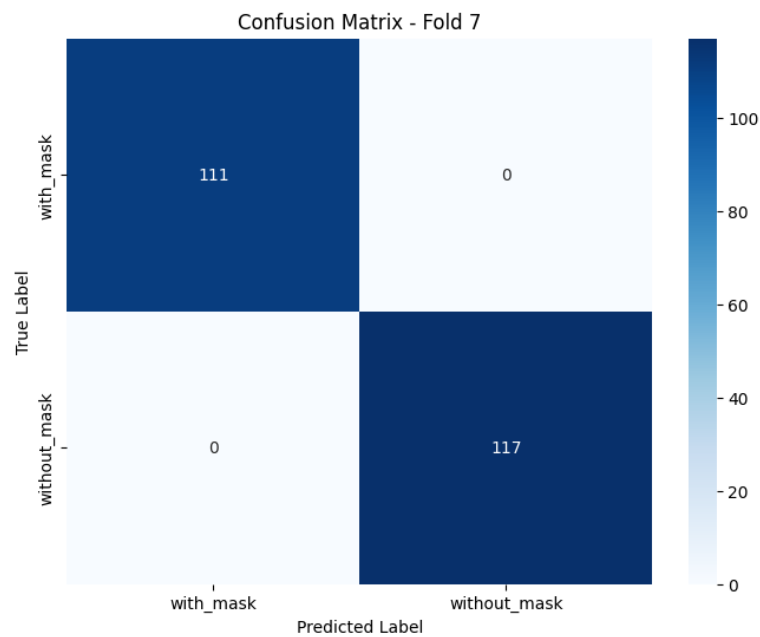
Nilai-nilai dalam matriks adalah: TP (True Positive) = 145, FN (False Negative) = 0, FP (False Positive) = 1, TN (True Negative) = 174. Ini menunjukkan

bahwa model memiliki kinerja yang baik dalam mengklasifikasikan data dengan tepat.

Tabel 9. Hasil Adam 7 Fold 30 Epochs

<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	667.82	0.9738	0.9737	0.9732	0.9756
2	594.83	0.9782	0.9781	0.9791	0.9773
3	616.90	0.9869	0.9868	0.9860	0.9880
4	729.01	0.9956	0.9956	0.9956	0.9957
5	648.70	0.9956	0.9956	0.9960	0.9952
6	609.66	10000	10000	10000	10000
7	650.00	10000	10000	10000	10000
Avg	644.26	99.44	99.43	99.44	99.43

Skenario 30 Epochs menggunakan metode Adam 7 fold menghasilkan performa yang luar biasa. Waktu pelatihan rata-rata adalah 644.26 detik dengan akurasi, F1 Score, Precision, dan Recall masing-masing sekitar 99.44%. Hal ini menunjukkan bahwa model mampu mengklasifikasikan data dengan tepat dan efisien.

**Gambar 24.** Hasil Confusion Matrix Adam 7 Fold 30 Epochs

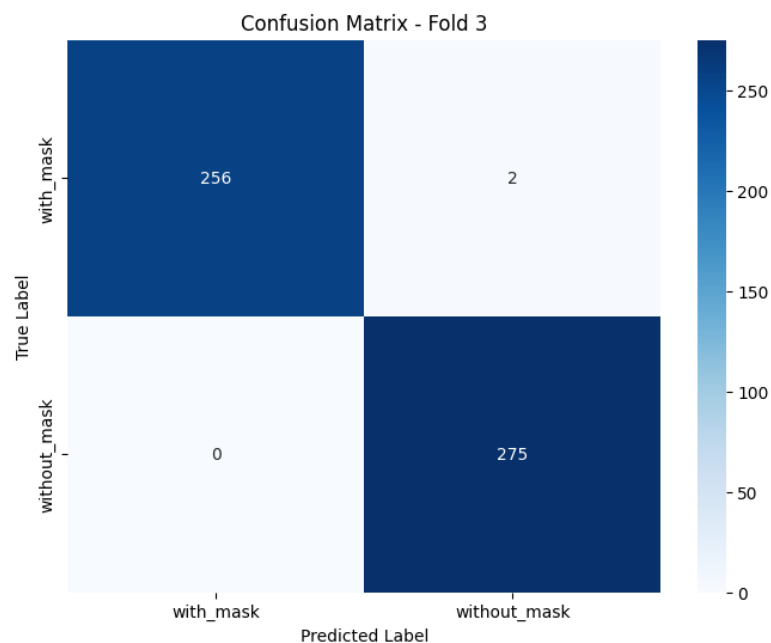
Nilai-nilai dalam matriks adalah: TP (True Positive) = 111, FN (False Negative) = 0, FP (False Positive) = 1, TN (True Negative) = 117. Hal ini menunjukkan bahwa model mampu mengklasifikasikan data dengan tepat.

4.9.1.2 Hasil Skenario 50 Epochs

Tabel 10. Hasil Adam 3 Fold 50 Epochs

<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	987.04	0.9869	0.9869	0.9867	0.9871
2	972.32	0.9831	0.9831	0.9837	0.9829
3	932.10	0.9962	0.9962	0.9964	0.9961
Avg	963.15	99.04	99.04	99.04	99.04

Hasil skenario 50 Epochs dengan metode Adam 3 fold menunjukkan performa yang sangat baik. Rata-rata waktu pelatihan adalah 963.15 detik dengan akurasi, F1 Score, Precision, dan Recall masing-masing sekitar 99.04%. Ini menunjukkan bahwa model memiliki kinerja yang konsisten dan efisien dalam mengklasifikasikan data dengan tepat.



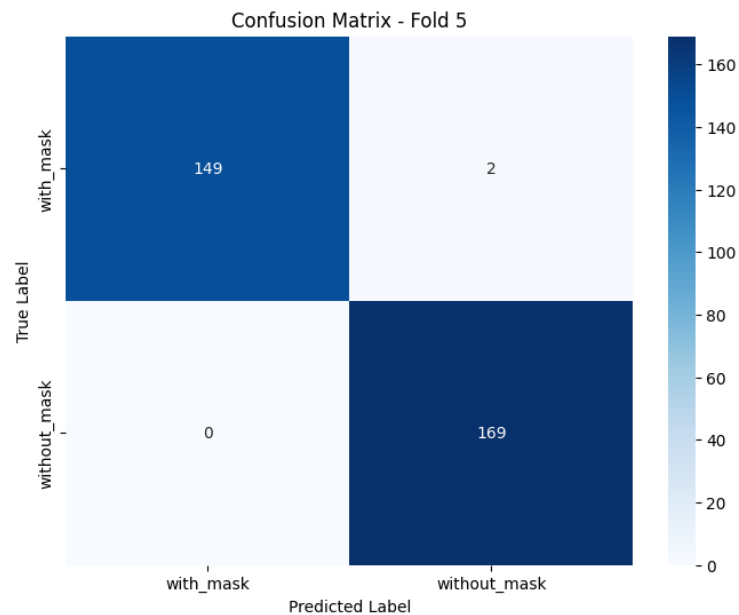
Gambar 25. Confusion Matrix Adam 3 fold 50 Epochs

Nilai-nilai dalam matriks adalah: TP (True Positive) = 256, FN (False Negative) = 2, FP (False Positive) = 0, TN (True Negative) = 275. Ini menunjukkan bahwa model memiliki kinerja yang baik dalam mengklasifikasikan data dengan tepat.

Tabel 11. Hasil Adam 5 fold 50 epochs

<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	1025.13	0.9844	0.9844	0.9844	0.9845
2	1010.50	0.9812	0.9812	0.9817	0.9815
3	1017.58	0.9937	0.9937	0.9935	0.9941
4	974.55	0.9969	0.9969	0.9968	0.9970
5	966.59	0.9937	0.9937	0.9942	0.9934
<i>Avg</i>	991.91	99.34	99.34	99.34	99.34

Hasil skenario 50 Epochs dengan metode Adam 5 fold menunjukkan performa yang sangat baik. Rata-rata waktu pelatihan adalah 991.91 detik dengan akurasi, F1 Score, Precision, dan Recall masing-masing sekitar 99.34%. Ini menunjukkan bahwa model memiliki kinerja yang konsisten dan efisien dalam mengklasifikasikan data dengan tepat.

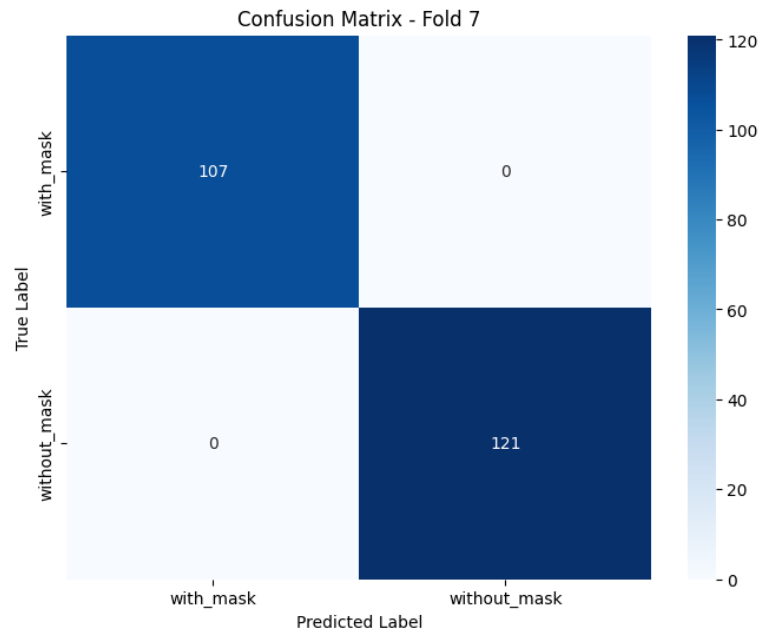
**Gambar 26.** Confusion Matrix Adam 5 Fold 50 epochs

Nilai-nilai dalam matriks adalah: TP (True Positive) = 149, FN (False Negative) = 2, FP (False Positive) = 0, TN (True Negative) = 169. Ini menunjukkan bahwa model memiliki kinerja yang baik dalam mengklasifikasikan data dengan tepat.

Tabel 12. Hasil Adam 7 fold 50 epochs

<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	1055.74	0.9738	0.9737	0.9748	0.9732
2	971.85	0.9956	0.9956	0.9956	0.9957
3	953.33	0.9913	0.9912	0.9912	0.9912
4	947.38	0.9956	0.9956	0.9953	0.9960
5	958.95	0.9956	0.9956	0.9958	0.9955
6	957.35	10000	10000	10000	10000
7	936.86	10000	10000	10000	10000
<i>Avg</i>	996.71	99.24	99.24	99.24	99.24

Hasil skenario 50 Epochs dengan metode Adam 7 fold menunjukkan performa yang sangat baik. Rata-rata waktu pelatihan adalah 996.71 detik dengan akurasi, F1 Score, Precision, dan Recall masing-masing sekitar 99.24%. Ini menunjukkan bahwa model memiliki kinerja yang konsisten dan efisien dalam mengklasifikasikan data dengan tepat.

**Gambar 27.** Confusion Matrix Adam 7 fold 50 epochs

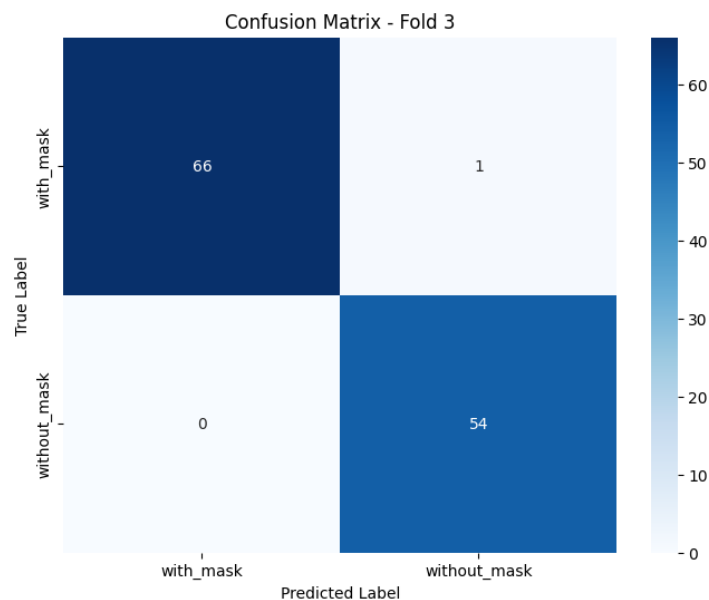
Nilai-nilai dalam matriks adalah: TP (True Positive) = 107, FN (False Negative) = 0, TN (True Negative) = 121. Ini menunjukkan bahwa model memiliki kinerja yang baik dalam mengklasifikasikan data dengan tepat.

4.9.1.3 Hasil Skenario 100 Epochs

Tabel 13. Hasil Adam 3 fold 100 epochs

<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	438.55	0.9918	0.9918	0.9912	0.9924
2	383.61	0.9918	0.9918	0.9918	0.9919
3	409.35	0.9917	0.9917	0.9909	0.9925
<i>Avg</i>	410.17	99.17	99.17	99.13	99.23

Hasil skenario 100 Epochs dengan metode Adam 3 fold menunjukkan performa yang sangat baik. Rata-rata waktu pelatihan adalah 410.17 detik dengan akurasi, F1 Score, Precision, dan Recall masing-masing sekitar 99.17%. Ini menunjukkan bahwa model memiliki kinerja yang konsisten dan efisien dalam mengklasifikasikan data dengan tepat.



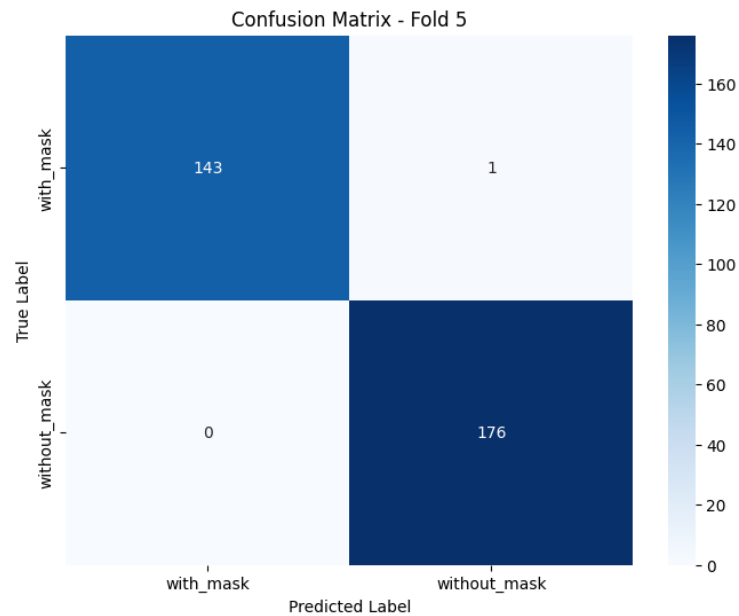
Gambar 28. Confusion Matrix Adam 3 fold 100 epochs

Nilai-nilai dalam matriks adalah: TP (True Positive) = 66, FN (False Negative) = 1, TN (True Negative) = 54. Ini menunjukkan bahwa model memiliki kinerja yang baik dalam mengklasifikasikan data dengan tepat.

Tabel 14. Hasil Adam 5 fold 100 epochs

<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	2017.13	0.9781	0.9781	0.9789	0.9783
2	1951.92	0.9937	0.9937	0.9943	0.9932
3	1933.03	10000	10000	10000	10000
4	1986.06	10000	10000	10000	10000
5	1935.51	0.9969	0.9968	0.9972	0.9965
<i>Avg</i>	1962.92	99.17	99.16	99.17	99.16

Berdasarkan Tabel 14, dapat dilihat bahwa model yang dilatih dengan metode Adam selama 5 fold dan 100 epochs menunjukkan performa yang sangat baik. Waktu pelatihan rata-rata adalah 1962.92 detik per fold. Akurasi, F1 Score, Presisi, dan Recall rata-rata mencapai lebih dari 99%, kecuali pada fold pertama yang memiliki nilai sedikit lebih rendah namun tetap tinggi yaitu 97%.

**Gambar 29.** Confusion Matrix Adam 5 fold 100 epochs

Matriks kebingungan adalah tabel yang digunakan untuk menggambarkan kinerja model klasifikasi. Nilai-nilai dalam matriks adalah

True Positives (TP) = 66: Ini adalah kasus di mana model dengan benar memprediksi kelas positif.

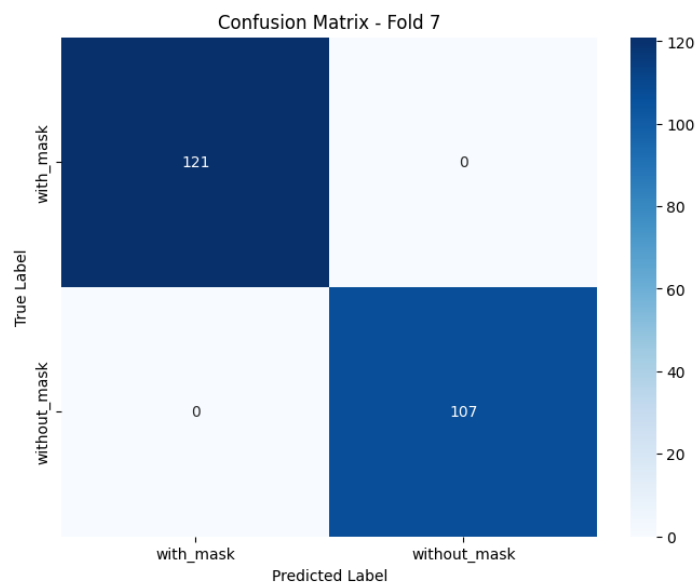
False Negatives (FN) = 1: Ini adalah kasus di mana model salah memprediksi kelas negatif ketika sebenarnya positif.

True Negatives (TN) = 54: Ini adalah kasus di mana model dengan benar memprediksi kelas negatif.

Tabel 15. Hasil Adam 7 fold 100 epochs

<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	1976.62	0.9825	0.9822	0.9822	0.9822
2	1931.75	0.9956	0.9956	0.9956	0.9957
3	1933.72	0.9956	0.9956	0.9955	0.9958
4	1943.72	10000	10000	10000	10000
5	2037.40	10000	10000	10000	10000
6	1961.05	10000	10000	10000	10000
7	1981.51	10000	10000	10000	10000
<i>Avg</i>	1932.10	99.19	99.19	99.18	99.19

Berdasarkan Tabel 15, dapat dilihat bahwa model yang dilatih dengan metode Adam selama 7 fold dan 100 epochs menunjukkan performa yang sangat baik. Waktu pelatihan rata-rata adalah 1932.10 detik per fold. Akurasi, F1 Score, Presisi, dan Recall rata-rata mencapai lebih dari 99%

**Gambar 30.** Confusion Matrix Adam 7 fold 100 epochs

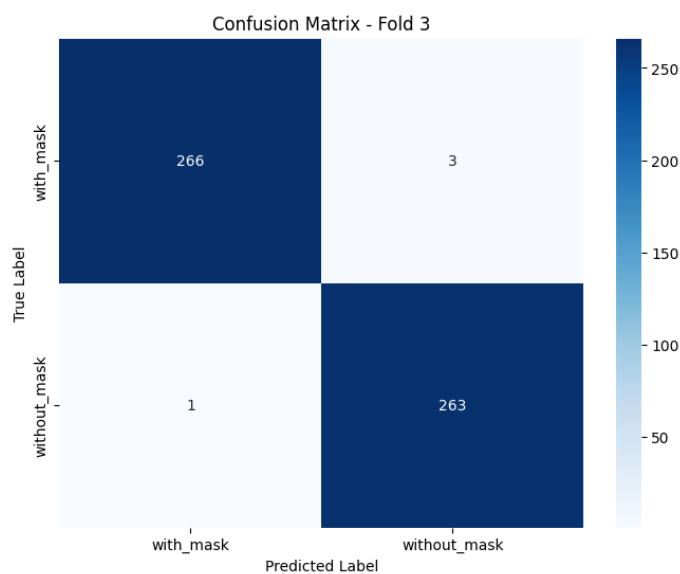
4.9.2 Hasil *Root Mean Square Propagation*

4.9.2.1 Hasil Skenario 30 Epochs

Tabel 16. Hasil *RMSprop* 3 fold 30 epochs

<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	572.97	0.9813	0.9813	0.9814	0.9814
2	527.99	0.9831	0.9831	0.9837	0.9829
3	563.46	0.9925	0.9925	0.9925	0.9925
Avg	554.47	98.97	98.97	98.98	98.97

Berdasarkan Tabel 16, dapat dilihat bahwa model yang dilatih dengan metode RMSprop selama 3 fold dan 30 epochs menunjukkan performa yang sangat baik. Waktu pelatihan rata-rata adalah 554.47 detik per fold. Akurasi, F1 Score, Presisi, dan Recall rata-rata mencapai lebih dari 98.97%, yang menunjukkan bahwa model memiliki kinerja yang konsisten dan efisien dalam mengklasifikasikan data dengan tepat.



Gambar 31. *Confusion Matrix RMSprop* 3 fold 30 epochs

Matriks kebingungan adalah tabel yang digunakan untuk menggambarkan kinerja model klasifikasi. Nilai-nilai dalam matriks adalah:

True Positives (TP) = 266: Ini adalah kasus di mana model dengan benar memprediksi kelas positif.

False Negatives (FN) = 1: Ini adalah kasus di mana model salah memprediksi kelas negatif ketika sebenarnya positif.

True Negatives (TN) = 263: Ini adalah kasus di mana model dengan benar memprediksi kelas negatif.

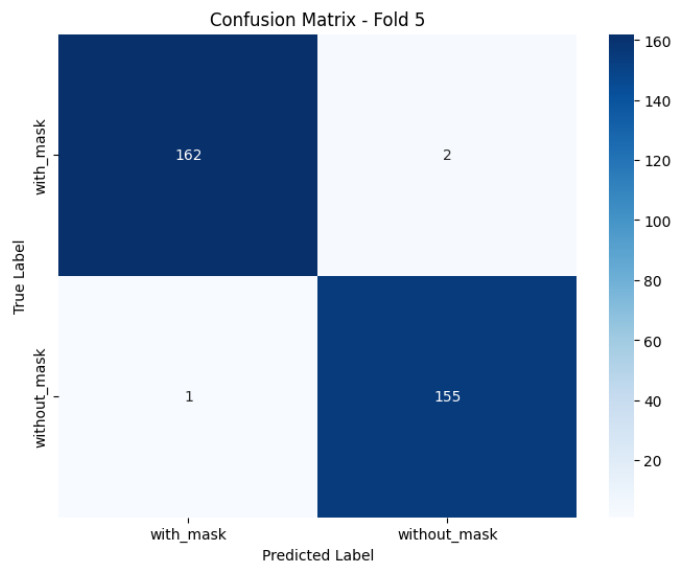
False Positives (FP) = 3: Ini adalah kasus di mana model salah memprediksi kelas positif ketika sebenarnya negatif.

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kinerja yang konsisten dan efisien dalam mengklasifikasikan data dengan tepat.

Tabel 17. Hasil *RMSprop* 5 fold 30 epochs

<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	608.22	0.9781	0.9780	0.9794	0.9771
2	580.78	0.9937	0.9937	0.9937	0.9937
3	585.96	0.9906	0.9906	0.9906	0.9909
4	577.98	0.9812	0.9812	0.9810	0.9818
5	583.17	0.9906	0.9906	0.9906	0.9907
<i>Avg</i>	582.27	98.02	98.02	98.03	98.02

Berdasarkan Tabel 17, dapat dilihat bahwa model yang dilatih dengan metode RMSprop selama 5 fold dan 30 epochs menunjukkan performa yang sangat baik. Waktu pelatihan rata-rata adalah 582.27 detik per fold. Akurasi, F1 Score, Presisi, dan Recall rata-rata mencapai lebih dari 98.02%, yang menunjukkan bahwa model memiliki kinerja yang konsisten dan efisien dalam mengklasifikasikan data dengan tepat.

**Gambar 32.** Confusion Matrix *RMSprop* 5 fold 30 Epochs

Matriks kebingungan adalah tabel yang digunakan untuk menggambarkan kinerja model klasifikasi. Nilai-nilai dalam matriks adalah:

True Positives (TP) = 162: Ini adalah kasus di mana model dengan benar memprediksi kelas positif.

False Negatives (FN) = 2: Ini adalah kasus di mana model salah memprediksi kelas negatif ketika sebenarnya positif.

True Negatives (TN) = 155: Ini adalah kasus di mana model dengan benar memprediksi kelas negatif.

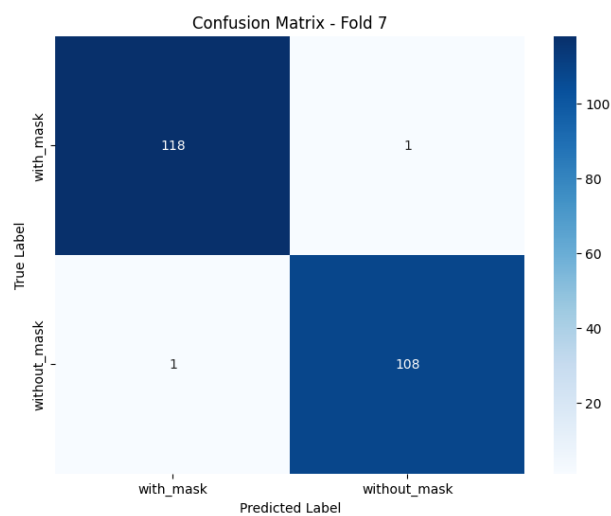
False Positives (FP) = 3: Ini adalah kasus di mana model salah memprediksi kelas positif ketika sebenarnya negatif.

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kinerja yang konsisten dan efisien dalam mengklasifikasikan data dengan tepat.

Tabel 18. Hasil *RMSprop* 7 fold 30 epochs

<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	646.84	0.9782	0.9782	0.9782	0.9782
2	639.78	0.9956	0.9956	0.9956	0.9957
3	626.03	0.9825	0.9822	0.9848	0.9802
4	632.94	0.9956	0.9956	0.9953	0.9959
5	635.77	0.9868	0.9868	0.9871	0.9870
6	595.36	0.9956	0.9956	0.9958	0.9955
7	592.86	0.9912	0.9912	0.9912	0.9912
<i>Avg</i>	634.94	99.00	98.99	98.99	98.99

Berdasarkan Tabel 18, dapat dilihat bahwa model yang dilatih dengan metode RMSprop selama 7 fold dan 30 epochs menunjukkan performa yang sangat baik. Waktu pelatihan rata-rata adalah 634.94 detik per fold. Akurasi, F1 Score, Presisi, dan Recall rata-rata mencapai lebih dari 98.99%, yang menunjukkan bahwa model memiliki kinerja yang konsisten dan efisien dalam mengklasifikasikan data dengan tepat.

**Gambar 33.** *Confusion Matrix RMSprop* 7 fold 30 epochs

Matriks kebingungan adalah tabel yang digunakan untuk menggambarkan kinerja model klasifikasi. Nilai-nilai dalam matriks adalah:

True Positives (TP) = 118: Ini adalah kasus di mana model dengan benar memprediksi kelas positif.

False Negatives (FN) = 1: Ini adalah kasus di mana model salah memprediksi kelas negatif ketika sebenarnya positif.

True Negatives (TN) = 107: Ini adalah kasus di mana model dengan benar memprediksi kelas negatif.

False Positives (FP) = 1: Ini adalah kasus di mana model salah memprediksi kelas positif ketika sebenarnya negatif.

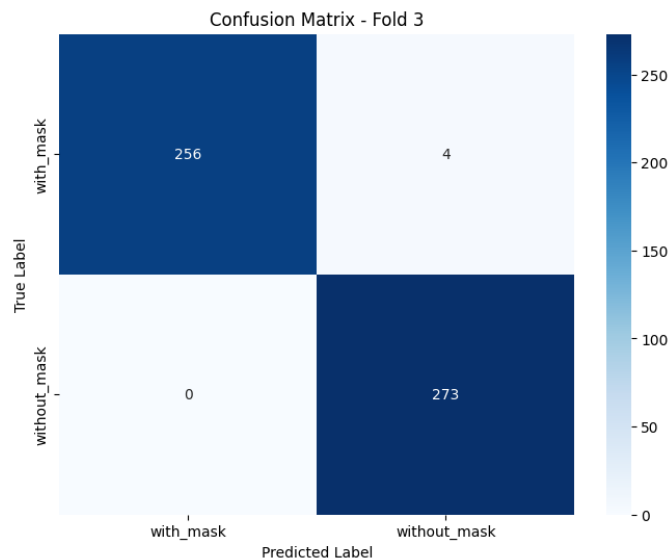
Dengan demikian, hasil ini menunjukkan bahwa model memiliki kinerja yang konsisten dan efisien dalam mengklasifikasikan data dengan tepat.

4.9.2.2 Hasil Skenario 50 Epochs

Tabel 19. Hasil *RMSprop* 3 fold 50 epochs

<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	1001.41	0.9888	0.9887	0.9886	0.9889
2	946.25	0.9850	0.9850	0.9857	0.9847
3	929.85	0.9925	0.9925	0.9928	0.9923
<i>Avg</i>	958.83	98.88	98.87	98.88	98.88

Berdasarkan Tabel 19, dapat dilihat bahwa model yang dilatih dengan metode RMSprop selama 3 fold dan 50 epochs menunjukkan performa yang sangat baik. Waktu pelatihan rata-rata adalah 958.83 detik per fold. Akurasi, F1 Score, Presisi, dan Recall rata-rata mencapai lebih dari 98.88%, yang menunjukkan bahwa model memiliki kinerja yang konsisten dan efisien dalam mengklasifikasikan data dengan tepat.



Gambar 34. Confusion Matrix *RMSprop* 3 fold 50 epochs

Matriks kebingungan untuk Fold 3 adalah tabel yang digunakan untuk menggambarkan kinerja model klasifikasi. Nilai-nilai dalam matriks adalah:

True Positives (TP) = 256: Ini adalah kasus di mana model dengan benar memprediksi kelas positif.

False Negatives (FN) = 4: Ini adalah kasus di mana model salah memprediksi kelas negatif ketika sebenarnya positif.

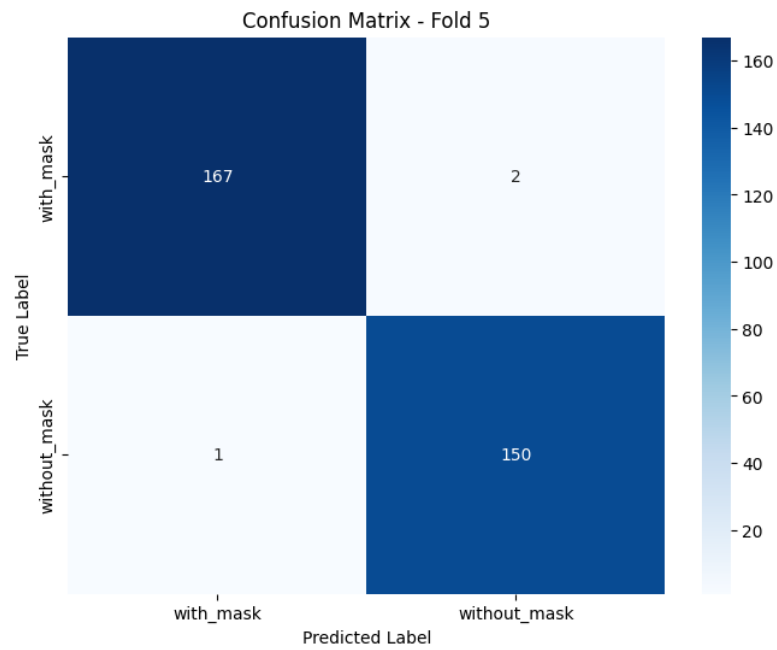
True Negatives (TN) = 273: Ini adalah kasus di mana model dengan benar memprediksi kelas negatif.

False Positives (FP) = 0: Ini adalah kasus di mana model salah memprediksi kelas positif ketika sebenarnya negatif.

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kinerja yang konsisten dan efisien dalam mengklasifikasikan data dengan tepat.

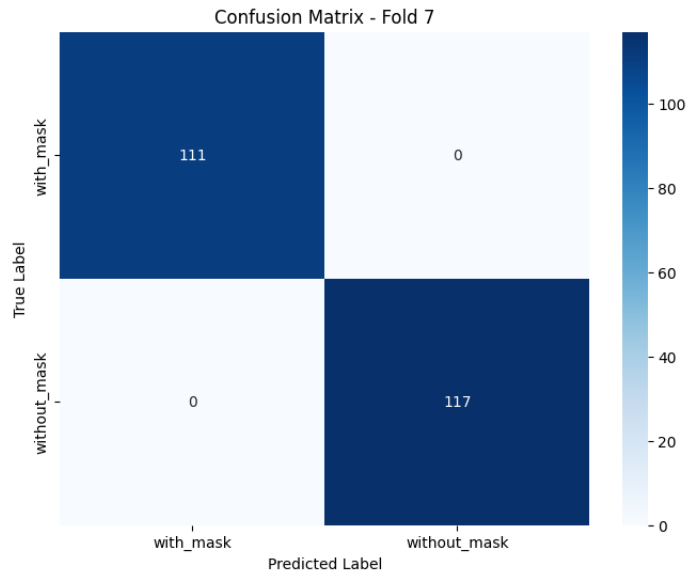
Tabel 20. Hasil *RMSprop* 5 fold 50 epochs

<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	976.64	0.9781	0.9781	0.9782	0.9785
2	1003.52	0.9969	0.9969	0.9968	0.9970
3	976.48	0.9906	0.9905	0.9915	0.9898
4	973.72	0.9969	0.9969	0.9970	0.9968
5	985.76	0.9906	0.9906	0.9904	0.9908
<i>Avg</i>	981.38	98.60	98.60	98.64	98.59

**Gambar 35.** Confusion Matrix *RMSprop* 5 fold 50 epochs

Tabel 21. Hasil *RMSprop* 7 fold 50 epochs

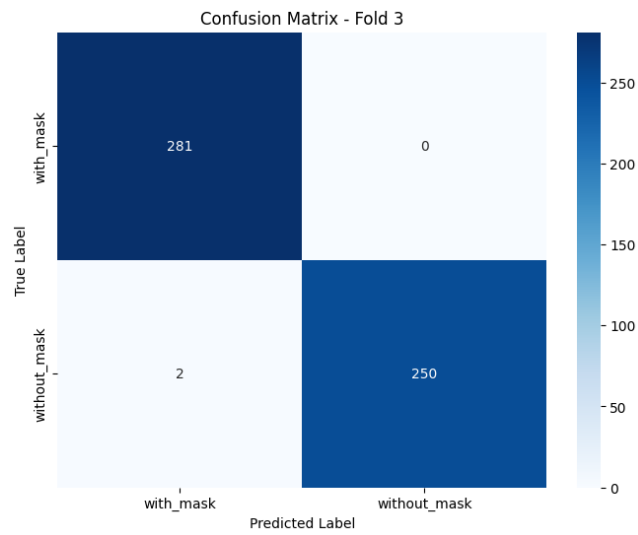
<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	980.81	0.9825	0.9825	0.9828	0.9823
2	971.00	0.9782	0.9782	0.9790	0.9783
3	999.07	0.9956	0.9956	0.9954	0.9959
4	952.50	10000	10000	10000	10000
5	1001.94	0.9956	0.9956	0.9955	0.9957
6	980.84	0.9912	0.9912	0.9906	0.9919
7	991.53	10000	10000	10000	10000
<i>Avg</i>	991.01	98.90	98.89	98.88	98.90

**Gambar 36.** *Confusion Matrix RMSprop* 7 fold 50 epochs

4.9.2.3 Hasil Skenario 100 Epochs

Tabel 22. Hasil *RMSprop* 3 fold 100 epochs

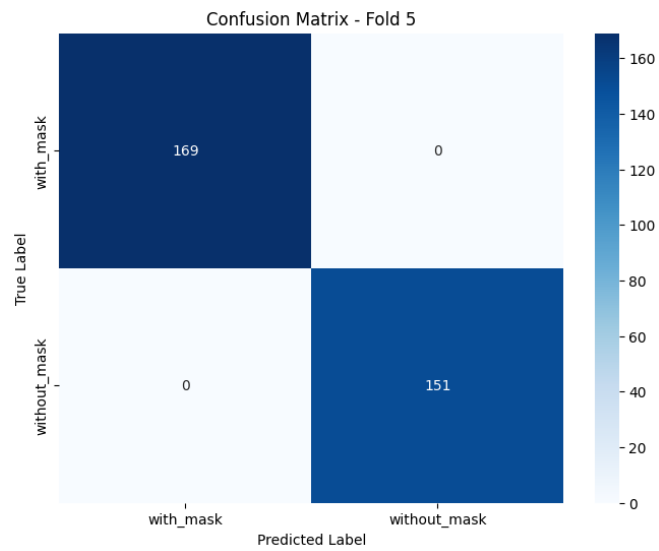
<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	1872.01	0.9888	0.9888	0.9890	0.9888
2	1843.87	0.9906	0.9906	0.9913	0.9901
3	1824.93	0.9962	0.9962	0.9965	0.9960
<i>Avg</i>	1846.94	99.18	99.17	99.18	99.17



Gambar 37. Confusion Matrix *RMSprop* 3 fold 100 epochs

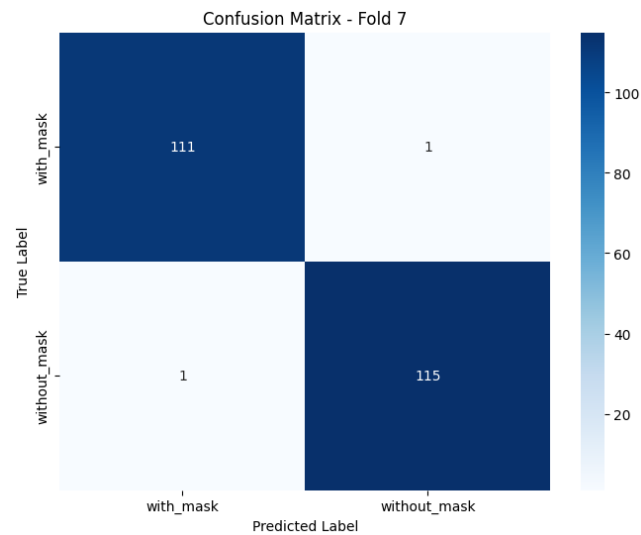
Tabel 23. Hasil *RMSprop* 5 fold 100 epochs

<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	1920.81	0.9688	0.9687	0.9706	0.9688
2	1853.51	0.9906	0.9906	0.9906	0.9907
3	1892.16	0.9969	0.9969	0.9969	0.9969
4	1891.39	10000	10000	10000	10000
5	1882.47	10000	10000	10000	10000
<i>Avg</i>	1892.57	99.06	99.05	99.07	99.05

**Gambar 38.** Confusion Matrix *RMSprop* 5 fold 100 epochs

Tabel 24. Hasil *RMSprop* 7 fold 100 epochs

<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	1580.14	0.9913	0.9912	0.9905	0.9921
2	1524.02	0.9913	0.9913	0.9917	0.9910
3	1515.46	0.9956	0.9956	0.9957	0.9957
4	1510.66	10000	10000	10000	10000
5	1523.11	0.9956	0.9956	0.9951	0.9960
6	1513.67	10000	10000	10000	10000
7	1517.22	0.9912	0.9912	0.9912	0.9912
<i>Avg</i>	1917.11	99.18	99.17	99.17	99.17

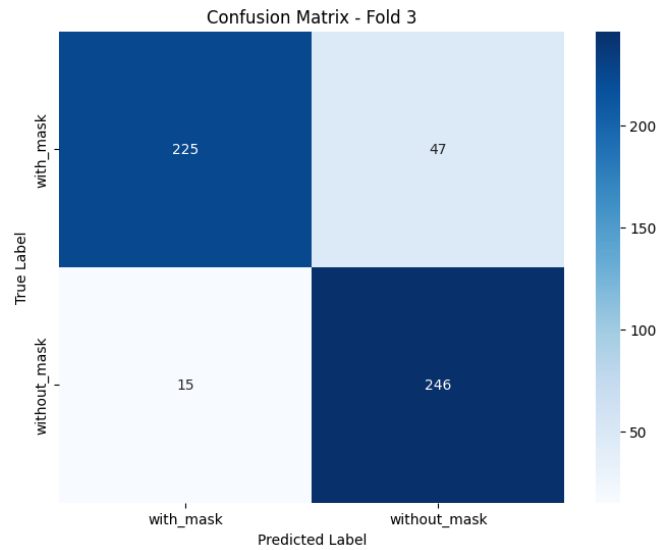
**Gambar 39.** Confusion Matrix *RMSprop* 7 fold 100 epochs

4.9.3 Hasil *Stochastic Gradient Descent*

4.9.3.1 Hasil Skenario *SGD 30 Epochs*

Tabel 25. Hasil *SGD 3 fold 30 epochs*

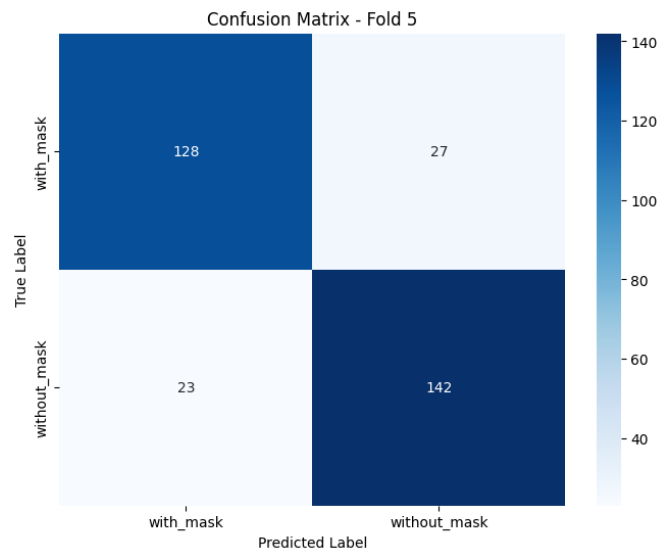
<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	603.68	0.6161	0.6157	0.6174	0.6167
2	600.09	0.7955	0.7953	0.7965	0.7953
3	577.15	0.8837	0.8835	0.8885	0.8849
Avg	593.97	69.83	69.62	70.60	69.57



Gambar 40. *Confusion Matrix SGD 3 fold 30 epochs*

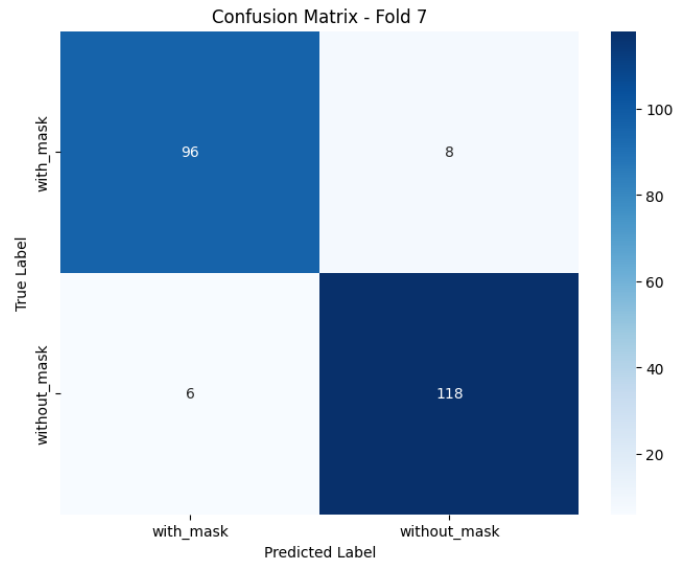
Tabel 26. Hasil *SGD 5 fold 30 epochs*

<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	603.10	0.3063	0.2754	0.2663	0.3025
2	575.32	0.3844	0.3720	0.3787	0.3913
3	566.63	0.5938	0.5934	0.5948	0.5942
4	579.46	0.7875	0.7867	0.8056	0.7956
5	587.69	0.8438	0.8435	0.8440	0.8432
<i>Avg</i>	583.70	61.63	60.85	68.64	65.63

**Gambar 41.** *Confusion Matrix SGD 5 fold 30 epochs*

Tabel 27. Hasil *SGD 7 fold 30 epochs*

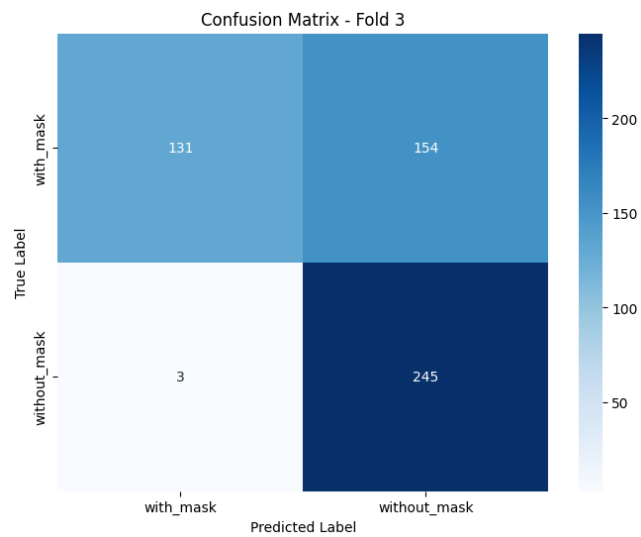
<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	615.58	0.3063	0.2754	0.2663	0.3025
2	586.69	0.8384	0.8382	0.8413	0.8390
3	580.33	0.8777	0.8776	0.8779	0.8775
4	587.32	0.8952	0.8952	0.8952	0.8953
5	591.50	0.9079	0.9076	0.9109	0.9072
6	591.08	0.9123	0.9123	0.9180	0.9180
7	584.90	0.9386	0.9380	0.9388	0.9373
<i>Avg</i>	627.56	91.68	91.37	91.84	91.25

**Gambar 42.** *Confusion Matrix SGD 7 fold 30 epochs*

4.9.3.2 Hasil Skenario *SGD 50 Epochs*

Tabel 28. Hasil *SGD 3 fold 50 epochs*

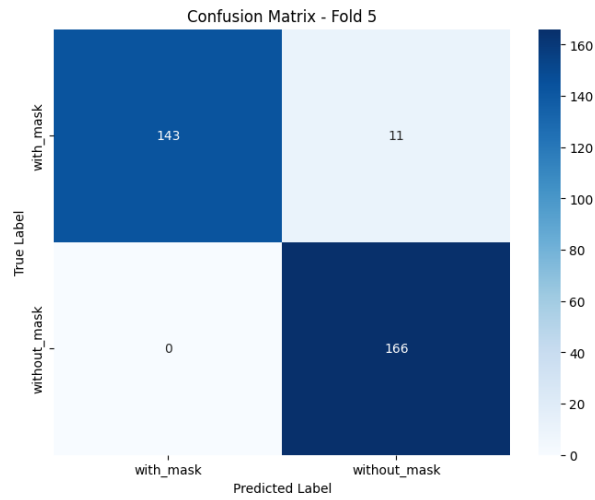
<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	938.66	0.5094	0.5087	0.5140	0.5138
2	890.81	0.7223	0.7111	0.7605	0.7212
3	843.75	0.7054	0.6913	0.7958	0.7238
<i>Avg</i>	892.41	65.27	64.80	70.77	67.94



Gambar 43. *Confusion Matrix SGD 3 fold 50 epochs*

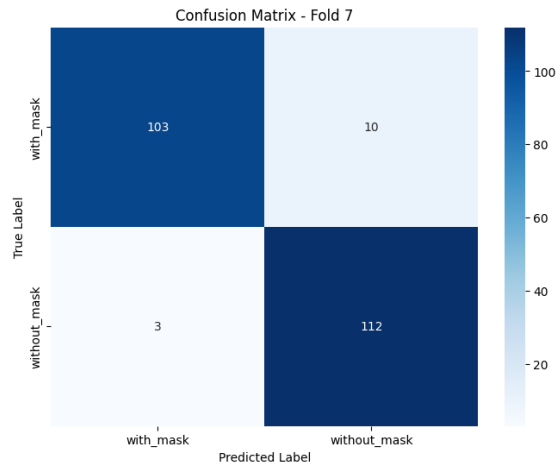
Tabel 29. Hasil *SGD 5 fold 50 epochs*

<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	969.38	0.8406	0.8401	0.8522	0.8440
2	996.15	0.9031	0.9029	0.9127	0.9062
3	969.30	0.9656	0.9655	0.9684	0.9650
4	976.22	0.9594	0.9592	0.9631	0.9586
5	977.07	0.9656	0.9654	0.9689	0.9643
<i>Avg</i>	976.98	87.43	87.20	89.03	87.76

**Gambar 44.** *Confusion Matrix SGD 5 fold 50 epochs*

Tabel 30. Hasil *SGD 7 fold 50 epochs*

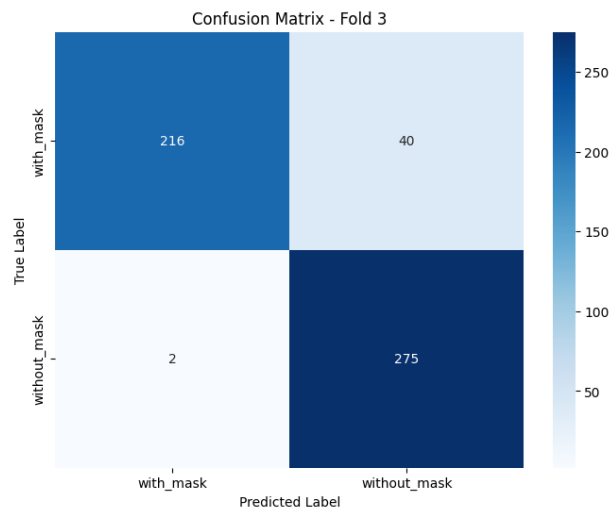
<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	1187.41	0.7467	0.7426	0.7863	0.7583
2	1205.36	0.8952	0.8950	0.8972	0.8950
3	1040.41	0.8341	0.8329	0.8415	0.8581
4	1039.93	0.9083	0.9069	0.9042	0.9158
5	1086.09	0.9474	0.9473	0.9502	0.9481
6	1012.70	0.9430	0.9428	0.9430	0.9471
7	1016.12	0.9430	0.9429	0.9449	0.9427
<i>Avg</i>	992.38	94.57	94.38	94.45	94.61

**Gambar 45.** *Confusion Matrix SGD 7 fold 50 epochs*

4.9.3.3 Hasil Skenario SGD 100 epochs

Tabel 31. Hasil SGD 3 fold 100 epochs

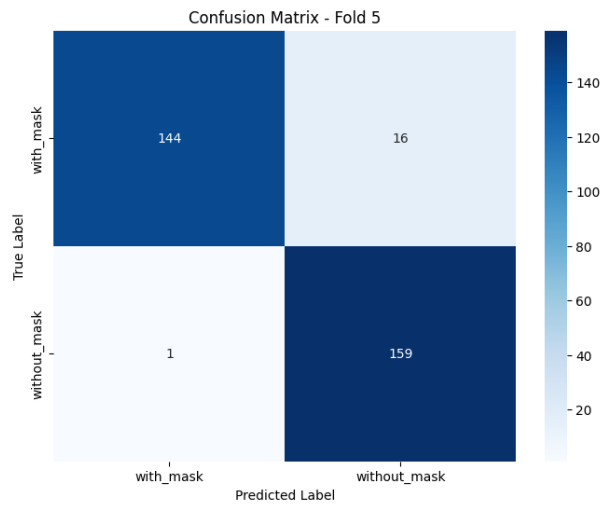
<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	1930.73	0.6948	0.6765	0.7924	0.7104
2	1914.06	0.9043	0.9035	0.9150	0.9030
3	1911.60	0.9212	0.9202	0.9319	0.9183
<i>Avg</i>	1918.13	83.88	82.49	85.57	83.57



Gambar 46. Confusion Matrix SGD 3 fold 100 epochs

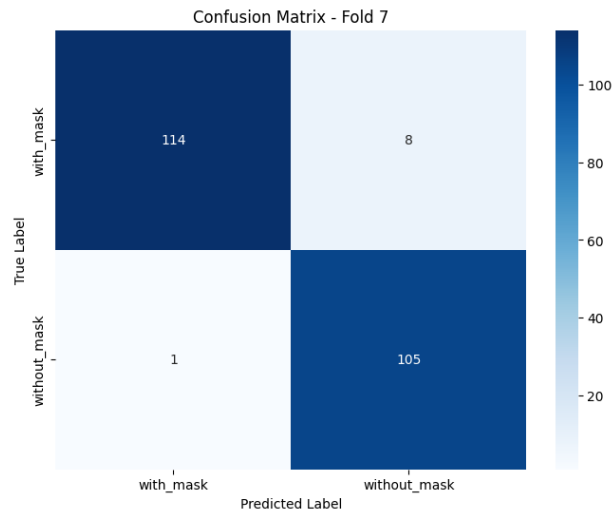
Tabel 32. Hasil *SGD 5 fold 100 epochs*

<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	1918.78	0.7750	0.7679	0.7881	0.7666
2	1918.57	0.9187	0.9178	0.9286	0.9161
3	1903.79	0.9000	0.8997	0.9135	0.9042
4	1946.69	0.9125	0.9124	0.9213	0.9176
5	1931.27	0.9469	0.9468	0.9508	0.9469
<i>Avg</i>	1919.57	92.08	91.63	92.18	91.42

**Gambar 47.** *Confusion Matrix SGD 5 Fold 100 epochs*

Tabel 33. Hasil *SGD* 7 fold 100 epochs

<i>Fold</i>	<i>Training Time (seconds)</i>	<i>Accuracy</i>	<i>F1 Score</i>	<i>Precision</i>	<i>Recall</i>
1	1948.19	0.8122	0.8108	0.8099	0.8142
2	1890.39	0.9039	0.9038	0.9100	0.9065
3	1918.21	0.9520	0.9516	0.9549	0.9501
4	1950.74	0.9345	0.9336	0.9425	0.9311
5	1942.36	0.9737	0.9737	0.9750	0.9737
6	1957.27	0.9211	0.9208	0.9217	0.9291
7	1944.66	0.9605	0.9605	0.9603	0.9625
<i>Avg</i>	1936.38	95.68	95.51	95.66	95.72

**Gambar 48.** *Confusion Matrix SGD* 7 fold 100 epochs

4.10 Hasil Rata-Rata Training

Tabel 34. Tabel Hasil Rata-Rata Training

<i>Optimizer</i>	<i>Epochs</i>	<i>Fold</i>	<i>Avg Training Time (s)</i>	<i>Avg Accuracy (%)</i>	<i>Avg F1 Score</i>	<i>Avg Precision</i>	<i>Avg Recall</i>
<i>Adam</i>	30	3	591.47	98.95	98.95	98.93	98.93
<i>Adam</i>	30	5	578.45	99.74	99.74	99.74	99.74
<i>Adam</i>	30	7	644.26	99.44	99.43	99.44	99.43
<i>Adam</i>	50	3	963.15	99.04	99.04	99.04	99.04
<i>Adam</i>	50	5	991.91	99.34	99.34	99.34	99.34
<i>Adam</i>	50	7	996.71	99.24	99.24	99.24	99.24
<i>Adam</i>	100	3	410.17	99.17	99.17	99.13	99.23
<i>Adam</i>	100	5	1962.92	99.17	99.16	99.17	99.16
<i>Adam</i>	100	7	1932.10	99.19	99.19	99.18	99.19
<i>RMSprop</i>	30	3	554.47	98.97	98.97	98.98	98.97
<i>RMSprop</i>	30	5	582.27	98.02	98.02	98.03	98.02
<i>RMSprop</i>	30	7	634.94	99.00	98.99	98.99	98.99
<i>RMSprop</i>	50	3	958.83	98.88	98.87	98.88	98.88
<i>RMSprop</i>	50	5	981.38	98.60	98.60	98.64	98.59
<i>RMSprop</i>	50	7	991.01	98.90	98.89	98.88	98.90
<i>RMSprop</i>	100	3	1846.94	99.18	99.17	99.18	99.17
<i>RMSprop</i>	100	5	1892.57	99.06	99.05	99.07	99.05
<i>RMSprop</i>	100	7	1917.11	99.18	99.17	99.17	99.17
<i>SGD</i>	30	3	593.97	69.83	69.62	70.60	69.57
<i>SGD</i>	30	5	583.70	61.63	60.85	68.64	65.63
<i>SGD</i>	30	7	627.56	91.68	91.37	91.84	91.25
<i>SGD</i>	50	3	892.41	65.27	64.80	70.77	67.94
<i>SGD</i>	50	5	976.98	87.43	87.20	89.03	87.76
<i>SGD</i>	50	7	992.38	94.57	94.38	94.45	94.61
<i>SGD</i>	100	3	1918.13	83.88	82.49	85.57	83.57
<i>SGD</i>	100	5	1919.57	92.08	91.63	92.18	91.42
<i>SGD</i>	100	7	1936.38	95.68	95.51	95.66	95.72

Berdasarkan hasil dari Tabel 34, Data ini menunjukkan hasil dari tiga jenis (*Adam*, *RMSprop*, dan *SGD*) dengan variasi jumlah *epochs* (30, 50, dan 100) dan jumlah *fold* (3, 5, dan 7). Metrik evaluasi yang digunakan adalah rata-rata waktu pelatihan, akurasi, F1 Score, presisi, dan recall.

1. **Adam**: Dengan jumlah *epoch* 30, *Adam* mencapai akurasi tertinggi sebesar 99.74% pada *fold* 5 dengan waktu pelatihan rata-rata 578.45 detik. Ketika jumlah *epoch* ditingkatkan menjadi 50 dan 100, akurasi tertinggi yang dicapai adalah 99.34% (*fold* 5, *epoch* 50, waktu pelatihan 991.91 detik) dan 99.19% (*fold* 7, *epoch* 100, waktu pelatihan 1932.10 detik).
2. **RMSprop**: Dengan jumlah *epoch* 30, *RMSprop* mencapai akurasi tertinggi sebesar 99.00% pada *fold* 7 dengan waktu pelatihan rata-rata 634.94 detik. Ketika jumlah *epoch* ditingkatkan menjadi 50 dan 100, akurasi tertinggi yang dicapai adalah 98.90% (*fold* 7, *epoch* 50, waktu pelatihan 991.01 detik) dan 99.18% (*fold* 3 dan 7, *epoch* 100, waktu pelatihan 1846.94 dan 1917.11 detik).
3. **SGD**: Dengan jumlah *epoch* 30, *SGD* mencapai akurasi tertinggi sebesar 91.68% pada *fold* 7 dengan waktu pelatihan rata-rata 627.56 detik. Ketika jumlah *epoch* ditingkatkan menjadi 50 dan 100, akurasi tertinggi yang dicapai adalah 94.57% (*fold* 7, *epoch* 50, waktu pelatihan 992.38 detik) dan 95.68% (*fold* 7, *epoch* 100, waktu pelatihan 1936.38 detik).

Dari data ini, dapat disimpulkan bahwa *Adam* dan *RMSprop* cenderung memberikan hasil yang lebih baik dibandingkan dengan *SGD*, terutama ketika jumlah *epoch* ditingkatkan.




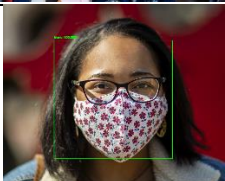
Untuk penelitian ini, berdasarkan data ini, akan menggunakan hasil dari *Adam* dengan 30 *epochs* dan 5 *folds*. Alasan utamanya adalah karena model ini menunjukkan performa terbaik dalam data yang telah dianalisis


dan memiliki waktu pelatihan yang relatif lebih pendek dibandingkan dengan konfigurasi lainnya.

4.11 Hasil Pengujian Data Tunggal

Pada bagian ini akan dilakukan pengujian data tunggal analisis metode *CNN* dengan arsitektur *MTCNN* pada objek orang menggunakan masker dan tidak menggunakan masker. Pengujian ini dilakukan dengan menggunakan 10 data citra, dimana masing-masing citra memiliki kelas yang berbeda-beda dan dibagi menjadi 5 menggunakan masker dan 5 tidak menggunakan masker.

Tabel 35. Sampel Pengujian Data Tunggal

No	Kelas	Citra	Output	Akurasi	Status Identifikasi
1	<i>Mask</i>		Mask	100%	Benar
2	<i>Mask</i>		Mask	100%	Benar
3	<i>Mask</i>		Mask	100%	Benar
4	<i>Mask</i>		Mask	100%	Benar

No	Kelas	Citra	Output	Akurasi	Status Identifikasi
5	Mask		Mask	0%	Salah
6	WithoutMask		No Mask	100%	Benar
7	WithoutMask		No Mask	100%	Benar
8	WithoutMask		No Mask	100%	Benar
9	WithoutMask		Mask	0%	Salah
10	WithoutMask		No Mask	100%	Benar
Hasil Rata-Rata				80%	

Dari hasil pengujian yang dilakukan menggunakan 10 citra yang dibagi masing-masing menjadi 5 dengan masker dan 5 tidak menggunakan masker, Dimana pada kelas *Mask* 4 terdeteksi dengan benar dan 1 salah mendeteksi. Sedangkan pada kelas *WithoutMask* berhasil mendeteksi 4 benar dan 1 salah terutama bagian orang yang menutup wajahnya menggunakan objek lain yang mengeluarkan *output Mask* yang pada kenyataannya orang tersebut tidak menggunakan masker.

BAB V

PENUTUP

Pada bab ini menjelaskan tentang kesimpulan dari hasil percobaan yang telah dilakukan. Bab ini juga berisikan saran perbaikan dalam penelitian yang akan datang.

5.1 KESIMPULAN

Hasil penelitian deteksi masker menggunakan metode *convolutional neural network* pada studi kasus covid-19 dapat disimpulkan :

1. Pengujian yang dilakukan dengan 1600 data citra yang terdiri dari 800 citra masker dan 800 citra tanpa masker menggunakan 3 *optimizer* (*Adam*, *RMSprop*, dan *SGD*), *Optimizer Adam* dan *RMSprop* menunjukkan performa yang lebih baik dibandingkan dengan *SGD* dalam hal akurasi, *f1-score*, *precision*, dan *recall*.
2. Hasil analisis yang telah dilakukan dalam penelitian ini, model *Convolutional Neural Network* menggunakan arsitektur *VGG16net* dengan *optimizer Adam* menghasilkan rata-rata akurasi 99.74% dengan proses *training 30 epochs, 5 fold*. *optimizer RMSprop* menghasilkan rata-rata akurasi 99.18%, 3 *fold*. sedangkan *optimizer SGD* menghasilkan akurasi rata-rata 95.68% pada proses training 100 *epochs, 7 fold*.
3. Dari tiga *optimizer* yang digunakan (*Adam*, *RMSprop*, dan *SGD*), *optimizer Adam* dan *RMSprop* menunjukkan performa yang lebih baik dibandingkan *SGD* dalam hal akurasi, *f1-score*, *precision*, dan *recall*.

5.2 SARAN

1. Untuk penelitian selanjutnya, disarankan menggunakan *optimizer Adam* atau *RMSprop* karena menunjukkan performa yang lebih baik dibandingkan dengan *SGD*
2. Mengingat peningkatan jumlah *epochs* tidak memberikan peningkatan performa yang signifikan untuk *optimizer Adam* dan *RMSprop*, disarankan untuk menggunakan *epochs* yang lebih rendah untuk menghemat waktu pelatihan.
3. Penelitian selanjutnya bisa menggunakan arsitektur CNN yang lain seperti *ResNet* atau *EfficientNet*.
4. Pada pengembangan selanjutnya model bisa di implementasi di perangkat bergerak seperti *smartphone* karena sudah menggunakan *Tensorflow Lite* untuk format model tersebut.

DAFTAR PUSTAKA

- Agarap, A. F. (2018). *Deep Learning using Rectified Linear Units (ReLU)*. <http://arxiv.org/abs/1803.08375>
- Basha, S. H. S., Dubey, S. R., Pulabaigari, V., & Mukherjee, S. (2019). Impact of Fully Connected Layers on Performance of Convolutional Neural Networks for Image Classification. *Neurocomputing*, 378, 112–119. <https://doi.org/10.1016/j.neucom.2019.10.008>
- Gholamalinezhad, H., & Khosravi, H. (2020). *Pooling Methods in Deep Neural Networks, a Review*. <https://arxiv.org/abs/2009.07485v1>
- Gonzalez, R. C. (1992). *Digital image processing / by Rafael C. Gonzalez, Richard E. Woods*. Addison-Wesley. <https://lib.ui.ac.id>
- Hao, X., Zhang, G., & Ma, S. (2016). Deep Learning. *International Journal of Semantic Computing*, 10(03), 417–439. <https://doi.org/10.1142/S1793351X16500045>
- Kingma, D. P., & Ba, J. L. (2014). Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. <https://arxiv.org/abs/1412.6980v9>
- Neapolitan, R. E., & Jiang, X. (2018). Neural Networks and Deep Learning. *Artificial Intelligence*, 389–411. <https://doi.org/10.1201/B22400-15>
- Needell, D., Srebro, N., & Ward, R. (2016). Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm. *Mathematical Programming*, 155(1–2), 549–573. <https://doi.org/10.1007/s10107-015-0864-7>
- Nugroho, B., Yulia, E., #2, P., & Syahrul Munir, M. (n.d.). *JEPIN (Jurnal Edukasi dan Penelitian Informatika) Kinerja Algoritma Optimasi Root-Mean-Square Propagation dan Stochastic Gradient Descent pada Klasifikasi Pneumonia Covid-19 Menggunakan CNN*.
- Nurfita, R. D., & Ariyanto, G. (2018). Implementasi Deep Learning berbasis Tensorflow untuk Pengenalan Sidik Jari. *Emitor: Jurnal Teknik Elektro*, 18(1), 22–27. <https://doi.org/10.23917/EMITOR.V18I01.6236>
- Pedoman Pencegahan dan Pengendalian CORONAVIRUS DISEASE (COVID-19) Revisi ke-5 - Protokol | Covid19.go.id*. (n.d.). Retrieved

May 5, 2023, from <https://covid19.go.id/p/protokol/pedoman-pencegahan-dan-pengendalian-coronavirus-disease-covid-19-revisi-ke-5>

Putra, W. E., & Putra, W. S. E. (2016). Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) pada Caltech 101. *Jurnal Teknik ITS*, 5(1). <https://doi.org/10.12962/j23373539.v5i1.15696>

RMSprop - Cornell University Computational Optimization Open Textbook - Optimization Wiki. (n.d.). Retrieved December 5, 2023, from <https://optimization.cbe.cornell.edu/index.php?title=RMSprop>

Simonyan, K., & Zisserman, A. (2014). *Very Deep Convolutional Networks for Large-Scale Image Recognition*.

Widodo, S., Brawijaya, H., & Samudi, S. (2022). Stratified K-fold cross validation optimization on machine learning for prediction. *Sinkron : Jurnal Dan Penelitian Teknik Informatika*, 7(4), 2407–2414. <https://doi.org/10.33395/SINKRON.V7I4.11792>

Yun, H. (2021). Prediction model of algal blooms using logistic regression and confusion matrix. *International Journal of Electrical and Computer Engineering (IJECE)*, 11(3), 2407. <https://doi.org/10.11591/ijece.v11i3.pp2407-2413>

Zufar, M., Setiyono, B., & Matematika, J. (2016). Convolutional Neural Networks Untuk Pengenalan Wajah Secara Real-time. *Jurnal Sains Dan Seni ITS*, 5(2), 128862. <https://www.neliti.com/id/publications/128862/>