### **BAB I** 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)



Gambar 1.1 Berita Covid-19 muncul pertama kali di Wuhan.

Pada Gambar 1.1 dalam berita tersebut Virus Corona atau COVID-19 pertama kali ditemukan di Wuhan, China pada akhir 2019. Kasus ini diduga berkaitan dengan pasar hewan Huanan di Wuhan yang menjual berbagai jenis daging binatang, termasuk yang tidak biasa dikonsumsi, misal ular, kelelawar, dan berbagai jenis tikus 12. Virus Corona atau COVID-19 diduga dibawa kelelawar dan hewan lain yang dimakan manusia hingga terjadi penularan. Coronavirus sebetulnya tidak asing dalam dunia kesehatan hewan, tapi hanya beberapa jenis yang mampu menginfeksi manusia hingga menjadi penyakit radang paru 1. Sebelum COVID-19 mewabah, dunia sempat heboh dengan SARS dan MERS, yang juga berkaitan dengan virus Corona.

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

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

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

*Machine Learning* adalah cabang ilmu kecerdasan buatan yang berkembang pesat dalam tahun terakhir, yang bertujuan untuk memungkinkan mesin dapat melakukan pekerjaannya dengan terampil dan mampu melakukan tugas – tugas tertentu 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) Pada Studi Kasus COVID-19” 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 implementasi algoritma *Convolutional Neural Network* dalam mendeteksi penggunaan masker.
2. Berapa besar akurasi yang dihasilkan dari prediksi menggunakan *Convolutional Neural Network*

## 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.*

## Tujuan dan Manfaat Penelitian

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

1. Mengembangkan sebuah sistem deteksi masker menggunakan *Convolutional Neural Network*
2. Sistem ini diharapkan dapat secara akurat dan otomatis mendeteksi apakah seseorang menggunakan masker atau tidak menggunakan masker
3. Untuk mendeteksi mana yang menggunakan masker atau tidak menggunakan masker

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.

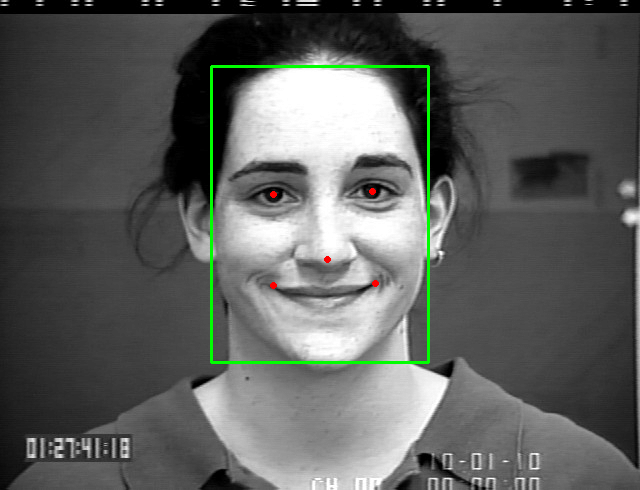
## Multi-Task Cascaded Convolutional Networks

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.



Gambar 2.1 Deteksi dengan MTCNN

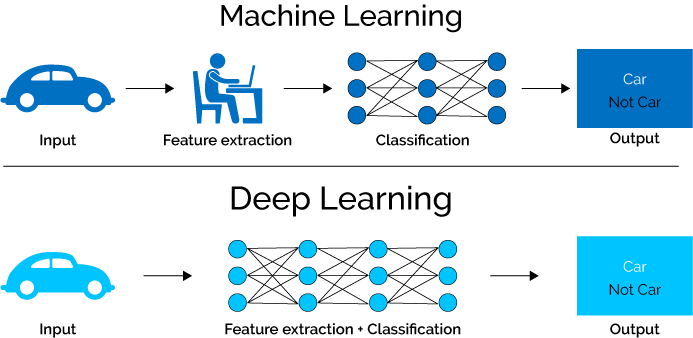
## Sistem Pengenalan Wajah

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

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

## Deep Learning

Mengatakan *Deep Learning* adalah salah satu metode pembelajaran mesin yang menggunakan jaringan saraf dengan banyak layer untuk memproses data dan menghasilkan output yang akurat. Jaringan saraf ini terdiri dari banyak neuron yang saling terhubung dan bekerja secara bersama-sama untuk mengekstraksi fitur dari data yang kompleks. Teknik ini telah digunakan dalam berbagai aplikasi, seperti pengenalan wajah, pengenalan suara, pemrosesan bahasa alami, dan lain-lain (Hao et al., 2016) .



Gambar 2.2 Struktur arsitektur *deep learning*

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

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.

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

## 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.

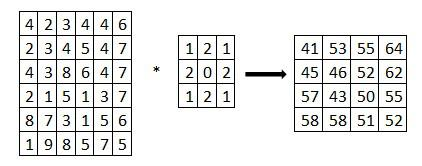


Gambar 2.3 Ilustrasi Arsitektur *Convolutional Neural Network*

### Convolutional Layer

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

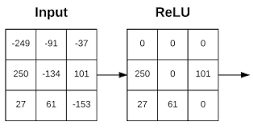
Proses ini ditunjukkan dalam Gambar 1.3, yang merupakan contoh dari layer konvolusi.



Gambar 2.4 Proses *convolutional layer*

### *Rectifield Linear Unit*

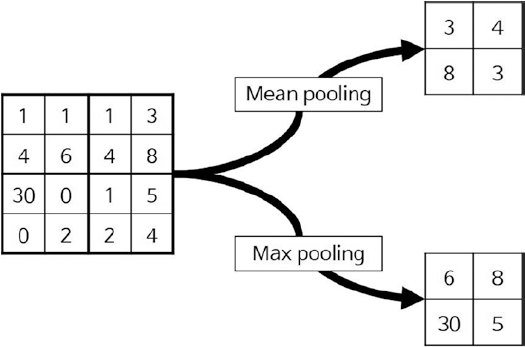
Aktivasi ReLu (*Rectifield Linear Unit)* adalah fungsi aktivasi yang paling umum digunakan pada layer CNN dan jaringan saraf dalam pemrosesan gambar. Fungsi f(x)=max(0,x) ini menghasilkan nilai nol untuk semua nilai input yang negatif, dan menghasilkan nilai input itu sendiri untuk nilai input yang positif. 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 2.5 Contoh Penerapan Aktivasi ReLu

### *Pooling Layer*

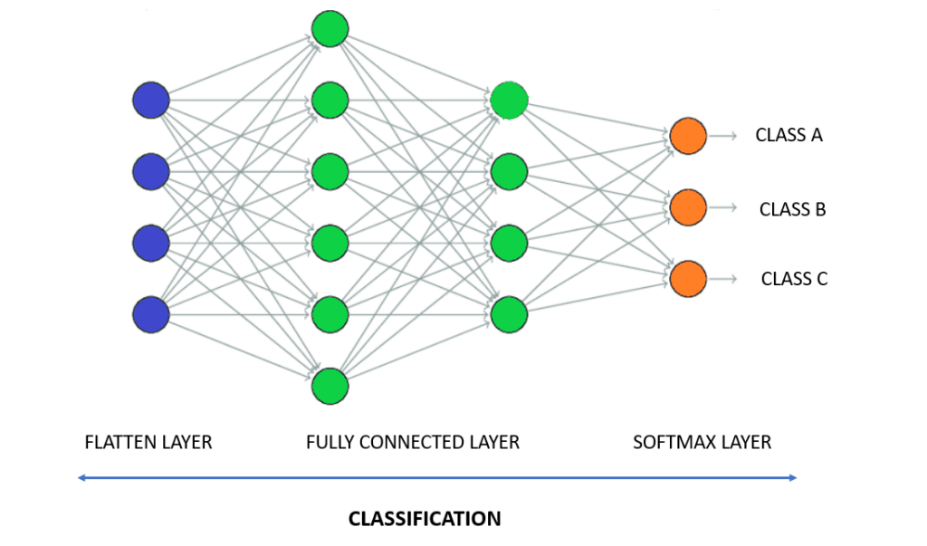
*Pooling Layer* merupakan salah satu tahapan proses dari *convolutional neural network* setelah dari tahap aktivasi biasa disebut *subsampling* atau *downsampling* yang bertujuan untuk mengurangi dimensi dari *feature map* tanpa mengurangi informasi penting di dalamnya. Langkah awal pada layer *pooling* 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 berikut.



Gambar 2.6 Matriks *feature map* 4x4 dengan proses *pooling* 2x2

### *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).

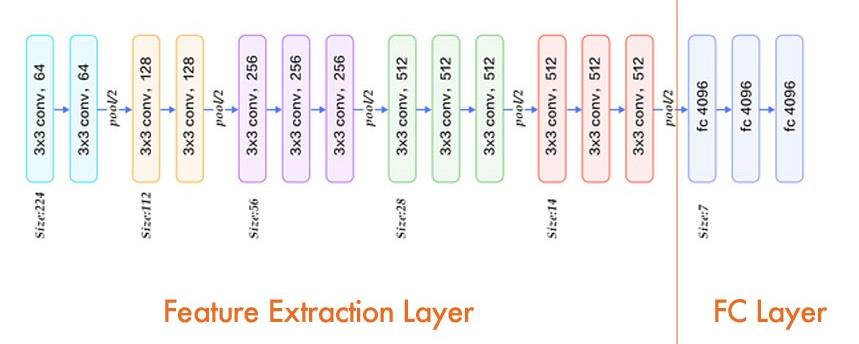


Gambar 2.7 *Fully Connected Layer*

### 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 2.8 Arsitektur jaringan VGG16Net

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)

## Optimizer Adam

Adam adalah sebuah algoritme optimasi gradien stokastik yang sangat popular 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)

## Keras

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

## TensorFlow

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

## 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. false negative (fn)
4. true negative (tn)

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

**Tabel 2.1** *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 mengitung presisi dan secara matematis dapat dirumuskan sebagai berikut:

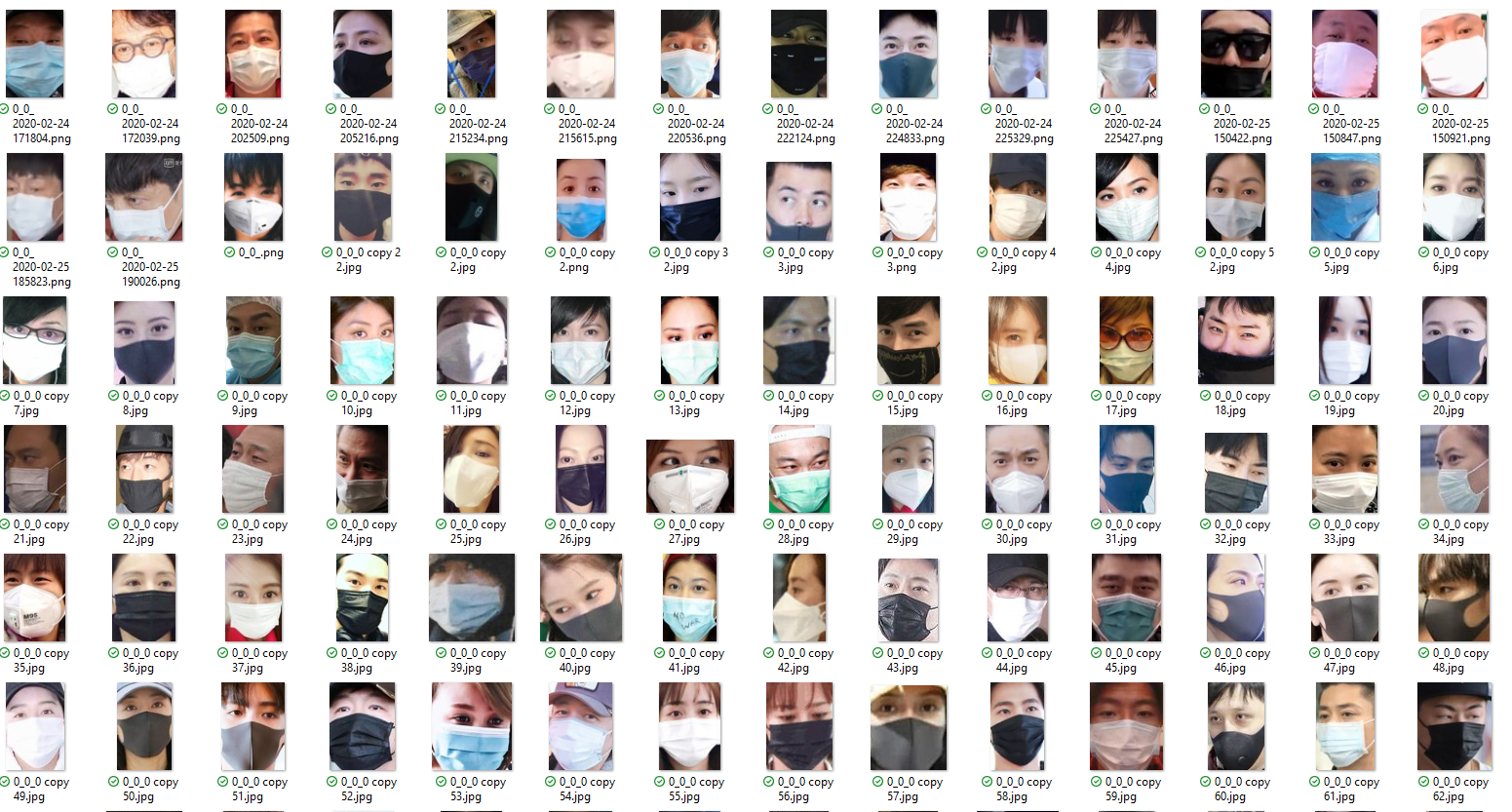
Recall, secara definisi, mengukur seberapa banyak *True Positive* (TP) yang berhasil diidentifikasi oleh model, dibandingkan dengan total jumlah data yang sebenarnya positif. Secara matematis, recall dapat dituliskan dengan rumus sebagai berikut:

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

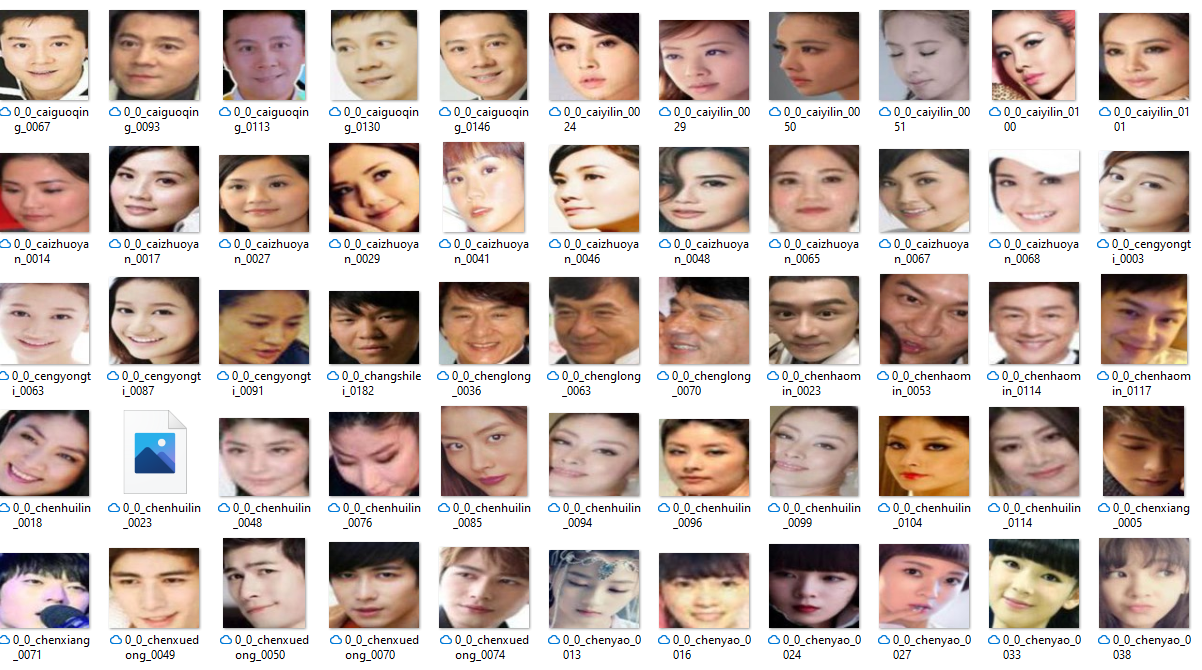
# METODE PENELITIAN

## Deskripsi Data

Data yang digunakan dalam penelitian ini diperolah dari pengumpulan data citra orang yang menggunakan masker dan tidak menggunakan masker, yang merupakan data citra digunakan untuk melatih *machine learning.* Data ini memiliki jumlah 3.833 data citra yang terdiri dari 1.918 tanpa masker dan 1.915 menggunakan masker. Gambar 3.1 merupakan isi dari dataset yang digunakan.



Gambar 3.1 Data penggunaan masker

****

Gambar 3.2 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 dan Visual Studio Code

## Perancangan Sistem

A screenshot of a computer

Description automatically generated with medium confidence

Gambar 3.3 Diagram Perancangan Sistem

Berdasarkan Gambar 3.3 diagram perancangan sistem dibagi menjadi 5 tahap yaitu *Preprocessing, Data Augmentation,* Latih Model CNN, Deteksi Masker dan Hitung Akurasi. Berikut penjelasan masing-masing tahapan tersebut.

### *Data Augmentation*

proses data augmentasi 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*.

### 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.

#### Konversi BGR

Setelah melakukan proses *rezise*, langkah berikutnya adalah melakukan proses konversi yang dimana arsitektur VGG16Net menggunakan saluran warna dalam urutan BGR (*Blue, Green, Red*) sebagai *input*. Jadi, jika awal citra dalam format RGB, maka perlu melakukan konversi saluran warna dari RGB ke BGR.

#### Mean Subtraction

Setelah dilakukan konversi maka nilai dari BGR akan dihitung rata-rata piksel dari setiap saluran BGR tersebut dari kumpulan data pelatihan *VGG16Net*. Setelah dihitung, nilai rata-rata tersebut akan dikurangkan dari setiap piksel citra. Hal ini bertujuan untuk membantu dalam memusatkan data dan menghilangkan komponen cahaya tidak relevan. Dalam tahap ini, citra menjadi *zero-centered*

#### Normalisasi

Setelah memasuki tahap *Mean Subtraction*. 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*”

### 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.

## Desain Antarmuka

Icon

Description automatically generated

Gambar 3.4 Desain Antar Muka

Desain antar muka pada Gambar 3.4 digunakan untuk melakukan pendeteksi uji data citra menggunakan masker dan tidak menggunakan masker. Kemudian pada panel dalam *frame window* langsung menampilkan output dari *webcam* yang menerima input berupa video dan foto secara *realtime* langsung menampilkan tingkat akurasi pada pendeteksian tersebut.

## Skenario Pengujian

Dalam penelitian ini dari jumlah dataset akan dibagi menjadi dua percobaan dari dataset awal berjumlah 3.833 (1.918 tanpa masker dan 1.915 menggunakan masker) dan 800 ( 400 tanpa masker dan 400 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 10-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*, *Gradient Descent*, dan *Root Mean* *Square Propagation* untuk membuat sebuah model CNN yang dibuat panjang iterasi atau epochs 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 secara realtime dengan variasi jenis masker seperti masker beda, masker kain, dan N95.

# HASIL DAN ANALISIS

## Data

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

### *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 :

A black screen with white numbers

Description automatically generated

**Gambar 4.1** Inisialisasi Parameter

**Tabel 4.1** Parameter Skenario Pengujian

|  |  |  |
| --- | --- | --- |
| No. | Parameter | Value |
| 1 | Epoch | 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 4.2** *Code Blok* untuk memuat data citra.

Pada gambar 4.2 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*

(Progress)

Dengan 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. Dalam penelitian ini peneliti melakukan 3 skenario pembagian dataset sebagai berikut

(Progress Revisi)

## Membuat Model Jaringan CNN Dengan arsitektur VGG16Net

**Sebuah gambar berisi teks, cuplikan layar, Font

Deskripsi dibuat secara otomatis**

**Gambar 4.3** Membangun *base model* VGG16Net

Pada gambar 4.5 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 4.4** *Code Blok* Pembuatan Model

Pada Gambar 4.6 *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.

(Progress)

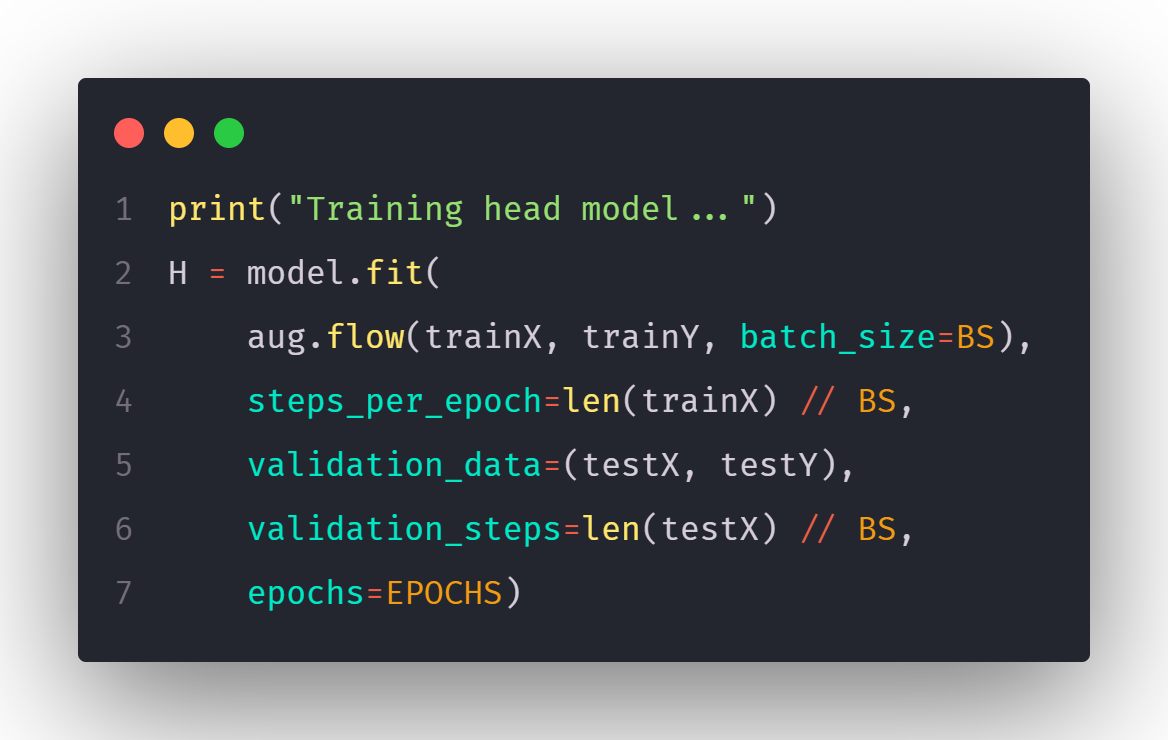
**Gambar 4.5** *Code Blok* Tahap Iterasi

Pada gambar 4.7 dalam tahap ini, pertama melakukan iterasi menggunakan *loop*, setiap lapisan dalam model dasar (VGG16 dalam kasus ini) dinyatakan sebagai tidak dapat diubah atau "*non-trainable*". Ini dilakukan dengan tujuan mematikan pelatihan pada lapisan-lapisan ini sehingga bobot dan bias mereka tidak akan diperbarui selama proses pelatihan berikutnya. Langkah ini mendukung penggunaan model dasar sebagai *ekstraktor* fitur yang stabil. Selanjutnya, kompilasi model dilakukan dengan konfigurasi yang relevan. Sebuah pengoptimalan *Adam* diatur dengan tingkat pembelajaran awal yang telah ditentukan dan tingkat pembelajaran yang berkurang seiring berjalannya *epoch (decay).* Pengoptimalan ini akan mengatur penyesuaian bobot model selama pelatihan. Model tersebut dikompilasi dengan fungsi kerugian *binary cross-entropy* (cocok untuk tugas klasifikasi biner), pengoptimalan yang telah diatur sebelumnya, serta metrik akurasi untuk evaluasi kinerja model. Akhirnya, model yang telah dikompilasi ditampilkan dalam bentuk ringkasan untuk memberikan wawasan tentang struktur keseluruhan model, termasuk lapisan-lapisan yang diikutsertakan, ukuran output, dan jumlah parameter yang dapat diatur.

Semua langkah ini membangun dasar untuk proses pelatihan model dalam penelitian ini, yang kemudian dapat diterapkan pada data pelatihan untuk menghasilkan hasil yang diharapkan sesuai dengan tujuan penelitian.

## Implementasi Tahap Pelatihan Model

Dalam tahap ini model akan dilakukan proses *training* untuk mengetahui seberapa akurasi dalam penelitian ini



**Gambar 4.6** *Code Blok* Pelatihan Model

Pada gambar 4.8 bagian ini, dilakukan pelatihan *head model*. Fungsi *model.fit()* digunakan untuk melatih model yang telah dibuat sebelumnya dengan data pelatihan dan validasi. Data pelatihan (*trainX dan trainY*) diberikan ke dalam generator augmentasi data (*aug.flow()*) dengan ukuran *batch* yang ditentukan (*batch\_size=BS*). Jumlah langkah per *epoch* ditentukan dengan *steps\_per\_epoch=len(trainX)* // *BS*. Data validasi *(testX dan testY)* diberikan langsung ke model untuk evaluasi dan validasi, dan jumlah langkah validasi per *epoch* ditentukan dengan *validation\_steps=len(testX) // BS*.

## Implementasi *Confusion Matrix*

Pada tahap ini, dilakukan visualisasi *confusion matrix*, suatu metode evaluasi performa model klasifikasi. *Confusion matrix* digunakan untuk memahami sejauh mana model dapat mengklasifikasikan data dengan benar. Struktur confusion matrix terdiri dari empat elemen utama, yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN).

A computer screen with many colorful text

Description automatically generated with medium confidence

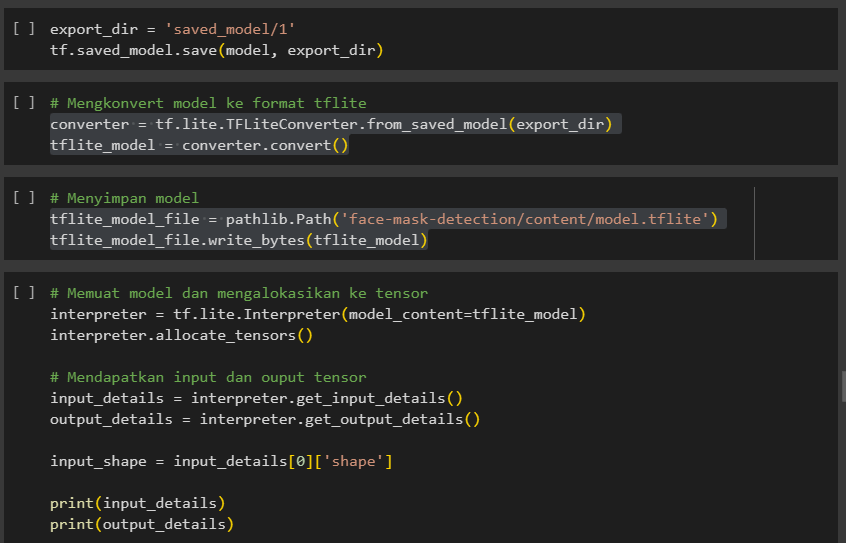
**Gambar 4.7** Implementasi *Confusion Matrix.*

Pada gambar 4.9 Berikut dalam konteks analisis model klasifikasi, konsep-konsep berikut menjadi krusial untuk memahami visualisasi *confusion matrix*. Pertama, istilah "*classes*" merujuk pada daftar kelas atau label yang digunakan dalam model klasifikasi. Selanjutnya, fungsi "*plt.imshow(...)"* dari modul *matplotlib* digunakan untuk menampilkan matriks sebagai gambar dengan skala warna hijau, menciptakan visualisasi yang memperlihatkan distribusi nilai di dalamnya. Untuk memberikan konteks lebih lanjut terhadap visualisasi ini, "plt.title(...)" digunakan untuk menambahkan judul "*Confusion Matrix*" pada plot. Pentingnya interpretasi terhadap nilai-nilai dalam matriks diperkuat dengan penambahan "*plt.colorbar()"* yang memberikan referensi visual berupa bar warna. Sementara itu, definisi label sumbu X dan Y diperoleh melalui "*tick\_marks, plt.xticks(...),* dan *plt.yticks(...)*", memberikan informasi representatif mengenai kelas-kelas pada sumbu X dan Y.

Demi penekanan visual pada nilai-nilai yang signifikan, parameter "fmt dan thresh" digunakan untuk menentukan format dan threshold yang mempengaruhi warna teks pada matriks. Untuk memperkaya informasi, "*plt.text(...)*" menambahkan teks pada setiap sel matriks, menyajikan nilai-nilai konkret di lokasi masing-masing elemen dalam matriks. Agar tata letak visual tetap rapi, "*plt.tight\_layout()*" diaplikasikan, meningkatkan kejelasan visualisasi secara keseluruhan. Akhirnya, label pada sumbu Y dan X diberikan oleh "*plt.ylabel(...)*" dan "*plt.xlabel(...)*", menyediakan panduan yang mempermudah interpretasi terhadap sumbu-sumbu tersebut. Dengan demikian, keseluruhan konsep ini memberikan fondasi yang kuat untuk memahami dan mengevaluasi performa model klasifikasi melalui visualisasi *confusion matrix.*

## Implementasi menyimpan dan konversi model

Pada bagian implementasi menyimpan dan konversi model, dilakukan ekspor (*export*) model ke format *TensorFlow SavedModel* dan konversi ke format *TensorFlow Lite* untuk implementasi di lingkungan perangkat seluler atau sumber daya terbatas lainnya.



**Gambar 4.8** Implementasi menyimpan dan konversi model.

Tahap implementasi menyimpan dan mengonversi model dimulai dengan menentukan direktori di mana model akan disimpan dalam format *TensorFlow SavedModel*. Model tersebut kemudian disimpan di direktori yang telah ditentukan. Setelah itu, dilakukan konversi model tersebut ke dalam format *TensorFlow Lite* menggunakan objek *konverter TensorFlow Lite*. Model yang telah dikonversi disimpan sebagai file .tflite pada lokasi yang telah ditentukan sebelumnya. Selanjutnya, sebuah *interpreter TensorFlow Lite* dibuat untuk model yang baru saja dikonversi, dengan alokasi memori yang sesuai.

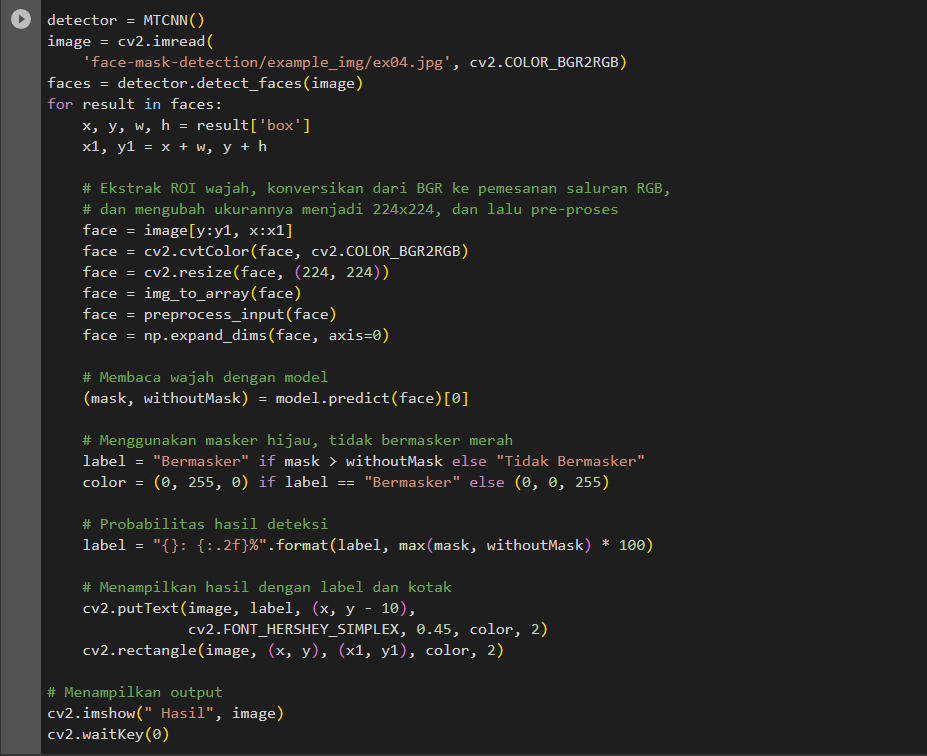
Informasi mengenai detail *input* dan *output* dari model *TensorFlow Lite* diperoleh untuk keperluan analisis lebih lanjut. Terakhir, bentuk (shape) dari input model *TensorFlow Lite* diekstraksi untuk memahami karakteristik input yang diharapkan. Semua langkah ini disajikan secara terstruktur untuk memfasilitasi pemahaman proses implementasi ini pada tahapan penyimpanan dan konversi model.

## Implementasi Pengujian Model dengan MTCNN

Dalam tahap ini, dilakukan implementasi pengujian model deteksi masker menggunakan MTCNN (*Multi-task Cascaded Convolutional Networks*). Pertama, dilakukan inisialisasi detektor wajah menggunakan MTCNN. Selanjutnya, citra yang akan diuji dibaca menggunakan *OpenCV*, dengan konversi warna dari BGR ke RGB. Detektor MTCNN kemudian digunakan untuk mendeteksi wajah pada citra, dan hasil deteksi disimpan dalam variabel *faces*

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

Hasil prediksi kemudian digunakan untuk menentukan label ("Bermasker" atau "Tidak Bermasker") dan warna yang sesuai (hijau untuk bermasker dan merah untuk tidak bermasker). Probabilitas hasil deteksi juga ditampilkan sebagai bagian dari label. Hasil akhir ditampilkan pada citra asli dengan penambahan label dan kotak pembatas berwarna.



**Gambar 4.9** 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 4.2** 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 4.5 berisi parameter-parameter yang digunakan dalam pengujian skenario. Parameter pertama adalah "*Epoch*" dengan nilai 100, yang mengacu pada jumlah iterasi saat melatih model. Parameter kedua adalah "*Target Size*" dengan nilai 224 x 224 RGB, yang menunjukkan dimensi target gambar yang digunakan dalam proses pelatihan. Parameter ketiga adalah "*Batch Size*" dengan nilai 32, yang menunjukkan jumlah sampel yang diproses dalam satu iterasi pelatihan. Parameter terakhir adalah "*Learning Rate*" dengan nilai 0,0001, yang mengindikasikan seberapa besar perubahan bobot model terjadi pada setiap langkah pembelajaran.

**Tabel 4.3** 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 4.4** 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 4.4 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 *VGG16ne*t 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 4.5** 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 4.7 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 skripsi.

Dalam konfigurasi tersebut melibatkan dua skenario *dataset* yang terbagi menjadi dua yaitu *full dataset* dan *half dataset* dengan menggunakan metode *K-fold Cross Validation* yang terdiri dari 3 fold, 5 fold, dan 10 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 Pengujian Data Tunggal