**DETEKSI PENGGUNAAN MASKER MENGGUNAKAN METODE** ***CONVOLUTIONAL NEURAL NETWORK***

**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**

**2024**

**MASK DETECTION USING CONVOLUTIONAL NEURAL NETWORK METHOD**

**THESIS**

Presented as Partial Fulfillment of the Requirements

to Obtain a Bachelor's Degree in Computer Science

in the Informatics Study Program.

****

By:

George Jusen

185314072

**FACULTY OF SCIENCE AND TECHNOLOGY**

**SANATA DHARMA UNIVERSITY**

**YOGYAKARTA**

**2024**

# HALAMAN PERSETUJUAN

**SKRIPSI**

**DETEKSI PENGGUNAAN MASKER MENGGUNAKAN METODE *CONVOLUTIONAL NEURAL NETWORK***

Disusun oleh :

George Jusen

NIM : 185314072

Yogyakarta, …………………… 2024

Dosen Pembimbing,

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

# HALAMAN PENGESAHAN

**DETEKSI PENGGUNAAN MASKER MENGGUNAKAN METODE *CONVOLUTIONAL NEURAL NETWORK***

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 : Drs. Hari Suparwito, S.J., M.App.IT ………….….

Sekretaris : Ir. Robertus Adi Nugroho, M.Eng. ……………..

Anggota : Ir. Kartono Pinaryanto, S.T., M.Cs. ……………..

Yogyakarta, …………………… 2024

Fakultas Sains dan Teknologi

Universitas Sanata Dharma

Dekan,

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

# PERNYATAAN KEASLIAN KARYA

Saya menyatakan dengan sesungguhnya bahwa skripsi yang saya tulis ini tidak memuat karya atau bagian orang lain, kecuali yang telah disebutkan dalam kutipan dan daftar Pustaka dengan mengikuti ketentuan sebagaimana layaknya karya ilmiah.

Apabila dikemudian hari ditemukan indikasi plagiarisme dalam naskah ini, saya bersedia menanggung segala sanksi sesuai peraturan perundang – undang yang berlaku.

|  |
| --- |
| Yogyakarta, Januari 2023 |
| Penulis |
| George Jusen |

# LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPERLUAN AKADEMIS

Yang bertanda tangan dibawah ini, saya mahasiswa Universitas Sanata Dharma :

Nama : George Jusen

NIM : 185314072

Demi pengembangan ilmu pengetahuan, saya memberikan kepada Perpustakaan Universitas Sanata Dharma karya ilmiah saya berjudul :

**DETEKSI PENGGUNAAN MASKER MENGGUNAKAN METODE *CONVOLUTIONAL NEURAL NETWORK***

Beserta perangkat yang diperlukan (bila ada). Dengan demikian saya memberikan hak sepada Perpustakaan Universitas Sanata Dharma baik untuk menyimpan, menaglihkan dalam bentuk lain, mengolah dalam pangkalan data, mendistribusikan secara terbatas, dan mempublikasikannya di internet atau media lain untuk kepentingan akademis tanpa perlu meminta izin dari saya atau memberikan royalty kepada saya selama tetap mencantumkan nama saya sebagai penulis.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di Yogyakarta

Pada tanggal : Januari 2024

Yang Menyatakan,

George Jusen

# MOTTO

信じる力、未来への一歩。

(Shinjiru chikara, mirai e no ippo.)

"Kekuatan dalam Kepercayaan, Satu Langkah Menuju Masa Depan."

# KATA PENGANTAR

Puji Syukur kepada Tuhan Yesus Kristus yang senantiasa memberikan berkat dan Rahmat-Nya, sehingga penulis dapat menyelesaikan skripsi ini. Dalam perjalanan penyusunan skripsi ini penulis mendapat banyak dukungan dan bimbingan dari berbagai pihak. Sehingga kesempatan ini penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Tuhan Yesus Kristus, karena kebaikan dan berkatnya penulis dapat kuat menghadapi segala kesulitan dan maju terus sampai titik ini.
2. Kepada orang tua yang selalu menjadi motivasi penulis untuk segera menyelesaikan kuliah dan selalu ada mendukung segala Keputusan penulis
3. Kepada Bapak Ir. Kartono Pinaryanto, S.T., M.Cs. Selaku dosen pembimbing akademik penulis yang memberikan arahan yang baik untuk penyelesaian proses skripsi penulis.
4. Kepada dosen penguji Drs. Hari Suparwito, S.J., M.App.IT dan Ir. Robertus Adi Nugroho, M.Eng. yang telah memerikan saran dan koreksi yang membangun dalam penyempurnaan skripsi
5. Dan teman-teman seperjuangan saya yang tidak bisa saya sebutkan satu persatu Namanya

Penulis menyadari masih banyak kekurangan dan kesalahan yang terdapat dalam skripsi yang telah disusun. Oleh karena itu, penulis mengharapkan kritik dan saran yang membangun demi penyempurnaan skripsi ini. Penulis berharap skripsi ini dapat berguna bagi banyak orang.

Yogyakarta, Januari 2024

Penulis,

George Jusen

# DAFTAR ISI

[HALAMAN PERSETUJUAN iii](#_Toc156404082)

[HALAMAN PENGESAHAN iv](#_Toc156404083)

HALAMAN [PERNYATAAN KEASLIAN KARYA v](#_Toc156404084)

[HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI vi](#_Toc156404085)

HALAMAN [MOTTO vii](#_Toc156404086)

[KATA PENGANTAR viii](#_Toc156404087)

[DAFTAR ISI ix](#_Toc156404088)

[DAFTAR TABEL xiii](#_Toc156404089)

[DAFTAR GAMBAR xv](#_Toc156404090)

[ABSTRAK xvii](#_Toc156404091)

[ABSTRACK xviii](#_Toc156404092)

[BAB I PENDAHULUAN 1](#_Toc156404093)

[1.1 Latar Belakang 1](#_Toc156404094)

[1.2 Rumusan Masalah 6](#_Toc156404095)

[1.3 Batasan Masalah 6](#_Toc156404096)

[1.4 Tujuan dan Manfaat Penelitian 7](#_Toc156404097)

[1.5 Sistematika Penelitian 8](#_Toc156404098)

[BAB II LANDASAN TEORI 9](#_Toc156404099)

[2.1 Coronavirus Disease 2019 (COVID-19) 9](#_Toc156404100)

[2.1.1 Pencegahan dan Pengendalian COVID-19 9](#_Toc156404101)

[2.1.2 Pentingnya Penggunaan Masker 11](#_Toc156404102)

[2.2 Citra Digital 12](#_Toc156404103)

[2.3 Multi-Task Cascaded Convolutional Networks 12](#_Toc156404104)

[2.4 Sistem Pengenalan Wajah 14](#_Toc156404105)

[2.5 Deep Learning 14](#_Toc156404106)

[2.6 Convolutional Neural Network (CNN) 16](#_Toc156404107)

[2.6.1 Convolutional Layer 16](#_Toc156404108)

[2.6.2 *Rectifield Linear Unit* 17](#_Toc156404109)

[2.6.3 *Pooling Layer* 18](#_Toc156404110)

[2.6.4 *Fully Connected Layer* 19](#_Toc156404111)

[2.6.5 Arsitektur Jaringan *VGG16Net* 19](#_Toc156404112)

[2.7 *Adaptive Moment Estimation* 21](#_Toc156404113)

[2.8 *Root Mean Square Propagation* 21](#_Toc156404114)

[2.9 *Stochastic Gradient Descent* 22](#_Toc156404115)

[2.10 *Keras* 23](#_Toc156404116)

[2.11 *TensorFlow* 24](#_Toc156404117)

[2.12 *K-Fold Cross Validation* 24](#_Toc156404118)

[2.13 *Confusion Matrix* 25](#_Toc156404119)

[BAB III METODE PENELITIAN 27](#_Toc156404120)

[3.1 Deskripsi Data 27](#_Toc156404121)

[3.2 Kebutuhan Perangkat *Hardware* dan *Software* 37](#_Toc156404122)

[3.3 Perancangan Sistem 39](#_Toc156404123)

[3.3.1 *Preprocessing* 40](#_Toc156404124)

[3.3.2 *Data Augmentation* 41](#_Toc156404125)

[3.3.3 VGG16net 42](#_Toc156404126)

[3.3.4 *K-Fold Cross Validation* 42](#_Toc156404127)

[3.3.5 Latih Model CNN 44](#_Toc156404128)

[3.3.6 Evaluasi Model 44](#_Toc156404129)

[3.3.7 Model CNN 44](#_Toc156404130)

[3.4 Skenario Pengujian 44](#_Toc156404131)

[3.4.1 Skenario *K-Fold Cross Validation* 45](#_Toc156404132)

[3.4.2 Skenario Fungsi Pelatihan CNN 45](#_Toc156404133)

[3.4.3 Skenario Uji Data Tunggal 45](#_Toc156404134)

[BAB IV HASIL DAN ANALISIS 27](#_Toc156404135)

[4.1 Data 27](#_Toc156404136)

[4.1.1 *Preprocessing* 27](#_Toc156404137)

[4.1.2 Membuat Variabel Parameter 27](#_Toc156404138)

[4.1.3 Implementasi Memuat Data Citra 46](#_Toc156404139)

[4.1.4 *Data Augmentation* 47](#_Toc156404140)

[4.2 Membuat Model Jaringan CNN Dengan arsitektur *VGG16Net* 47](#_Toc156404141)

[4.3 Implementasi Tahap Pembuatan Model 48](#_Toc156404142)

[4.4 Implementasi *K-Fold Cross Validation* 50](#_Toc156404143)

[4.5 Implementasi Tahap Pelatihan Model 50](#_Toc156404144)

[4.6 Implementasi *Confusion Matrix* 51](#_Toc156404145)

[4.7 Implementasi menyimpan dan konversi model 52](#_Toc156404146)

[4.8 Implementasi Pengujian Model dengan *MTCNN* 53](#_Toc156404147)

[4.9 Hasil Pengujian Skenario Training 56](#_Toc156404148)

[4.9.1 Hasil *Adaptive Moment Estimation* 61](#_Toc156404149)

[4.9.2 Hasil *Root Mean Square Propagation* 81](#_Toc156404150)

[4.9.3 Hasil *Stochastic Gradient Descent* 99](#_Toc156404151)

[4.10 Hasil Rata-Rata Training 118](#_Toc156404152)

[4.11 Hasil Pengujian Data Tunggal 120](#_Toc156404153)

[BAB V PENUTUP 45](#_Toc156404154)

[5.1 KESIMPULAN 45](#_Toc156404155)

[5.2 SARAN 124](#_Toc156404156)

[DAFTAR PUSTAKA 125](#_Toc156404157)

# DAFTAR TABEL

[**Tabel 1.** *Confusion Matrix* 26](#_Toc156395983)

[**Tabel 2.** Parameter Skenario Pengujian 27](#_Toc156395984)

[**Tabel 3.** Parameter Untuk Pengujian Skenario 56](#_Toc156395985)

[**Tabel 4.** Parameter Data Augmentasi 57](#_Toc156395986)

[**Tabel 5.** Arsitektur Jaringan *VGG16Net* 57](#_Toc156395987)

[**Tabel 6.** Susunan Pengujian Model *Convolutional Neural Network .* 59](#_Toc156395988)

[**Tabel 7.** Hasil*Adam* 3 *fold* 30 *epoch*s 61](#_Toc156395989)

[**Tabel 8.** Hasil *Adam* 5 *fold* 30 *epoch*s 64](#_Toc156395990)

[**Tabel 9.** Hasil *Adam* 7 *Fold* 30 *Epoch*s 66](#_Toc156395991)

[**Tabel 10.** Hasil *Adam* 3 *Fold* 50 *Epoch*s 68](#_Toc156395992)

[**Tabel 11.** Hasil *Adam* 5 *fold* 50 *epoch*s 70](#_Toc156395993)

[**Tabel 12.** Hasil *Adam* 7 *fold* 50 *epoch*s 72](#_Toc156395994)

[**Tabel 13.** Hasil *Adam* 3 *fold* 100 *epoch*s 74](#_Toc156395995)

[**Tabel 14.** Hasil *Adam* 5 *fold* 100 *epoch*s 76](#_Toc156395996)

[**Tabel 15.** Hasil *Adam* 7 *fold* 100 *epoch*s 78](#_Toc156395997)

[**Tabel 16.** Hasil *RMSprop* 3 *fold* 30 *epoch*s 81](#_Toc156395998)

[**Tabel 17.** Hasil *RMSprop* 5 *fold* 30 *epoch*s 83](#_Toc156395999)

[**Tabel 18.** Hasil *RMSprop* 7 *fold* 30 *epoch*s 85](#_Toc156396000)

[**Tabel 19.** Hasil *RMSprop* 3 *fold* 50 *epoch*s 87](#_Toc156396001)

[**Tabel 20**. Hasil *RMSprop* 5 *fold* 50 *epoch*s 89](#_Toc156396002)

[**Tabel 21.** Hasil *RMSprop* 7 *fold* 50 *epoch*s 91](#_Toc156396003)

[**Tabel 22.** Hasil *RMSprop* 3 *fold* 100 *epoch*s 93](#_Toc156396004)

[**Tabel 23.** Hasil *RMSprop* 5 *fold* 100 *epoch*s 95](#_Toc156396005)

[**Tabel 24.** Hasil *RMSprop* 7 *fold* 100 *epoch*s 97](#_Toc156396006)

[**Tabel 25.** Hasil *SGD* 3 *fold* 30 *epoch*s 99](#_Toc156396007)

[**Tabel 26.** Hasil *SGD* 5 *fold* 30 *epoch*s 101](#_Toc156396008)

[**Tabel 27.** Hasil *SGD* 7 *fold* 30 *epoch*s 103](#_Toc156396009)

[**Tabel 28.** Hasil *SGD* 3 *fold* 50 *epoch*s 106](#_Toc156396010)

[**Tabel 29.** Hasil *SGD* 5 *fold* 50 *epoch*s 108](#_Toc156396011)

[**Tabel 30.** Hasil *SGD* 7 *fold* 50 *epoch*s 110](#_Toc156396012)

[**Tabel 31.** Hasil *SGD* 3 *fold* 100 *epoch*s 112](#_Toc156396013)

[**Tabel 32.** Hasil *SGD* 5 *fold* 100 *epoch*s 114](#_Toc156396014)

[**Tabel 33.** Hasil *SGD* 7 *fold* 100 *epoch*s 116](#_Toc156396015)

[**Tabel 34.** Tabel Hasil Rata-Rata Training 118](#_Toc156396016)

[**Tabel 35.** Sampel Pengujian Data Tunggal 120](#_Toc156396017)

# DAFTAR GAMBAR

[**Gambar 1**. Berita Covid-19 muncul pertama kali di Wuhan 2](#_Toc156395893)

[**Gambar 3.** Deteksi dengan *MTCNN* 14](#_Toc156395894)

[**Gambar 4**. Struktur arsitektur *deep learning* 15](#_Toc156395895)

[**Gambar 5.** Ilustrasi Arsitektur *Convolutional Neural Network* 16](#_Toc156395896)

[**Gambar 6.** Proses *convolutional layer* 17](#_Toc156395897)

[**Gambar 7.** Contoh Penerapan Aktivasi *ReLu* 18](#_Toc156395898)

[**Gambar 8.** Matriks *feature map* 4x4 dengan proses *pooling* 2x2 19](#_Toc156395899)

[**Gambar 9.** *Fully Connected Layer* 19](#_Toc156395900)

[**Gambar 10.** Arsitektur jaringan *VGG16Net* 20](#_Toc156395901)

[**Gambar 11.** Ilustrasi Proses 5-*fold Cross Validation* 25](#_Toc156395902)

[**Gambar 12.** Data penggunaan masker 37](#_Toc156395903)

[**Gambar 13.** Data penggunaan tanpa masker 37](#_Toc156395904)

[**Gambar 14.** Diagram Perancangan Sistem 39](#_Toc156395905)

[**Gambar 15.** *Code Blok* untuk memuat data citra. 46](#_Toc156395906)

[**Gambar 16.** Implementasi *Data Augmentation* 47](#_Toc156395907)

[**Gambar 17.** Membangun *base model VGG16Net* 47](#_Toc156395908)

[**Gambar 18.** *Code Blok* Pembuatan Model 48](#_Toc156395909)

[**Gambar 19.** Inisialisasi *K-Fold Cross Validation* 50](#_Toc156395910)

[**Gambar 20.** Tahap Pelatihan Model 51](#_Toc156395911)

[**Gambar 21.** Implementasi *Confusion Matrix* 52](#_Toc156395912)

[**Gambar 22.** Menyimpan dan konversi model 53](#_Toc156395913)

[**Gambar 23.** Hasil *Confusion Matrix* *Adam* 3 *Fold* 30 *epoch*s 62](#_Toc156395914)

[**Gambar 24.** Hasil Confusion Matrix *Adam* 5 *Fold* 30 *Epoch*s 64](#_Toc156395915)

[**Gambar 25.** Hasil Confusion Matrix *Adam* 7 *Fold* 30 *Epoch*s 66](#_Toc156395916)

[**Gambar 26.** *Confusion Matrix* *Adam* 3 *fold* 50 *Epoch*s 68](#_Toc156395917)

[**Gambar 27.** *Confusion Matrix* *Adam* 5 *Fold* 50 *epoch*s 70](#_Toc156395918)

[**Gambar 28.** *Confusion Matrix Adam* 7 *fold* 50 *epoch*s 72](#_Toc156395919)

[**Gambar 29**. *Confusion Matrix Adam* 3 *fold* 100 *epoch*s 74](#_Toc156395920)

[**Gambar 30.** *Confusion Matrix Adam* 5 *fold* 100 *epoch*s 76](#_Toc156395921)

[**Gambar 31.** *Confusion Matrix Adam* 7 *fold* 100 *epoch*s 78](#_Toc156395922)

[**Gambar 32.** *Confusion Matrix RMSprop* 3 *fold* 30 *epoch*s 81](#_Toc156395923)

[**Gambar 33.** *Confusion Matrix RMSprop* 5 *fold* 30 *Epoch*s 83](#_Toc156395924)

[**Gambar 34.** *Confusion Matrix RMSprop* 7 *fold* 30 *epoch*s 85](#_Toc156395925)

[**Gambar 35.** *Confusion Matrix RMSprop* 3 *fold* 50 *epoch*s 87](#_Toc156395926)

[**Gambar 36.** *Confusion Matrix RMSprop* 5 *fold* 50 *epoch*s 89](#_Toc156395927)

[**Gambar 37.** *Confusion Matrix RMSprop* 7 *fold* 50 *epoch*s 91](#_Toc156395928)

[**Gambar 38.** *Confusion Matrix RMSprop* 3 *fold* 100 *epoch*s 93](#_Toc156395929)

[**Gambar 39.** *Confusion Matrix RMSprop* 5 *fold* 100 *epoch*s 95](#_Toc156395930)

[**Gambar 40.** *Confusion Matrix RMSprop* 7 *fold* 100 *epoch*s 97](#_Toc156395931)

[**Gambar 41.** *Confusion Matrix SGD* 3 *fold* 30 *epoch*s 99](#_Toc156395932)

[**Gambar 42.** *Confusion Matrix SGD* 5 *fold* 30 *epoch*s 101](#_Toc156395933)

[**Gambar 43.** *Confusion Matrix SGD* 7 *fold* 30 *epoch*s 104](#_Toc156395934)

[**Gambar 44.** *Confusion Matrix SGD* 3 *fold* 50 *epoch*s 106](#_Toc156395935)

[**Gambar 45.** *Confusion Matrix SGD* 5 *fold* 50 *epoch*s 108](#_Toc156395936)

[**Gambar 46.** *Confusion Matrix SGD* 7 *fold* 50 *epoch*s 110](#_Toc156395937)

[**Gambar 47.** *Confusion Matrix SGD* 3 *fold* 100 *epoch*s 112](#_Toc156395938)

[**Gambar 48.** *Confusion Matrix SGD* 5 *Fold* 100 *epoch*s 114](#_Toc156395939)

[**Gambar 49.** *Confusion Matrix SGD* 7 *fold* 100 *epoch*s 116](#_Toc156395940)

# ABSTRAK

*Coronavirus Disease 2019 (Covid-19)* merupakan penyakit menular yang disebabkan oleh virus corona jenis baru, penyakit ini dapat menular dengan cepat secara droplet. Sehingga semua orang harus menjaga dan melindungi diri dengan berbagai macam protokol kesehatan, salah satunya adalah memakai masker ketika keluar rumah. Dalam kebiasaan baru penggunaan masker muka merupakan suatu kewajiban. Maka dari itu diperlukan sebuah *Machine Learning* untuk mendeteksi penggunaan masker muka agar orang-orang lebih disiplin dan peduli terhadap penggunaan masker muka.

Pendeteksi masker muka dibangun dengan menggunakan teknik deep learning. Dengan arsitektur yang digunakan yaitu *VGG16Net* serta menggunakan deteksi muka dengan *MTCNN*. Pelatihan dan pengujian model dilakukan pada infrastruktur *Google Colaboratory*.

Melalui penelitian yang telah dilakukan, diperoleh bahwa arsitektur model *CNN* yaitu *VGG16Net* dengan metode optimizer Adam K-Fold 5, epoch 30 lebih unggul dari optimizer SGD dan RMSprop, dikarenakan mencapai nilai akurasi optimal 99,74% dan hasil dari uji data tunggal berhasil menghasilkan 80% akurasi. Dengan demikian hal tersebut menunjukan bahwa deteksi masker menggunakan CNN pada citra masker berhasil dan baik digunakan.

**Kata Kunci:** deteksi masker muka, *deep learning*, *VGG16Ne*t, *CNN*, *MTCNN*, *Google Colaboratory*.

# ABSTRACK

Coronavirus Disease 2019 (Covid-19) is an infectious disease caused by a novel coronavirus, which can rapidly spread through droplets. Consequently, everyone must take precautions and protect themselves by adhering to various health protocols, one of which is wearing a mask when leaving home. Wearing a face mask has become a new norm and is considered a mandatory practice. Therefore, the implementation of Machine Learning is essential to detect the usage of face masks, encouraging people to be more disciplined and conscientious about wearing them.

The face mask detector is constructed using deep learning techniques, employing the VGG16Net architecture and facial detection with MTCNN. Model training and testing are conducted on the Google Colaboratory infrastructure.

Through the conducted research, it was found that the Convolutional Neural Network (CNN) model architecture, specifically VGG16Net, using the Adam optimizer with K-Fold 5 and 30 epochs, outperformed the SGD and RMSprop optimizers. This was evidenced by achieving an optimal accuracy of 99.74%, and single test data yielded an accuracy of 80%. Consequently, this indicates that mask detection using CNN on mask images is effective and suitable for use.

**Keywords:** face mask detection, deep learning, VGG16Net, CNN, MTCNN, Google Colaboratory.

# PENDAHULUAN

## 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)

Penggunaan masker merupakan bagian dari rangkaian komprehensif langkah pencegahan dan pengendalian yang dapat membatasi penyebaran penyakit – penyakit virus saluran pernapasan tertentu, termasuk COVID-19. Masker dapat digunakan baik untuk melindungi orang yang sehat (dipakai untuk melindungi diri sendiri saat berkontak dengan orang yang terinfeksi) atau untuk mengendalikan sumber (dipakai oleh orang yang terinfeksi untuk mencegah penularan lebih lanjut).

Adanya protokol kesehatan yang ditetapkan oleh pemerintah membuat tugas baru untuk petugas keamanan fasilitas umum dalam memeriksa apakah masyarakat memakai masker dengan benar. Permasalahan yang didapat adalah pengenalan objek pada video yang digunakan untuk pengenalan masker.

Berkaitan dengan sistem keamanan untuk orang yang tidak memakai masker, fasilitas umum biasanya membutuhkan bantuan nasihat dari petugas keamanan fasilitas umum. Otomatisasi sangat diinginkan untuk memantau orang yang tidak memakai masker dan juga mengurangi jumlah manusia sebagai sumber daya yang dibutuhkan. Solusi yang tepat untuk menghadapi masalah ini adalah menggunakan *object detection* pada video kamera pengintai di tempat umum.

Dalam konteks tingginya tingkat polusi udara di Jakarta, penggunaan masker menjadi langkah kritis untuk melindungi sistem pernapasan masyarakat. Masker tidak hanya berfungsi sebagai penghalang fisik terhadap partikel-partikel berbahaya dalam udara, tetapi juga membantu mengurangi risiko terpapar zat-zat polutan yang dapat menyebabkan penyakit pernapasan dan masalah kesehatan lainnya. Dengan memperkuat kesadaran akan pentingnya menggunakan masker di tengah polusi udara yang tinggi, masyarakat Jakarta dapat lebih efektif menjaga kesehatan pernapasan mereka dan mengurangi dampak negatif yang mungkin timbul akibat paparan polusi udara yang berkelanjutan.

*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 secata otomatis melalui pembelajaran dari data yang diberikan. Pada *Machine Learning* terdapat tiga tipe yaitu *Supervised Learning, Unsupervised Learing* 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)” terdapat beberapa penelitian terdahulu yang berhubungan mengenai membahas klasifikasi obyek dengan citra menggunakan *machine learning.* Penelitian tersebut dapat diliat sebagai berikut:

* 1. 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%.

1. 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.
2. 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.

## Rumusan Masalah

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

1. Bagaimana performa model CNN dalam mengidentifikasi penggunaan masker pada kondisi waktu nyata, terutama dalam hal ketepatan (akurasi)?
2. Sejauh mana efektivitas algoritma *Convolutional Neural Network* dalam mendeteksi masker dalam beberapa jenis masker?

## Batasan Masalah

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

1. Metode *Deep Learning* yang digunakan untuk pendeteksi masker terdiri dari satu komponen yaitu pendeteksi muka menggunakan *MTCNN*, serta algoritma klasifikasi wajah dengan *VGG16Net*
2. Program dibuat untuk membedakan pengguna menggunakan masker atau tidak menggunakan masker dan bersifat tunggal.
3. Program pendeteksi masker menggunakan bahasa pemrograman *Python* dengan *Platform Google Colaboratory*.
4. Membangun model *Deep Learning* citra pendeteksi masker menggunakan *Framework* *Keras* dan *Tensorflow.*
5. Data masker yang digunakan adalah masker kain, masker bedah dan masker N95.

## Tujuan dan Manfaat Penelitian

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

Berapa besar akurasi yang dihasilkan dari prediksi menggunakan *Convolutional Neural Network* dengan arsitektur VGG16Net.

Seberapa baik model tersebut untuk mendeteksi seseorang menggunakan masker dan tidak menggunakan masker *dengan face detection* MTCNN.

Manfaat yang diperoleh dari penelitian ini adalah:

1. Bagi peneliti:

Mampu menerapkan ilmu yang didapatkan pada bangku perkuliahan untuk menyelesaikan tugas akhir.

1. Bagi masyarakat:

Membantu mempercepat masyarakat dalam mendeteksi penggunaan masker di dalam suatu instansi seperti rumah sakit dan lain sebagainya

1. Bagi Universitas:

Menambang kontribusi dalam ilmu informatika serta dapat menjadi referensi bagi mahasiswa lain yang sedang melakukan penelitian mengenai *Convolutional Neural Network*.

## Sistematika Penelitian

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

1. **BAB I PENDAHULUAN**

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

1. **BAB II DASAR TEORI**

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

1. **BAB III METODE PENELITIAN**

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

1. **BAB IV HASIL DAN PEMBAHASAN**

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

1. **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.

# 

# LANDASAN TEORI

## 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.)

### 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 :

1. 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.
2. 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*.
3. Menjaga jarak minimal 1 meter dengan orang lain agar terhindar terkena *droplet* dari orang-orang yang batuk atau bersin.
4. Menggunakan masker kain bila berada di luar rumah, setelah 4 jam dipakai, dan cuci hingga bersih setelah dipakai.
5. Disarankan menggunakan masker jika ingin berobat ke fasyankes.

### 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 :

1. 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*.

1. 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.

1. 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.

## 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)* < ∞

Citra digital juga bisa di representasikan dalam sebuah *matrix* M x 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.

## 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.(Zhang et al., 2016)



**Gambar 1.** Deteksi dengan *MTCNN*

Sumber : (Canedo & Neves, 2019)

## 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. (Teoh et al., 2021)

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

## *Supervised Learning*

*Supervised learning* adalah paradigma dalam *machine learning* di mana model dilatih untuk melakukan prediksi atau klasifikasi berdasarkan *input* dan *output* yang sudah diberikan sebelumnya. Dalam konteks ini, model belajar dari contoh-contoh yang sudah memiliki label atau kategori yang diketahui. Tujuannya adalah untuk membuat model yang dapat memberikan prediksi yang akurat untuk data baru yang belum pernah dilihat sebelumnya.(Santoso et al., 2021)

Proses *supervised learning* umumnya melibatkan langkah-langkah berikut:

1. Pengumpulan Data Latih:

Mengumpulkan dataset yang terdiri dari pasangan input dan output yang sudah dilabel. Contohnya, dalam kasus penelitian Anda, dataset dapat berisi gambar wajah orang dengan label apakah mereka menggunakan masker atau tidak.

1. Pembagian Data:

Dataset dibagi menjadi dua bagian utama: data latih (train) dan data uji (test). Data latih digunakan untuk melatih model, sedangkan data uji digunakan untuk menguji performa model di luar data yang sudah dilihat.

1. Pemilihan Model:

Memilih jenis model yang akan digunakan untuk tugas klasifikasi. Dalam kasus deteksi masker, Convolutional Neural Network (CNN) adalah pilihan umum karena kemampuannya dalam menangani data gambar.

1. Pelatihan Model:

Model dilatih menggunakan data latih. Proses pelatihan ini melibatkan penyesuaian parameter model sedemikian rupa sehingga kesalahan prediksi terkecil. Model memperbarui bobotnya berdasarkan perbandingan antara prediksi dan label yang sebenarnya.

1. Evaluasi dan Validasi:

Model diuji pada data uji yang belum pernah dilihat sebelumnya untuk mengukur sejauh mana kemampuannya dalam melakukan prediksi yang akurat. Pengukuran performa model, seperti akurasi, *presisi*, *recall*, dan *F1-score*, dapat digunakan untuk mengevaluasi keefektifannya.

## *Deep Learning*

*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 2**. Struktur arsitektur *deep learning*

Sumber : (Çakir & Çınarer, n.d.)

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:

1. *Training Set*

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

1. *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.

1. *Testing set*

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

## *Data Augmentation*

*Data Augmentation* adalah teknik dalam *machine learning* yang digunakan untuk menambah jumlah data dengan cara memodifikasi data yang sudah ada. Modifikasi ini dapat dilakukan dengan berbagai cara, seperti mengubah ukuran, *rotasi, cropping,* dan penambahan *noise.*(Perez & Wang, 2017)

Secara ilmiah, data augmentasi dapat meningkatkan kinerja model machine learning dengan cara:

1. Mencegah *overfitting. Overfitting* adalah kondisi di mana model *machine learning* terlalu cocok dengan data pelatihan, sehingga tidak dapat generalisasi ke data baru. Data augmentasi dapat membantu mencegah *overfitting* dengan menambahkan data baru yang berbeda dari data pelatihan.
2. Meningkatkan akurasi. *Data Augmentation* dapat membantu meningkatkan akurasi model *machine learning* dengan memberikan lebih banyak contoh untuk model *machine learning* tersebut belajar.
3. Meningkatkan interpretabilitas. *Data Augmentation* dapat membantu meningkatkan interpretabilitas model machine learning dengan memberikan lebih banyak informasi tentang bagaimana model bekerja.

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



**Gambar 3.** Ilustrasi Arsitektur *Convolutional Neural Network*

Sumber : (Kumar et al., 2020)

### *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 4, yang merupakan contoh dari layer konvolusi.

A diagram of a mathematical equation

Description automatically generated

**Gambar 4.** Proses *convolutional layer*

Sumber : (Dhanny et al., n.d.)

### *Rectifield Linear Unit*

Aktivasi ReLu (*Rectifield 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 5.** Contoh Penerapan Aktivasi *ReLu*

Sumber : (Sheta & Byers, n.d.)

### *Softmax*

Aktivasi *Softmax*, yang juga dikenal sebagai “*soft maximum*”, adalah fungsi matematika yang mengubah vektor angka nyata menjadi distribusi probabilitas. Fungsi ini mengubah angka atau logit menjadi probabilitas. Output dari softmax adalah vektor dengan probabilitas untuk setiap kemungkinan hasil atau kelas. Probabilitas dalam vektor ini berjumlah satu untuk semua kemungkinan hasil atau kelas.(Shen et al., 2023)

Secara matematis, *softmax* didefinisikan sebagai berikut:

di mana adalah elemen ke-i dari vektor input, dan adalah jumlah dari semua nilai eksponensial dari elemen-elemen vektor input. Illustrasi penggunaan *softmax* dapat dilihat pada gambar berikut.



Gambar 6. Ilustrasi Output Layer

Sumber : (Jurjawi, 2020)

### *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

Sumber : (El-Moneim et al., 2021)

### *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*

Sumber : (Meruva et al., 2022)

### 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*

Sumber : (Khattar & Quadri, 2023)

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)

## *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)

## *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* (RMSProp). 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* (RMSProp), 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.)

## *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.

## *Framework TensorFlow* dan *Keras*

Terdapat berbagai macam kerangka kerja yang dapat digunakan untuk *deep learning,* di antaranya adalah *TensorFlow* dan *Keras*, yang merupakan dua kerangka kerja yang paling sering digunakan dalam bidang ini.

*TensorFlow* adalah platform *open-source* yang dirancang untuk *deep learning.* Ini adalah perpustakaan dengan API tingkat tinggi yang memungkinkan pembuatan jaringan saraf dalam skala besar. Di sisi lain, *Keras* adalah kerangka kerja *deep learning* untuk *Python* yang menyediakan cara yang mudah untuk mendefinisikan dan melatih hampir semua jenis model *deep learning.* *Keras* awalnya dikembangkan dengan tujuan untuk mempercepat eksperimen.(Zollanvari, 2023)

Berikut adalah beberapa fitur utama dari Keras:

Keras memungkinkan kode untuk berjalan baik di CPU maupun GPU.

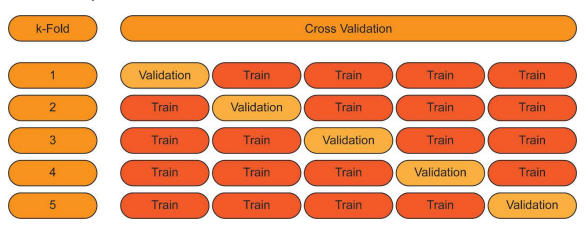
Keras memiliki API yang ramah pengguna yang memudahkan pembuatan prototipe model *deep learning* dengan cepat.

Keras mendukung *convolutional networks, recurrent network,* dan kombinasi keduanya.

Keras mendukung berbagai arsitektur jaringan, termasuk model *multi-input* atau *multi-output*, berbagai lapisan, dan berbagai model.

## *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*

Sumber : (FUADAH et al., 2022)

## *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 :

* + - * 1. *True Positive (tp)*
        2. *False Positive (fp)*
        3. *True Negative (tn)*
        4. *False Negative (tn)*

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

**Tabel 1.** *Confusion Matrix*

|  |  |  |
| --- | --- | --- |
| kelas | *withmask* | *withoutmask* |
| *withmask* | *True Positive (tp)* | *false positive (fp)* |
| *withoutmask* | *false negative (fn)* | *true negative (tn)* |

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

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

*Presisi* adalah metrik evaluasi yang mengukur seberapa baik model membuat prediksi yang benar untuk kelas positif dari total prediksi positif yang dilakukan.

*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:

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

# METODE PENELITIAN

## Deskripsi Data

Dataset yang digunakan dalam penelitian ini adalah citra masker wajah yang diperoleh dari Kaggle [https://www.kaggle.com/*dataset*s/omkargurav/face-mask-*dataset*](https://www.kaggle.com/datasets/omkargurav/face-mask-dataset) (Omkargurav, 2023) di akses pada tanggal 5 Juni 2023. Data ini memiliki jumlah 5777 data citra yang terdiri dari 3725 tanpa masker dan 3828 menggunakan masker. Karena *dataset* citra masker wajah memiliki ukuran citra yang berbeda, maka ukuran citra diubah menjadi 224x224 piksel.



**Gambar 11.** Data penggunaan masker

****

**Gambar 12.** Data penggunaan tanpa masker

## Kebutuhan Perangkat *Hardware* dan *Software*

1. Spesifikasi Hardware
   1. Processor AMD Ryzen 5 3550H 4 Core 8 Threads
   2. RAM 16 GB DDR4 Dual-Channel 2400 Mhz
   3. Nvidia GTX 1050 3GB
2. Spesifikasi Software
   1. Sistem Operasi Windows 10 Home 64—bit
   2. Google Collab

## Perancangan Sistem

A screenshot of a computer screen

Description automatically generated

**Gambar 13.** Diagram Perancangan Sistem

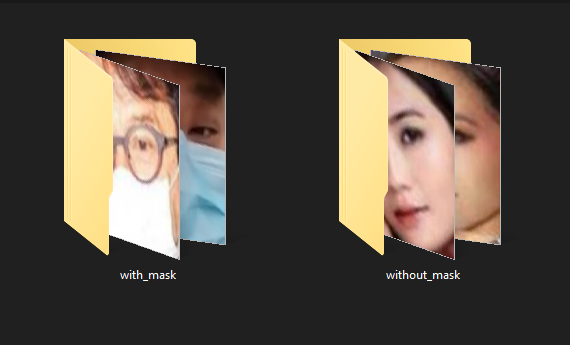
Berdasarkan Gambar 13 diagram perancangan sistem dibagi menjadi 6 tahap yaitu *Preprocessing, Data Augmentation,* arsitektur *VGG16net*, *K-Fold Cross Validation*, Latih Model CNN, dan evaluasi model CNN. Berikut penjelasan masing-masing tahapan tersebut.

## Pengambilan Data

*Dataset* tersebut memiliki jumlah 5777 data citra yang terdiri dari 3725 tanpa masker dan 3828 menggunakan masker. Maka penelitian ini hanya mengambil 1600 data citra yang terdiri dari 800 citra masker dan 800 citra tanpa masker untuk mengoptimalkan proses pelatihan model. Keputusan ini diambil dengan pertimbangan untuk mengatasi keterbatasan sumber daya *hardware*, mempercepat waktu pelatihan, dan tetap mempertahankan representasi yang cukup dari variasi situasi yang mungkin ditemui dalam penggunaan masker di dunia nyata. Dengan demikian, pengambilan sampel ini diharapkan dapat memberikan hasil pelatihan yang efisien tanpa mengorbankan kualitas dan generalisasi model terhadap situasi umum.

## Labeling Data

Dalam *dataset* tersebut sudah tersedia 2 folder yang bernama *with\_mask* dan *without\_mask* seperti Gambar 14 sebagai berikut :



**Gambar 14**. Isi direktori dari dataset

Direktori tersebut memiliki dua folder bernama "*with\_mask*" dan "*without\_mask*," kita dapat menentukan bahwa label kelas untuk gambar-gambar tersebut adalah "*with\_mask*" (dengan masker) dan "*without\_mask*" (tanpa masker).

A computer screen shot of a program

Description automatically generated

**Gambar 15.** Proses Labeling *Dataset*

Berdasarkan Gambar 15. Code tersebut bertanggung jawab untuk memproses *dataset* gambar yang terletak dalam direktori "/content/Skripsi/dataset\_small". Dalam proses tersebut, setiap gambar dipetakan ke dalam list data, sedangkan label kelas (masker atau tanpa masker) diekstrak dari nama direktori tempat gambar tersebut disimpan (*with\_mask* dan *without\_mask*) dan dimasukkan ke dalam list labels.

Setelah eksekusi *loop* selesai, *list* data berisi representasi numerik dari gambar-gambar dalam bentuk array, dan *list labels* berisi label kelas yang sesuai dengan setiap gambar. Data ini dapat digunakan sebagai *input* dalam melatih model *machine learning.*

### *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.

#### *Rezise*

*Rezise* merupakan proses untuk mengubah ukuran citra menjadi dimensensi 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.

#### 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*”

#### 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].

### *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*.

### 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.

### *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*.*

### 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).

### 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 dam F1-Score.*

### Model CNN

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

## 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)

### 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.

### 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 *epoch*s sebesar 30, 50, dan 100 iterasi.

### 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.

**Tabel 2.** Rencana Eksperimen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No. | K-Fold Cross Validation | Optimizer | Learning Rate | Batch Size | Epochs |
| 1 | 3-fold | Adam | 0.0001 | 32 | 30 |
| 2 | 3-fold | Adam | 0.0001 | 32 | 50 |
| 3 | 3-fold | Adam | 0.0001 | 32 | 100 |
| 4 | 5-fold | Adam | 0.0001 | 32 | 30 |
| 5 | 5-fold | Adam | 0.0001 | 32 | 50 |
| 6 | 5-fold | Adam | 0.0001 | 32 | 100 |
| 7 | 7-fold | Adam | 0.0001 | 32 | 30 |
| 8 | 7-fold | Adam | 0.0001 | 32 | 50 |
| 9 | 7-fold | Adam | 0.0001 | 32 | 100 |
| 10 | 3-fold | SGD | 0.0001 | 32 | 30 |
| 11 | 3-fold | SGD | 0.0001 | 32 | 50 |
| 12 | 3-fold | SGD | 0.0001 | 32 | 100 |
| 13 | 5-fold | SGD | 0.0001 | 32 | 30 |
| 14 | 5-fold | SGD | 0.0001 | 32 | 50 |
| 15 | 5-fold | SGD | 0.0001 | 32 | 100 |
| 16 | 7-fold | SGD | 0.0001 | 32 | 30 |
| 17 | 7-fold | SGD | 0.0001 | 32 | 50 |
| 18 | 7-fold | SGD | 0.0001 | 32 | 100 |
| 19 | 3-fold | RMSprop | 0.0001 | 32 | 30 |
| 20 | 3-fold | RMSprop | 0.0001 | 32 | 50 |
| 21 | 3-fold | RMSprop | 0.0001 | 32 | 100 |
| 22 | 5-fold | RMSprop | 0.0001 | 32 | 30 |
| 23 | 5-fold | RMSprop | 0.0001 | 32 | 50 |
| 24 | 5-fold | RMSprop | 0.0001 | 32 | 100 |
| 25 | 7-fold | RMSprop | 0.0001 | 32 | 30 |
| 26 | 7-fold | RMSprop | 0.0001 | 32 | 50 |
| 27 | 7-fold | RMSprop | 0.0001 | 32 | 100 |

# HASIL DAN ANALISIS

## Data

Data yang digunakan adalah data yang diperoleh dari dari laman [https://www.kaggle.com/*dataset*s/omkargurav/face-mask-*dataset*](https://www.kaggle.com/datasets/omkargurav/face-mask-dataset). Data tersebut memiliki dua kelas yaitu with\_mask dan without mask untuk melatih *machine learning*

### *Preprocessing*

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

### Membuat Variabel Parameter

Pada tahap ini akan mendefinisikan variabel konstanta sebagai berikut :

**Tabel 3.** Parameter Skenario Pengujian

|  |  |  |
| --- | --- | --- |
| No. | Parameter | Value |
| 1 | *Epoch* | 30,50,100 |
| 2 | *Target Size* | 224 x 224 |
| 3 | *Batch Size* | 32 |
| 4 | *Learning Rate* | 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.

### Implementasi Memuat Data Citra

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

A screen shot of a computer program

Description automatically generated

**Gambar 16.** *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.

### *Data Augmentation*

A screen shot of a computer program

Description automatically generated

**Gambar 17.** 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.

## Membuat Model Jaringan CNN Dengan arsitektur *VGG16Net*

**Sebuah gambar berisi teks, cuplikan layar, Font

Deskripsi dibuat secara otomatis**

**Gambar 18.** 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).

## Implementasi Tahap Pembuatan Model

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

Sebuah gambar berisi teks, cuplikan layar, software, Font

Deskripsi dibuat secara otomatis

**Gambar 19.** *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.

## Implementasi *K-Fold Cross Validation*

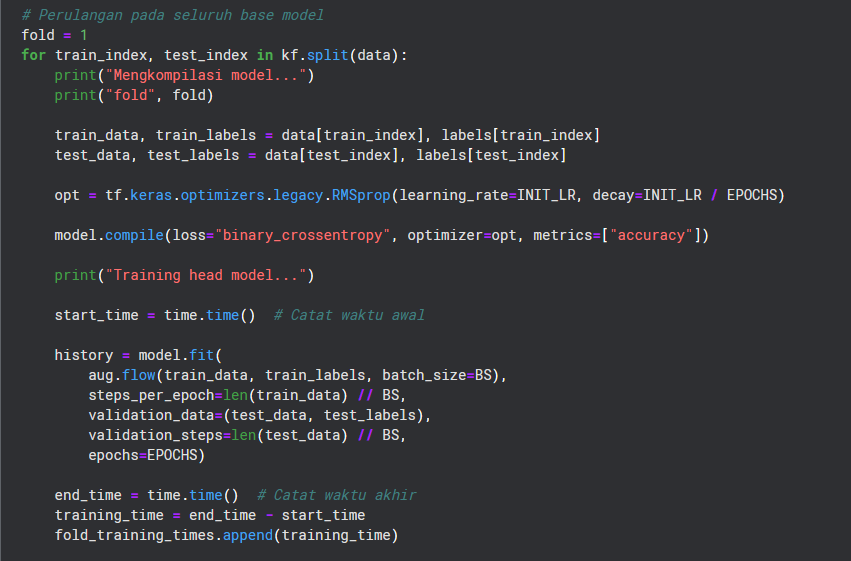
A screen shot of a computer

Description automatically generated

**Gambar 20.** 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 (*fold*s) sebanyak *n\_fold* dan mengaktifkan pengacakan data.

## Implementasi Tahap Pelatihan Model



**Gambar 21.** 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.

## Implementasi *Confusion Matrix*

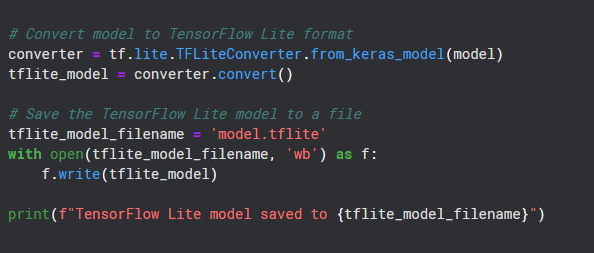
A screen shot of a computer program

Description automatically generated

**Gambar 22.** 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.*

## Implementasi menyimpan dan konversi model



**Gambar 23.** 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.

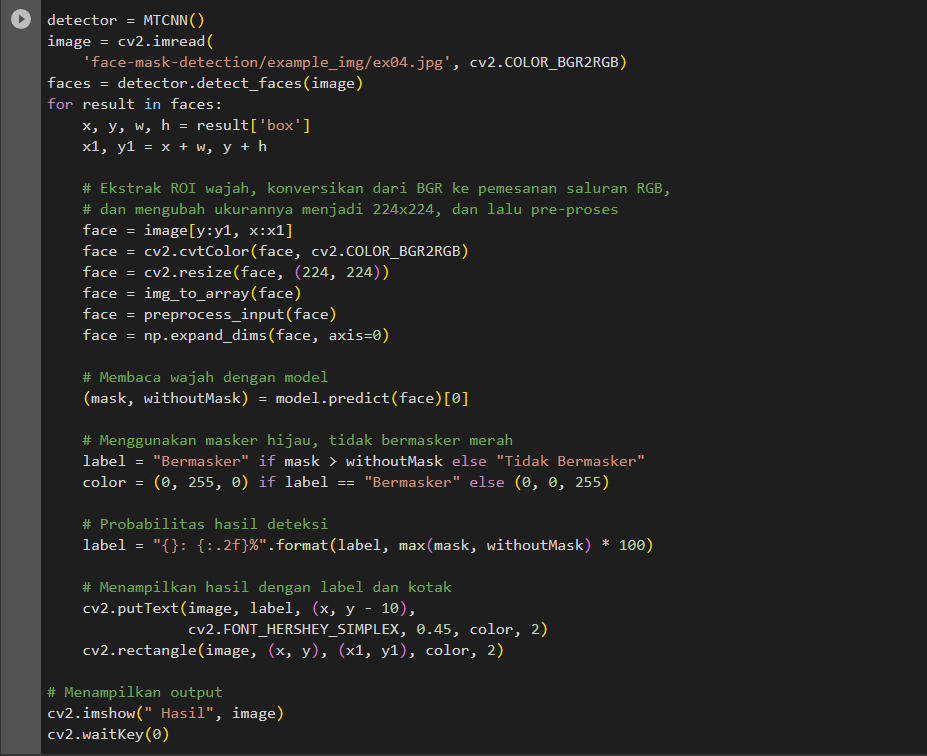
## Implementasi Pengujian Model dengan *MTCNN*

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 ("*Mask*" atau "*No Mask*") 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.



**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.

## Hasil Pengujian Skenario Training

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

**Tabel 4.** Parameter Untuk Pengujian Skenario

|  |  |  |
| --- | --- | --- |
| No. | Parameter | Value |
| 1 | *Epoch* | 30, 50, dan 100 |
| 2 | *Target Size* | 224 x 224 RGB |
| 3 | *Batch Size* | 32 |
| 4 | *Learning Rate* | 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 5.** Parameter Data Augmentasi

|  |  |
| --- | --- |
| Parameter | Penjelasan |
| *rotation\_range=20* | Rentang rotasi dalam derajat untuk memutar Gambar. |
| *zoom\_range=0.15* | Rentang level zoom-in dan zoom-out pada Gambar. |
| *width\_shift\_range=0.2* | Rentang pergeseran horizontal Gambar. |
| *height\_shift\_range=0.2* | Rentang pergeseran vertikal Gambar. |
| *shear\_range=0.15* | Rentang pergeseran sudut pemotongan. |
| *horizontal\_flip=True* | Kemungkinan untuk memutar Gambar secara horizontal. |
| *fill\_mode="nearest"* | Cara mengisi piksel yang kosong setelah augmentasi (dalam hal ini, menggunakan piksel terdekat). |

**Tabel 6.** Arsitektur Jaringan *VGG16Net*

| Layer (type) | Output Shape | Param |
| --- | --- | --- |
| *input\_1 (InputLayer)* | (None, 224, 224, 3) | 0 |
| *block1\_conv1 (Conv2D)* | (None, 224, 224, 64) | 1792 |
| *block1\_conv2 (Conv2D)* | (None, 224, 224, 64) | 36928 |
| *block1\_pool (MaxPooling2D)* | (None, 112, 112, 64) | 0 |
| *block2\_conv1 (Conv2D)* | (None, 112, 112, 128) | 73856 |
| *block2\_conv2 (Conv2D)* | (None, 112, 112, 128) | 147584 |
| *block2\_pool (MaxPooling2D)* | (None, 56, 56, 128) | 0 |
| *block3\_conv1 (Conv2D)* | (None, 56, 56, 256) | 295168 |
| *block3\_conv2 (Conv2D)* | (None, 56, 56, 256) | 590080 |
| *block3\_conv3 (Conv2D)* | (None, 56, 56, 256) | 590080 |
| *block3\_pool (MaxPooling2D)* | (None, 28, 28, 256) | 0 |
| *block4\_conv1 (Conv2D)* | (None, 28, 28, 512) | 1180160 |
| *block4\_conv2 (Conv2D)* | (None, 28, 28, 512) | 2359808 |
| *block4\_conv3 (Conv2D)* | (None, 28, 28, 512) | 2359808 |
| *block4\_pool (MaxPooling2D)* | (None, 14, 14, 512) | 0 |
| *block5\_conv1 (Conv2D)* | (None, 14, 14, 512) | 2359808 |
| *block5\_conv2 (Conv2D)* | (None, 14, 14, 512) | 2359808 |
| *block5\_conv3 (Conv2D)* | (None, 14, 14, 512) | 2359808 |
| *block5\_pool (MaxPooling2D)* | (None, 7, 7, 512) | 0 |
| *Total params:* |  | 14,714,688 |
| *Trainable params:* |  | 0 |
| *Non-trainable params:* |  | 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 7.** Susunan Pengujian Model *Convolutional Neural Network .*

| Layer (type) | | Output Shape | Param |
| --- | --- | --- | --- |
| *input\_1 (InputLayer)* | (None, 224, 224, 3) | | 0 |
| *block1\_conv1 (Conv2D)* | (None, 224, 224, 64) | | 1792 |
| *block1\_conv2 (Conv2D)* | (None, 224, 224, 64) | | 36928 |
| *block1\_pool (MaxPooling2D)* | (None, 112, 112, 64) | | 0 |
| *block2\_conv1 (Conv2D)* | (None, 112, 112, 128) | | 73856 |
| *block2\_conv2 (Conv2D)* | (None, 112, 112, 128) | | 147584 |
| *block2\_pool (MaxPooling2D)* | (None, 56, 56, 128) | | 0 |
| *block3\_conv1 (Conv2D)* | (None, 56, 56, 256) | | 295168 |
| *block3\_conv2 (Conv2D)* | (None, 56, 56, 256) | | 590080 |
| *block3\_conv3 (Conv2D)* | (None, 56, 56, 256) | | 590080 |
| *block3\_pool (MaxPooling2D)* | (None, 28, 28, 256) | | 0 |
| *block4\_conv1 (Conv2D)* | (None, 28, 28, 512) | | 1180160 |
| *block4\_conv2 (Conv2D)* | (None, 28, 28, 512) | | 2359808 |
| *block4\_conv3 (Conv2D)* | (None, 28, 28, 512) | | 2359808 |
| *block4\_pool (MaxPooling2D)* | (None, 14, 14, 512) | | 0 |
| *block5\_conv1 (Conv2D)* | (None, 14, 14, 512) | | 2359808 |
| *block5\_conv2 (Conv2D)* | (None, 14, 14, 512) | | 2359808 |
| *block5\_conv3 (Conv2D)* | (None, 14, 14, 512) | | 2359808 |
| *block5\_pool (MaxPooling2D)* | (None, 7, 7, 512) | | 0 |
| *average\_pooling2d (AveragePooling2D)* | (None, 1, 1, 512) | | 0 |
| *flatten (Flatten)* | (None, 512) | | 0 |
| *dense (Dense)* | (None, 128) | | 65664 |
| *dropout (Dropout)* | (None, 128) | | 0 |
| *dense\_1 (Dense)* | (None, 2) | | 258 |
| *Total params:* |  | | 14,780,610 |
| *Trainable params* |  | | 65,922 |
| *Non-trainable params:* |  | | 14,714,688 |

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 None x 224 x 224 x 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 skenariodengan 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.

### Hasil *Adaptive Moment Estimation*

#### Hasil Skenario 30 *Epoch*s

**Tabel 8.** Hasil*Adam* 3 *fold* 30 *epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 |
| *Avg* | 591.47 | 98.95 | 98.95 | 98.93 | 98.93 |

Berdasarkan Tabel 7, model yang dijalankan dengan metode Adaptive Moment Estimation (Adam) melalui 3 *fold* dan 30 *epoch*s menunjukkan performa yang sangat baik. Waktu pelatihan rata-rata mencapai 591.47 detik per *fold*. Akurasi, F1 *Score*, Presisi, dan *Recall* rata-rata mencapai sekitar 98.95%, 98.95%, 98.93%, dan 98.93% masing-masing, menunjukkan bahwa model ini memiliki kinerja yang sangat baik dalam mengklasifikasikan data dengan efisiensi dan konsistensi yang tinggi.

A blue squares with white text

Description automatically generated

**Gambar 24.** Hasil *Confusion Matrix* *Adam* 3 *Fold* 30 *epoch*s

Berdasarkan Gambar 22, menunjukkan Confusion Matrix untuk Adam 3 *fold* 30 *epoch*s menggambarkan kinerja model klasifikasi pada *Fold* ke-3. Nilai-nilai dalam matriks adalah:

*True Positives* (TP) = 261: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas positif dengan benar.

*False Negatives* (FN) = 6: Ini adalah jumlah kasus di mana model salah dalam memprediksi kelas negatif ketika sebenarnya adalah kelas positif.

*True Negatives* (TN) = 265: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas negatif dengan benar.

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

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kemampuan yang sangat baik dalam mengklasifikasikan data antara individu yang menggunakan masker dan tidak menggunakan masker.

**Tabel 9.** Hasil *Adam* 5 *fold* 30 *epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 |
| *Avg* | 578.45 | 99.74 | 99.74 | 99.74 | 99.74 |

Berdasarkan Tabel 8, Hasil skenario 30 *Epoch*s 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.

A blue squares with white text

Description automatically generated

**Gambar 25.** Hasil Confusion Matrix *Adam* 5 *Fold* 30 *Epoch*s

Berdasarkan Gambar 23, Confusion Matrix untuk Adam 5 fold 30 epochs menggambarkan kinerja model klasifikasi pada Fold ke-5. Nilai-nilai dalam matriks adalah:

True Positives (TP) = 145: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas positif dengan benar.

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

True Negatives (TN) = 174: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas negatif dengan benar.

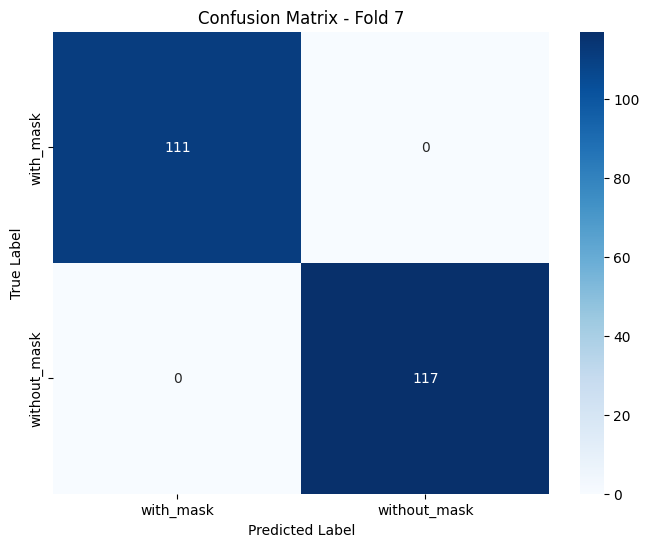
False Positives (FP) = 0: Ini adalah jumlah kasus di mana model salah dalam memprediksi kelas positif ketika sebenarnya adalah kelas negatif.

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kemampuan yang sangat baik dalam mengklasifikasikan data antara individu yang menggunakan masker dan tidak menggunakan masker.

**Tabel 10.** Hasil *Adam* 7 *Fold* 30 *Epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 |

Berdasarkan Tabel 9, Skenario 30 *Epoch*s menggunakan metode Adam 7 *fold* menghasilkan performa yang baik. 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 26.** Hasil Confusion Matrix *Adam* 7 *Fold* 30 *Epoch*s

Berdasarkan Gambar 24, Confusion Matrix untuk Adam 7 fold 30 epochs menggambarkan kinerja model klasifikasi pada Fold ke-7. Nilai-nilai dalam matriks adalah:

True Positives (TP) = 111: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas positif dengan benar.

False Negatives (FN) = 0: Ini adalah jumlah kasus di mana model salah dalam memprediksi kelas negatif ketika sebenarnya adalah kelas positif.

True Negatives (TN) = 117: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas negatif dengan benar.

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

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kemampuan yang sangat baik dalam mengklasifikasikan data antara individu yang menggunakan masker dan tidak menggunakan masker.

#### Hasil Skenario 50 *Epoch*s

**Tabel 11.** Hasil *Adam* 3 *Fold* 50 *Epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 |

Berdasarkan Tabel 10, Hasil skenario 50 *Epoch*s 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.

A blue squares with white text

Description automatically generated

**Gambar 27.** *Confusion Matrix* *Adam* 3 *fold* 50 *Epoch*s

Berdasarkan Gambar 25, Confusion Matrix untuk Adam 3 fold 50 epochs menggambarkan kinerja model klasifikasi pada Fold ke-3. Nilai-nilai dalam matriks adalah:

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

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

True Negatives (TN) = 275: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas negatif dengan benar.

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

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kemampuan yang sangat baik dalam mengklasifikasikan data antara individu yang menggunakan masker dan tidak menggunakan masker.

**Tabel 12.** Hasil *Adam* 5 *fold* 50 *epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 |
| *Avg* | 991.91 | 99.34 | 99.34 | 99.34 | 99.34 |

Berdasarkan Tabel 11, Hasil skenario 50 *Epoch*s 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.

A blue squares with white text

Description automatically generated

**Gambar 28.** *Confusion Matrix* *Adam* 5 *Fold* 50 *epoch*s

Berdasarkan Gambar 26, Confusion Matrix untuk Adam 5 fold 50 epochs menggambarkan kinerja model klasifikasi pada Fold ke-5. Nilai-nilai dalam matriks adalah:

True Positives (TP) = 149: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas positif dengan benar.

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

True Negatives (TN) = 169: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas negatif dengan benar.

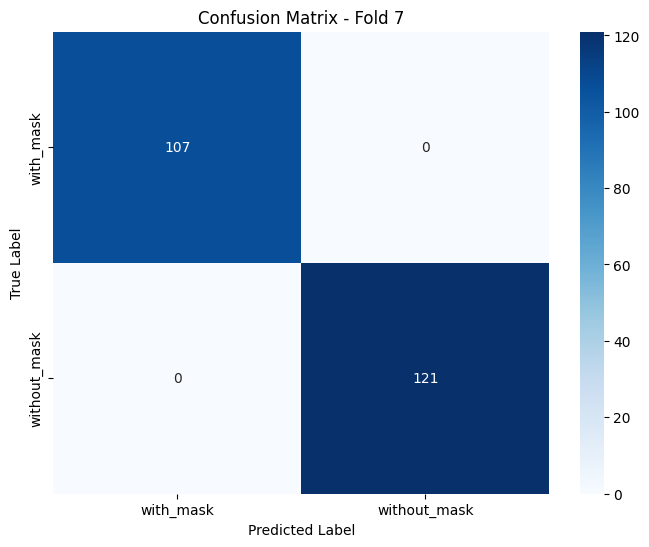
False Positives (FP) = 0: Ini adalah jumlah kasus di mana model salah dalam memprediksi kelas positif ketika sebenarnya adalah kelas negatif.

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kemampuan yang sangat baik dalam mengklasifikasikan data antara individu yang menggunakan masker dan tidak menggunakan masker.

**Tabel 13.** Hasil *Adam* 7 *fold* 50 *epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 |
| *Avg* | 996.71 | 99.24 | 99.24 | 99.24 | 99.24 |

Hasil skenario 50 *Epoch*s 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 29.** *Confusion Matrix Adam* 7 *fold* 50 *epoch*s

Berdasarkan Gambar 27, Confusion Matrix untuk Adam 7 fold 50 epochs menggambarkan kinerja model klasifikasi pada Fold ke-7. Nilai-nilai dalam matriks adalah:

True Positives (TP) = 107: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas positif dengan benar.

False Negatives (FN) = 0: Ini adalah jumlah kasus di mana model salah dalam memprediksi kelas negatif ketika sebenarnya adalah kelas positif.

True Negatives (TN) = 121: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas negatif dengan benar.

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

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kemampuan yang sangat baik dalam mengklasifikasikan data antara individu yang menggunakan masker dan tidak menggunakan masker.

#### Hasil Skenario 100 *Epoch*s

**Tabel 14.** Hasil *Adam* 3 *fold* 100 *epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 |
| *Avg* | 410.17 | 99.17 | 99.17 | 99.13 | 99.23 |

Hasil skenario 100 *Epoch*s 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.

A blue squares with white text

Description automatically generated

**Gambar 30**. *Confusion Matrix Adam* 3 *fold* 100 *epoch*s

Berdasarkan Gambar 28, Confusion Matrix untuk Adam 3 fold 100 epochs menggambarkan kinerja model klasifikasi pada Fold ke-3. Nilai-nilai dalam matriks adalah:

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

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

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

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

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kemampuan yang sangat baik dalam mengklasifikasikan data antara individu yang menggunakan masker dan tidak menggunakan masker.

**Tabel 15.** Hasil *Adam* 5 *fold* 100 *epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 |
| *Avg* | 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 *epoch*s 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%.

A blue squares with white text

Description automatically generated

**Gambar 31.** *Confusion Matrix Adam* 5 *fold* 100 *epoch*s

Berdasarkan Gambar 29, Confusion Matrix untuk Adam 5 fold 100 epochs menggambarkan kinerja model klasifikasi pada Fold ke-5. Nilai-nilai dalam matriks adalah:

True Positives (TP) = 143: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas positif dengan benar.

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

True Negatives (TN) = 176: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas negatif dengan benar.

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

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kemampuan yang sangat baik dalam mengklasifikasikan data antara individu yang menggunakan masker dan tidak menggunakan masker.

**Tabel 16.** Hasil *Adam* 7 *fold* 100 *epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 |
| *Avg* | 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 *epoch*s 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%

A blue squares with white text

Description automatically generated

**Gambar 32.** *Confusion Matrix Adam* 7 *fold* 100 *epoch*s

Berdasarkan Gambar 30, Confusion Matrix untuk Adam 7 fold 100 epochs menggambarkan kinerja model klasifikasi pada Fold ke-7. Nilai-nilai dalam matriks adalah:

True Positives (TP) = 121: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas positif dengan benar.

False Negatives (FN) = 0: Ini adalah jumlah kasus di mana model salah dalam memprediksi kelas negatif ketika sebenarnya adalah kelas positif.

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

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

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kemampuan yang sangat baik dalam mengklasifikasikan data antara individu yang menggunakan masker dan tidak menggunakan masker.

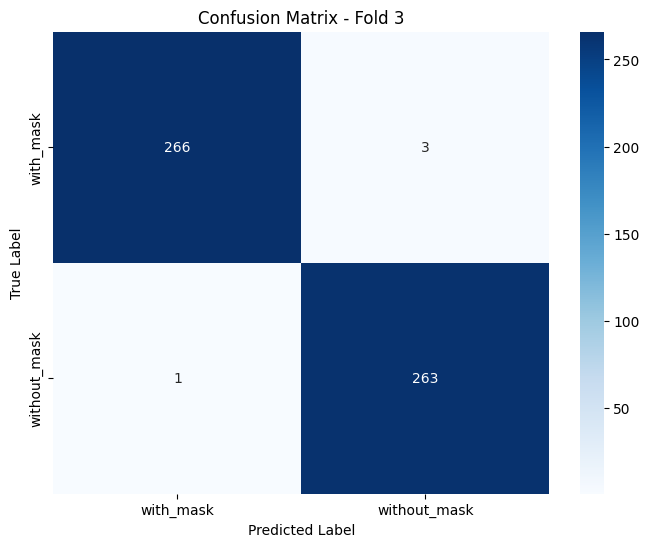
### Hasil *Root Mean Square Propagation*

#### Hasil Skenario 30 *Epoch*s

**Tabel 17.** Hasil *RMSprop* 3 *fold* 30 *epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 *epoch*s 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 33.** *Confusion Matrix RMSprop* 3 *fold* 30 *epoch*s

Berdasarkan Gambar 31, Confusion Matrix untuk RMSprop 3 fold 30 epochs menggambarkan kinerja model klasifikasi pada Fold ke-3. Nilai-nilai dalam matriks adalah:

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

False Negatives (FN) = 3: Ini adalah jumlah kasus di mana model salah dalam memprediksi kelas negatif ketika sebenarnya adalah kelas positif.

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

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

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kemampuan yang sangat baik dalam mengklasifikasikan data antara individu yang menggunakan masker dan tidak menggunakan masker.

**Tabel 18.** Hasil *RMSprop* 5 *fold* 30 *epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 |
| *Avg* | 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 *epoch*s 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 34.** *Confusion Matrix RMSprop* 5 *fold* 30 *Epoch*s

Berdasarkan Gambar 32, Confusion Matrix untuk RMSprop 5 fold 30 epochs menggambarkan kinerja model klasifikasi pada Fold ke-5. Nilai-nilai dalam matriks adalah:

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

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

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

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

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kemampuan yang sangat baik dalam mengklasifikasikan data antara individu yang menggunakan masker dan tidak menggunakan masker.

**Tabel 19.** Hasil *RMSprop* 7 *fold* 30 *epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 |
| *Avg* | 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 *epoch*s 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.

A blue squares with white text

Description automatically generated

**Gambar 35.** *Confusion Matrix RMSprop* 7 *fold* 30 *epoch*s

Berdasarkan Gambar 33, Confusion Matrix untuk RMSprop 7 fold 30 epochs menggambarkan kinerja model klasifikasi pada Fold ke-7. Nilai-nilai dalam matriks adalah:

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

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

True Negatives (TN) = 108: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas negatif dengan benar.

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

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kemampuan yang sangat baik dalam mengklasifikasikan data antara individu yang menggunakan masker dan tidak menggunakan masker.

#### Hasil Skenario 50 *Epoch*s

**Tabel 20.** Hasil *RMSprop* 3 *fold* 50 *epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 |
| *Avg* | 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 *epoch*s 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.

A blue squares with white text

Description automatically generated

**Gambar 36.** *Confusion Matrix RMSprop* 3 *fold* 50 *epoch*s

Berdasarkan Gambar 34, Confusion Matrix untuk RMSprop 3 fold 50 epochs menggambarkan kinerja model klasifikasi pada Fold ke-3. Nilai-nilai dalam matriks adalah:

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

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

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

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

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kemampuan yang sangat baik dalam mengklasifikasikan data antara individu yang menggunakan masker dan tidak menggunakan masker.

**Tabel 21**. Hasil *RMSprop* 5 *fold* 50 *epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 |
| *Avg* | 981.38 | 98.60 | 98.60 | 98.64 | 98.59 |

Berdasarkan Tabel 20, dapat dilihat bahwa model yang dilatih dengan metode *RMSprop* selama 5 *fold* dan 50 *epoch*s menunjukkan performa yang sangat baik. Waktu pelatihan rata-rata adalah 981.38 detik per *fold*. Akurasi, F1 *Score*, Presisi, dan *Recall* rata-rata mencapai lebih dari 98.60%, yang menunjukkan bahwa model memiliki kinerja yang konsisten dan efisien dalam mengklasifikasikan data dengan tepat.

A blue squares with white text

Description automatically generated

**Gambar 37.** *Confusion Matrix RMSprop* 5 *fold* 50 *epoch*s

Berdasarkan Gambar 35. Confusion Matrix untuk *Fold* 5 adalah tabel yang digunakan untuk menggambarkan kinerja model klasifikasi. Nilai-nilai dalam matriks adalah:

*True Positives* (TP) = 167: 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) = 150: 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.

**Tabel 22.** Hasil *RMSprop* 7 *fold* 50 *epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 |
| *Avg* | 991.01 | 98.90 | 98.89 | 98.88 | 98.90 |

Berdasarkan Tabel 21, terlihat bahwa model yang dijalankan dengan metode *RMSprop* melalui 7 *fold* dan 50 *epoch*s menunjukkan performa yang luar biasa. Waktu pelatihan rata-rata mencapai 991.01 detik per *fold*. Akurasi, F1 *Score*, Presisi, dan *Recall* rata-rata semua mencapai lebih dari 98.90%, menunjukkan bahwa model ini mampu mengklasifikasikan data dengan efisiensi dan konsistensi yang tinggi.

A blue squares with white text

Description automatically generated

**Gambar 38.** *Confusion Matrix RMSprop* 7 *fold* 50 *epoch*s

Gambar 36 menunjukkan Confusion Matrix untuk *Fold* 7 50 *epoch*s, yang digunakan untuk mengilustrasikan kinerja model klasifikasi. Nilai-nilai dalam matriks tersebut adalah:

*True Positives* (TP) = 111: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas positif dengan benar.

*False Negatives* (FN) = 0: Ini adalah jumlah kasus di mana model salah dalam memprediksi kelas negatif ketika sebenarnya adalah kelas positif.

*True Negatives* (TN) = 117: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas negatif dengan benar.

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

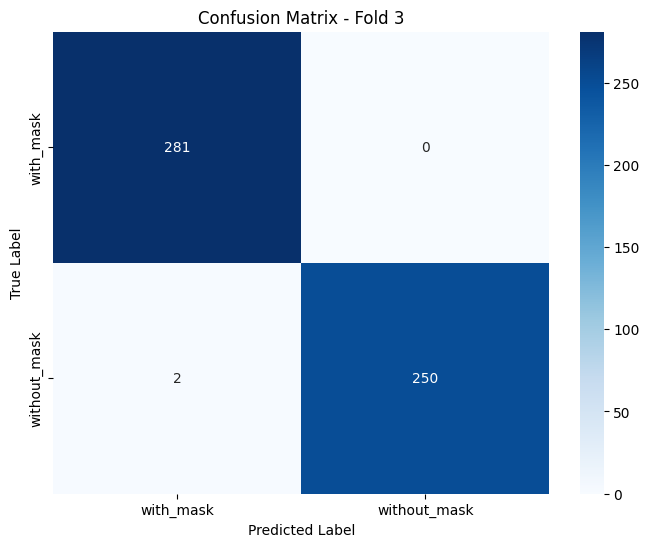
Dengan demikian, hasil ini menunjukkan bahwa model memiliki kinerja yang konsisten dan efisien dalam mengklasifikasikan data dengan tepat

#### Hasil Skenario 100 *Epoch*s

**Tabel 23.** Hasil *RMSprop* 3 *fold* 100 *epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 |
| *Avg* | 1846.94 | 99.18 | 99.17 | 99.18 | 99.17 |

Berdasarkan Tabel 22, terlihat bahwa model yang dijalankan dengan metode *RMSprop* melalui 3 *fold* dan 100 *epoch*s menunjukkan performa yang luar biasa. Waktu pelatihan rata-rata mencapai 1846.94 detik per *fold*. Akurasi, F1 *Score*, Presisi, dan *Recall* rata-rata semua mencapai lebih dari 99.17%, menunjukkan bahwa model ini mampu mengklasifikasikan data dengan efisiensi dan konsistensi yang tinggi.



**Gambar 39.** *Confusion Matrix RMSprop* 3 *fold* 100 *epoch*s

Gambar 37 menunjukkan Confusion Matrix untuk *RMSprop* 3 *fold* 100 *epoch*s menggambarkan kinerja model klasifikasi pada *Fold* ke-3. Nilai-nilai dalam matriks adalah:

*True Positives* (TP) = 281: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas positif dengan benar.

*False Negatives* (FN) = 0: Ini adalah jumlah kasus di mana model salah dalam memprediksi kelas negatif ketika sebenarnya adalah kelas positif.

*True Negatives* (TN) = 250: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas negatif dengan benar.

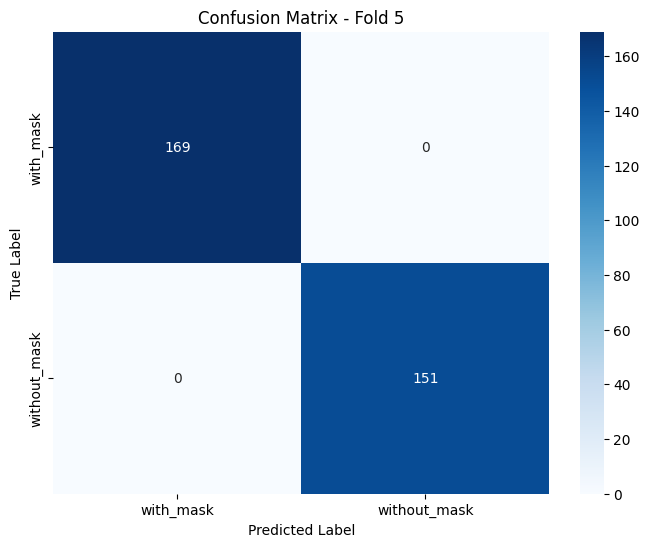
*False Positives* (FP) = 2: Ini adalah jumlah kasus di mana model salah dalam memprediksi kelas positif ketika sebenarnya adalah kelas negatif.

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kemampuan yang sangat baik dalam mengklasifikasikan data antara individu yang menggunakan masker dan tidak menggunakan masker.

**Tabel 24.** Hasil *RMSprop* 5 *fold* 100 *epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 |
| *Avg* | 1892.57 | 99.06 | 99.05 | 99.07 | 99.05 |

Berdasarkan Tabel 23, terlihat bahwa model yang dijalankan dengan metode *RMSprop* melalui 5 *fold* dan 100 *epoch*s menunjukkan performa yang luar biasa. Waktu pelatihan rata-rata mencapai 1892.57 detik per *fold*. Akurasi, F1 *Score*, Presisi, dan *Recall* rata-rata semua mencapai lebih dari 99.05%, menunjukkan bahwa model ini mampu mengklasifikasikan data dengan efisiensi dan konsistensi yang tinggi.



**Gambar 40.** *Confusion Matrix RMSprop* 5 *fold* 100 *epoch*s

Gambar 38 menunjukkan Confusion Matrix untuk *RMSprop* 5 *fold* 100 *epoch*s menggambarkan kinerja model klasifikasi pada *Fold* ke-5. Nilai-nilai dalam matriks adalah:

*True Positives* (TP) = 169: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas positif dengan benar.

*False Negatives* (FN) = 0: Ini adalah jumlah kasus di mana model salah dalam memprediksi kelas negatif ketika sebenarnya adalah kelas positif.

*True Negatives* (TN) = 151: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas negatif dengan benar.

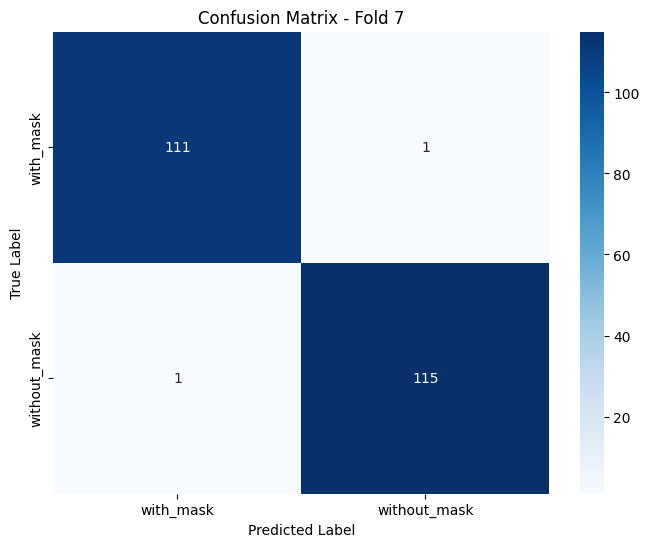
*False Positives* (FP) = 0: Ini adalah jumlah kasus di mana model salah dalam memprediksi kelas positif ketika sebenarnya adalah kelas negatif.

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kemampuan yang sangat baik dalam mengklasifikasikan data antara individu yang menggunakan masker dan tidak menggunakan masker.

**Tabel 25.** Hasil *RMSprop* 7 *fold* 100 *epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 |
| *Avg* | 1917.11 | 99.18 | 99.17 | 99.17 | 99.17 |

Berdasarkan Tabel 24, terlihat bahwa model yang dijalankan dengan metode *RMSprop* melalui 7 *fold* dan 100 *epoch*s menunjukkan performa yang luar biasa. Waktu pelatihan rata-rata mencapai 1917.11 detik per *fold*. Akurasi, F1 *Score*, Presisi, dan *Recall* rata-rata semua mencapai lebih dari 99.17%, menunjukkan bahwa model ini mampu mengklasifikasikan data dengan efisiensi dan konsistensi yang tinggi.



**Gambar 41.** *Confusion Matrix RMSprop* 7 *fold* 100 *epoch*s

Gambar 39 menunjukkan Confusion Matrix untuk *RMSprop* 7 *fold* 100 *epoch*s menggambarkan kinerja model klasifikasi pada *Fold* ke-7. Nilai-nilai dalam matriks adalah:

*True Positives* (TP) = 111: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas positif dengan benar.

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

*True Negatives* (TN) = 115: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas negatif dengan benar.

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

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kemampuan yang sangat baik dalam mengklasifikasikan data antara individu yang menggunakan masker dan tidak menggunakan masker.

### Hasil *Stochastic Gradient Descent*

#### Hasil Skenario *SGD* 30 *Epoch*s

**Tabel 26.** Hasil *SGD* 3 *fold* 30 *epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 |

Berdasarkan Tabel 25, terlihat bahwa model yang dijalankan dengan metode *Stochastic Gradient Descent* (SGD) melalui 3 *fold* dan 30 *epochs* menunjukkan performa yang bervariasi. Waktu pelatihan rata-rata mencapai 593.97 detik per *fold*. Akurasi, F1 *Score*, *Presisi*, dan *Recall* rata-rata mencapai sekitar 70%, menunjukkan bahwa model ini memiliki kinerja yang cukup baik dalam mengklasifikasikan data.

A blue squares with white text

Description automatically generated

**Gambar 42.** *Confusion Matrix SGD* 3 *fold* 30 *epoch*s

Berdasarkan Gambar 40, Confusion Matrix untuk SGD 3 *fold* 30 *epoch*s adalah tabel yang digunakan untuk menggambarkan kinerja model klasifikasi. Nilai-nilai dalam matriks adalah:

*True Positives* (TP) = 225: Ini adalah kasus di mana model dengan benar memprediksi kelas positif.

*False Negatives* (FN) = 47: Ini adalah kasus di mana model salah memprediksi kelas negatif ketika sebenarnya positif.

*True Negatives* (TN) = 246: Ini adalah kasus di mana model dengan benar memprediksi kelas negatif.

*False Positives* (FP) = 15: Ini adalah kasus di mana model salah memprediksi kelas positif ketika sebenarnya negatif.

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kinerja yang cukup baik dalam mengklasifikasikan data dengan tepat.

**Tabel 27.** Hasil *SGD* 5 *fold* 30 *epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 |
| *Avg* | 583.70 | 61.63 | 60.85 | 68.64 | 65.63 |

Berdasarkan Tabel 26, terlihat bahwa model yang dijalankan dengan metode *Stochastic Gradient Descent* (SGD) melalui 5 *fold* dan 30 *epoch*s menunjukkan performa yang bervariasi. Waktu pelatihan rata-rata mencapai 583.70 detik per *fold*. Akurasi, F1 *Score*, Presisi, dan *Recall* rata-rata mencapai sekitar 61.63%, 60.85%, 68.64%, dan 65.63% masing-masing, menunjukkan bahwa model ini memiliki kinerja yang cukup baik dalam mengklasifikasikan data.

A blue squares with white text

Description automatically generated

**Gambar 43.** *Confusion Matrix SGD* 5 *fold* 30 *epoch*s

Berdasarkan Gambar 41, Confusion Matrix untuk SGD 5 *fold* 30 *epoch*s adalah tabel yang digunakan untuk menggambarkan kinerja model klasifikasi. Nilai-nilai dalam matriks adalah:

*True Positives* (TP) = 128: Ini adalah kasus di mana model dengan benar memprediksi kelas positif.

*False Negatives* (FN) = 27: Ini adalah kasus di mana model salah memprediksi kelas negatif ketika sebenarnya positif.

*True Negatives* (TN) = 142: Ini adalah kasus di mana model dengan benar memprediksi kelas negatif.

*False Positives* (FP) = 23: Ini adalah kasus di mana model salah memprediksi kelas positif ketika sebenarnya negatif.

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kinerja yang cukup baik dalam mengklasifikasikan data dengan tepat.

**Tabel 28.** Hasil *SGD* 7 *fold* 30 *epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 |
| *Avg* | 627.56 | 91.68 | 91.37 | 91.84 | 91.25 |

Berdasarkan Tabel 27, terlihat bahwa model yang dijalankan dengan metode Stochastic Gradient Descent (SGD) melalui 7 *fold* dan 30 *epoch*s menunjukkan performa yang bervariasi. Waktu pelatihan rata-rata mencapai 627.56 detik per *fold*. Akurasi, F1 *Score*, Presisi, dan *Recall* rata-rata mencapai sekitar 91.68%, 91.37%, 91.84%, dan 91.25% masing-masing, menunjukkan bahwa model ini memiliki kinerja yang cukup baik dalam mengklasifikasikan data.

A blue squares with white text

Description automatically generated

**Gambar 44.** *Confusion Matrix SGD* 7 *fold* 30 *epoch*s

Berdasarkan Gambar 42, Confusion Matrix untuk SGD 7 *fold* 30 *epoch*s adalah tabel yang digunakan untuk menggambarkan kinerja model klasifikasi. Nilai-nilai dalam matriks adalah:

*True Positives* (TP) = 96: Ini adalah kasus di mana model dengan benar memprediksi kelas positif.

*False Negatives* (FN) = 8: Ini adalah kasus di mana model salah memprediksi kelas negatif ketika sebenarnya positif.

*True Negatives* (TN) = 118: Ini adalah kasus di mana model dengan benar memprediksi kelas negatif.

*False Positives* (FP) = 6: Ini adalah kasus di mana model salah memprediksi kelas positif ketika sebenarnya negatif.

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kinerja yang cukup baik dalam mengklasifikasikan data dengan tepat.

#### Hasil Skenario *SGD* 50 *Epoch*s

**Tabel 29.** Hasil *SGD* 3 *fold* 50 *epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 |
| *Avg* | 892.41 | 65.27 | 64.80 | 70.77 | 67.94 |

Berdasarkan Tabel 28, terlihat bahwa model yang dijalankan dengan metode Stochastic Gradient Descent (SGD) melalui 3 *fold* dan 50 *epoch*s menunjukkan performa yang bervariasi. Waktu pelatihan rata-rata mencapai 892.41 detik per *fold*. Akurasi, F1 *Score*, Presisi, dan *Recall* rata-rata mencapai sekitar 65.27%, 64.80%, 70.77%, dan 67.94% masing-masing, menunjukkan bahwa model ini memiliki kinerja yang cukup baik dalam mengklasifikasikan data

A blue squares with white text

Description automatically generated

**Gambar 45.** *Confusion Matrix SGD* 3 *fold* 50 *epoch*s

Berdasarkan Gambar 43, Confusion Matrix untuk SGD 3 *fold* 50 *epoch*s adalah tabel yang digunakan untuk menggambarkan kinerja model klasifikasi. Nilai-nilai dalam matriks adalah:

*True Positives* (TP) = 131: Ini adalah kasus di mana model dengan benar memprediksi kelas positif.

*False Negatives* (FN) = 154: Ini adalah kasus di mana model salah memprediksi kelas negatif ketika sebenarnya positif.

*True Negatives* (TN) = 245: Ini adalah kasus di mana model dengan benar memprediksi kelas negatif.

*False Positives* (FP) = 6: Ini adalah kasus di mana model salah memprediksi kelas positif ketika sebenarnya negatif.

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kinerja yang cukup baik dalam mengklasifikasikan data dengan tepat.

**Tabel 30.** Hasil *SGD* 5 *fold* 50 *epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 |
| *Avg* | 976.98 | 87.43 | 87.20 | 89.03 | 87.76 |

Berdasarkan Tabel 29, model yang menggunakan metode Stochastic Gradient Descent (SGD) dengan 5 *fold* dan 50 *epoch*s menunjukkan variasi dalam performa. Waktu pelatihan rata-rata adalah 976.98 detik per *fold*. Rata-rata Akurasi, F1 *Score*, Presisi, dan *Recall* adalah sekitar 87.43%, 87.20%, 89.03%, dan 87.76% masing-masing, yang menunjukkan bahwa model ini cukup baik dalam mengklasifikasi data.

A blue squares with white text

Description automatically generated

**Gambar 46.** *Confusion Matrix SGD* 5 *fold* 50 *epoch*s

Berdasarkan Gambar 44, Confusion Matrix untuk SGD 5 *fold* 50 *epoch*s menggambarkan kinerja model klasifikasi pada *Fold* ke-5. Nilai-nilai dalam matriks adalah:

*True Positives* (TP) = 143: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas positif dengan benar.

*False Negatives* (FN) = 11: Ini adalah jumlah kasus di mana model salah dalam memprediksi kelas negatif ketika sebenarnya adalah kelas positif.

*True Negatives* (TN) = 166: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas negatif dengan benar.

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

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kemampuan yang baik dalam mengklasifikasikan data antara individu yang menggunakan masker dan tidak menggunakan masker.

**Tabel 31.** Hasil *SGD* 7 *fold* 50 *epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 |
| *Avg* | 992.38 | 94.57 | 94.38 | 94.45 | 94.61 |

Berdasarkan Tabel 30, model yang dijalankan dengan metode Stochastic Gradient Descent (SGD) melalui 7 *fold* dan 50 *epoch*s menunjukkan performa yang bervariasi. Waktu pelatihan rata-rata mencapai 992.38 detik per *fold*. Akurasi, F1 *Score*, Presisi, dan *Recall* rata-rata mencapai sekitar 94.57%, 94.38%, 94.45%, dan 94.61% masing-masing, menunjukkan bahwa model ini memiliki kinerja yang cukup baik dalam mengklasifikasikan data.

A blue squares with white text

Description automatically generated

**Gambar 47.** *Confusion Matrix SGD* 7 *fold* 50 *epoch*s

Berdasarkan Gambar 45 menunjukkan Confusion Matrix untuk SGD 7 *fold* 50 *epoch*s menggambarkan kinerja model klasifikasi pada *Fold* ke-7. Nilai-nilai dalam matriks adalah:

*True Positives* (TP) = 103: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas positif dengan benar.

*False Negatives* (FN) = 10: Ini adalah jumlah kasus di mana model salah dalam memprediksi kelas negatif ketika sebenarnya adalah kelas positif.

*True Negatives* (TN) = 112: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas negatif dengan benar.

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

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kemampuan yang cukup baik dalam mengklasifikasikan data antara individu yang menggunakan masker dan tidak menggunakan masker.

#### Hasil Skenario *SGD* 100 *epoch*s

**Tabel 32.** Hasil *SGD* 3 *fold* 100 *epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 |
| *Avg* | 1918.13 | 83.88 | 82.49 | 85.57 | 83.57 |

Berdasarkan Tabel 31, model yang dijalankan dengan metode Stochastic Gradient Descent (SGD) melalui 3 *fold* dan 100 *epoch*s menunjukkan performa yang bervariasi. Waktu pelatihan rata-rata mencapai 1918.13 detik per *fold*. Akurasi, F1 *Score*, Presisi, dan *Recall* rata-rata mencapai sekitar 83.88%, 82.49%, 85.57%, dan 83.57% masing-masing, menunjukkan bahwa model ini memiliki kinerja yang cukup baik dalam mengklasifikasikan data.

A blue squares with white text

Description automatically generated

**Gambar 48.** *Confusion Matrix SGD* 3 *fold* 100 *epoch*s

Berdasarkan Gambar 46 menunjukkan Confusion Matrix untuk SGD 3 *fold* 100 *epoch*s menggambarkan kinerja model klasifikasi pada *Fold* ke-3. Nilai-nilai dalam matriks adalah:

*True Positives* (TP) = 216: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas positif dengan benar.

*False Negatives* (FN) = 40: Ini adalah jumlah kasus di mana model salah dalam memprediksi kelas negatif ketika sebenarnya adalah kelas positif.

*True Negatives* (TN) = 275: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas negatif dengan benar.

*False Positives* (FP) = 2: Ini adalah jumlah kasus di mana model salah dalam memprediksi kelas positif ketika sebenarnya adalah kelas negatif.

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kemampuan yang cukup baik dalam mengklasifikasikan data antara individu yang menggunakan masker dan tidak menggunakan masker.

**Tabel 33.** Hasil *SGD* 5 *fold* 100 *epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 |
| *Avg* | 1919.57 | 92.08 | 91.63 | 92.18 | 91.42 |

Berdasarkan Tabel 32, model yang dijalankan dengan metode Stochastic Gradient Descent (SGD) melalui 5 *fold* dan 100 *epoch*s menunjukkan performa yang bervariasi. Waktu pelatihan rata-rata mencapai 1919.57 detik per *fold*. Akurasi, F1 *Score*, Presisi, dan *Recall* rata-rata mencapai sekitar 92.08%, 91.63%, 92.18%, dan 91.42% masing-masing, menunjukkan bahwa model ini memiliki kinerja yang cukup baik dalam mengklasifikasikan data .

A blue squares with white text

Description automatically generated

**Gambar 49.** *Confusion Matrix SGD* 5 *Fold* 100 *epoch*s

Berdasarkan Gambar 47 menunjukkan Confusion Matrix untuk SGD 5 *fold* 100 *epoch*s menggambarkan kinerja model klasifikasi pada *Fold* ke-5. Nilai-nilai dalam matriks adalah:

*True Positives* (TP) = 144: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas positif dengan benar.

*False Negatives* (FN) = 16: Ini adalah jumlah kasus di mana model salah dalam memprediksi kelas negatif ketika sebenarnya adalah kelas positif.

*True Negatives* (TN) = 159: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas negatif dengan benar.

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

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kemampuan yang cukup baik dalam mengklasifikasikan data antara individu yang menggunakan masker dan tidak menggunakan masker.

**Tabel 34.** Hasil *SGD* 7 *fold* 100 *epoch*s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Fold* | *Training Time (seconds)* | *Accuracy* | *F1 Score* | *Precision* | *Recall* |
| 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 |
| *Avg* | 1936.38 | 95.68 | 95.51 | 95.66 | 95.72 |

Berdasarkan Tabel 33, model yang dijalankan dengan metode Stochastic Gradient Descent (SGD) melalui 7 *fold* dan 100 *epoch*s menunjukkan performa yang bervariasi. Waktu pelatihan rata-rata mencapai 1936.38 detik per *fold*. Akurasi, F1 *Score*, Presisi, dan *Recall* rata-rata mencapai sekitar 95.68%, 95.51%, 95.66%, dan 95.72% masing-masing, menunjukkan bahwa model ini memiliki kinerja yang cukup baik dalam mengklasifikasikan data dengan efisiensi dan konsistensi yang tinggi.

A blue squares with white text

Description automatically generated

**Gambar 50.** *Confusion Matrix SGD* 7 *fold* 100 *epoch*s

Berdasarkan Gambar 48 menunjukkan Confusion Matrix untuk SGD 7 *fold* 100 *epoch*s menggambarkan kinerja model klasifikasi pada *Fold* ke-7. Nilai-nilai dalam matriks adalah:

*True Positives* (TP) = 114: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas positif dengan benar.

*False Negatives* (FN) = 8: Ini adalah jumlah kasus di mana model salah dalam memprediksi kelas negatif ketika sebenarnya adalah kelas positif.

*True Negatives* (TN) = 105: Ini adalah jumlah kasus di mana model berhasil memprediksi kelas negatif dengan benar.

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

Dengan demikian, hasil ini menunjukkan bahwa model memiliki kemampuan yang cukup baik dalam mengklasifikasikan data antara individu yang menggunakan masker dan tidak menggunakan masker.

## Hasil Rata-Rata Training

**Tabel 35.** Tabel Hasil Eksperimen

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No. | *Optimizer* | *Epochs* | | | *Fold* | *Avg Training Time (s)* | *Avg Accuracy (%)* | *Avg F1 Score* | *Avg Precision* | *Avg Recall* |
| 1 | *Adam* | | 30 | 3 | | 591.47 | 98.95 | 98.95 | 98.93 | 98.93 |
| 2 | *Adam* | | 30 | 5 | | 578.45 | 99.74 | 99.74 | 99.74 | 99.74 |
| 3 | *Adam* | | 30 | 7 | | 644.26 | 99.44 | 99.43 | 99.44 | 99.43 |
| 4 | *Adam* | | 50 | 3 | | 963.15 | 99.04 | 99.04 | 99.04 | 99.04 |
| 5 | *Adam* | | 50 | 5 | | 991.91 | 99.34 | 99.34 | 99.34 | 99.34 |
| 6 | *Adam* | | 50 | 7 | | 996.71 | 99.24 | 99.24 | 99.24 | 99.24 |
| 7 | *Adam* | | 100 | 3 | | 410.17 | 99.17 | 99.17 | 99.13 | 99.23 |
| 8 | *Adam* | | 100 | 5 | | 1962.92 | 99.17 | 99.16 | 99.17 | 99.16 |
| 9 | *Adam* | | 100 | 7 | | 1932.10 | 99.19 | 99.19 | 99.18 | 99.19 |
| 10 | *RMSprop* | | 30 | 3 | | 554.47 | 98.97 | 98.97 | 98.98 | 98.97 |
| 11 | *RMSprop* | | 30 | 5 | | 582.27 | 98.02 | 98.02 | 98.03 | 98.02 |
| 12 | *RMSprop* | | 30 | 7 | | 634.94 | 99.00 | 98.99 | 98.99 | 98.99 |
| 13 | *RMSprop* | | 50 | 3 | | 958.83 | 98.88 | 98.87 | 98.88 | 98.88 |
| 14 | *RMSprop* | | 50 | 5 | | 981.38 | 98.60 | 98.60 | 98.64 | 98.59 |
| 15 | *RMSprop* | | 50 | 7 | | 991.01 | 98.90 | 98.89 | 98.88 | 98.90 |
| 16 | *RMSprop* | | 100 | 3 | | 1846.94 | 99.18 | 99.17 | 99.18 | 99.17 |
| 17 | *RMSprop* | | 100 | 5 | | 1892.57 | 99.06 | 99.05 | 99.07 | 99.05 |
| 18 | *RMSprop* | | 100 | 7 | | 1917.11 | 99.18 | 99.17 | 99.17 | 99.17 |
| 19 | *SGD* | | 30 | 3 | | 593.97 | 69.83 | 69.62 | 70.60 | 69.57 |
| 20 | *SGD* | | 30 | 5 | | 583.70 | 61.63 | 60.85 | 68.64 | 65.63 |
| 21 | *SGD* | | 30 | 7 | | 627.56 | 91.68 | 91.37 | 91.84 | 91.25 |
| 22 | *SGD* | | 50 | 3 | | 892.41 | 65.27 | 64.80 | 70.77 | 67.94 |
| 23 | *SGD* | | 50 | 5 | | 976.98 | 87.43 | 87.20 | 89.03 | 87.76 |
| 24 | *SGD* | | 50 | 7 | | 992.38 | 94.57 | 94.38 | 94.45 | 94.61 |
| 25 | *SGD* | | 100 | 3 | | 1918.13 | 83.88 | 82.49 | 85.57 | 83.57 |
| 26 | *SGD* | | 100 | 5 | | 1919.57 | 92.08 | 91.63 | 92.18 | 91.42 |
| 27 | *SGD* | | 100 | 7 | | 1936.38 | 95.68 | 95.51 | 95.66 | 95.72 |

Berdasarkan hasil dari Tabel 35, Data ini menunjukkan hasil dari tiga jenis *optimizer* (*Adam*, *RMSprop*, dan *SGD*) dengan variasi jumlah *epoch*s (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.

## 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 36.** Sampel Pengujian Data Tunggal

| No | Kelas | Citra | Output | Akurasi | Status Identifikasi |
| --- | --- | --- | --- | --- | --- |
| 1 | *Mask* |  | Mask | 100% | Benar |
| 2 | *Mask* |  | Mask | 100% | Benar |
| 3 | *Mask* |  | Mask | 100% | Benar |
| 4 | *Mask* |  | Mask | 100% | Benar |
| 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 pada bagian citra wanita yang menggunakan hijab, hal tersebut karena arsitektur deteksi wajah MTCNN tidak mendeteksi sehingga tidak bisa di prediksi dengan baik. 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, hal itu juga terjadi karena arsitektur MTCNN tidak bisa mendeteksi wajah saat adaa obyek lain selain masker yang menutup wajah orang tersebut.

# PENUTUP

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

## KESIMPULAN

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

Dalam penelitian ini, peneliti berhasil mengimplementasikan algoritma *Convolutional Neural Network* (CNN) dengan menggunakan arsitektur *VGG16Net* untuk mendeteksi penggunaan masker pada citra wajah. Proses deteksi tersebut melibatkan beberapa tahapan, seperti memilih arsitektur model CNN dengan VGG16Net sebagai *base model* untuk ekstraksi fitur wajah, selanjutnya *preprocessing* dataset seperti normalisasi, *resize*, *one-hot encoding*, dan *data augmentation*, pembentukan model CNN, dan pelatihan model menggunakan teknik *k-Fold Cross Validation* dan deteksi wajah menggunakan MTCNN*.*

Pemilihan metode *optimizer* sangatlah berpengaruh pada tingkat akurasian juga. Dalam penelitian ini metode *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*.

## SARAN

Untuk penelitian selanjutnya, disarankan menggunakan *optimizer* *Adam* atau *RMSprop* karena menunjukkan performa yang lebih baik dibandingkan dengan *SGD*

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.

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

Çakir, M., & Çınarer, G. (n.d.). *MECHANICAL OBJECT PARTS DETECTION USING DEEP LEARNING BASED YOLO MODELS*. https://doi.org/10.5281/zenodo.7496767

Canedo, D., & Neves, A. J. R. (2019). Facial expression recognition using computer vision: A systematic review. In *Applied Sciences (Switzerland)* (Vol. 9, Issue 21). MDPI AG. https://doi.org/10.3390/app9214678

Dhanny, S., Andikha, D. P., Kezia, S., & Jenisa, F. (n.d.). *Implementasi Convolutional Neural Network untuk Facial Recognition IMPLEMENTASI COVOLUTIONAL NEURAL NETWORK UNTUK FACIAL RECOGNITION*.

El-Moneim, S. A., Sedik, A., Nassar, M. A., El-Fishawy, A. S., Sharshar, A. M., Hassan, S. E. A., Mahmoud, A. Z., Dessouky, M. I., El-Banby, G. M., El-Samie, F. E. A., El-Rabaie, E. S. M., Neyazi, B., Seddeq, H. S., Ismail, N. A., Khalaf, A. A. M., & Elabyad, G. S. M. (2021). Text-dependent and text-independent speaker recognition of reverberant speech based on CNN. *International Journal of Speech Technology*, *24*(4), 993–1006. https://doi.org/10.1007/s10772-021-09805-3

FUADAH, Y. N., UBAIDULLAH, I. D., IBRAHIM, N., TALININGSING, F. F., SY, N. K., & PRAMUDITHO, M. A. (2022). Optimasi Convolutional Neural Network dan K-Fold Cross Validation pada Sistem Klasifikasi Glaukoma. *ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika*, *10*(3), 728. https://doi.org/10.26760/elkomika.v10i3.728

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

Jurjawi, I. (2020). *Implementasi Pengenalan Wajah Secara Real Time untuk Sistem Absensi Menggunakan Metode Pembelajaran Deep Learning dengan Pustaka Open CV (Computer Vision)*. https://repositori.usu.ac.id/handle/123456789/27828

Khattar, A., & Quadri, S. M. K. (2023). Multi-source domain adaptation of social media data for disaster management. *Multimedia Tools and Applications*, *82*(6), 9083–9111. https://doi.org/10.1007/s11042-022-13456-0

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

Kumar, A., Sarkar, S., & Pradhan, C. (2020). Malaria Disease Detection Using CNN Technique with SGD, RMSprop and ADAM Optimizers. In *Studies in Big Data* (Vol. 68, pp. 211–230). Springer Science and Business Media Deutschland GmbH. https://doi.org/10.1007/978-3-030-33966-1\_11

Meruva, S. K. R., Tulasi, V. G. S., Vinnakota, N., & Bhavana, V. (2022). Risk Level Prediction of Diabetic Retinopathy based on Retinal Images using Deep Learning Algorithm. *Procedia Computer Science*, *215*, 722–730. https://doi.org/10.1016/j.procs.2022.12.074

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*.

*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

Perez, L., & Wang, J. (2017). *The Effectiveness of Data Augmentation in Image Classification using Deep Learning*. http://arxiv.org/abs/1712.04621

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

Santoso, P., Abijono, H., & Anggreini, N. L. (2021). ALGORITMA SUPERVISED LEARNING DAN UNSUPERVISED LEARNING DALAM PENGOLAHAN DATA. *Unira Malang |*, *4*(2).

Shen, K., Guo, J., Tan, X., Tang, S., Wang, R., & Bian, J. (2023). *A Study on ReLU and Softmax in Transformer*. http://arxiv.org/abs/2302.06461

Sheta, A., & Byers, B. (n.d.). *Design of Convolutional Neural Networks for Fish Recognition and Tracking*. http://www.southernct.edu

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

Teoh, K., Ismail, R., Naziri, S., Hussin, R., Isa, M., & Basir, M. (2021). Face Recognition and Identification using Deep Learning Approach. *Journal of Physics: Conference Series*, *1755*(1), 012006. https://doi.org/10.1088/1742-6596/1755/1/012006

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

Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters*, *23*(10), 1499–1503. https://doi.org/10.1109/LSP.2016.2603342

Zollanvari, A. (2023). Deep Learning with Keras-TensorFlow. *Machine Learning with Python*, 351–391. https://doi.org/10.1007/978-3-031-33342-2\_13

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/