KLASIFIKASI *FACE MASK DETECTION* MENGGUNAKAN *VGG*-16 DENGAN *TRANSFER LEARNING*

TESIS

Diajukan sebagai salah satu syarat untuk memperoleh gelar

Magister Ilmu Komputer (M.Kom)

IRWAN HERLIAWAN

14002397

Program Studi Ilmu Komunikasi (S2)

Univesitas Nusa Mandiri

Jakarta

2021

KLASIFIKASI *FACE MASK DETECTION* MENGGUNAKAN *VGG*-16 DENGAN *TRANSFER LEARNING*

TESIS

Diajukan sebagai salah satu syarat untuk memperoleh gelar

Magister Ilmu Komputer (M.Kom)

IRWAN HERLIAWAN

14002397

Program Studi Ilmu Komunikasi (S2)

Univesitas Nusa Mandiri

Jakarta

2021

SURAT PERNYATAAN ORISINALITAS DAN BEBAS *PLAGIARISME*

Yang bertanda tangan di bawah ini:

Nama	: Irwan Herliawan
NIM	: 14002397
Program Studi	: Ilmu Komputer
Jenjang	: Strata Dua (S2)
Konsentrasi	: *Image Processing*

Dengan ini menyatakan bahwa tesis yang telah saya buat dengan judul: "Klasifikasi *Face Mask Detection* Menggunakan *VGG-16* dengan *Transfer Learning*" adalah hasil karya sendiri, dan semua sumber baik yang kutip maupun yang dirujuk telah saya nyatakan dengan benar dan tesis belum pernah diterbitkan atau dipublikasikan dimanapun dan dalam bentuk apapun.

Demikianlah surat pernyataan ini saya buat dengan sebenar-benarnya. Apabila dikemudian hari ternyata saya memberikan keterangan palsu dan atau ada pihak lain yang mengklaim bahwa tesis yang telah saya buat adalah hasil karya milik seseorang atau badan tertentu, saya bersedia diproses baik secara pidana maupun perdata dan kelulusan saya dari Program Studi Ilmu Komputer (S2) Universitas Nusa Mandiri dicabut/dibatalkan.

Jakarta, 31 Juli 2021
Yang menyatakan

Irwan Herliawan

HALAMAN PERSETUJUAN DAN PENGESAHAN TESIS

Tesis ini diajukan oleh:

Nama	:	Irwan Herliawan
NIM	:	14002397
Program Studi	:	Ilmu Komputer
Jenjang	:	Strata Dua (S2)
Konsentrasi	:	*Image Processing*
Judul Tesis	:	Klasifikasi Face Mask Detection Menggunakan VGG-16 Dengan Transfer Learning

Telah dipertahankan pada periode 2021-1 dihadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh Magister Ilmu Komputer (M.Kom) pada Program Studi Ilmu Komputer (S2) Universitas Nusa Mandiri.

Jakarta, 12 Agustus 2021

PEMBIMBING TESIS

Pembimbing I	:	Dr. Dwiza Riana, S.Si, MM, M.Kom	

D E W A N P E N G U J I

Penguji I	:	Dr. Yan Rianto, M.Eng	
Penguji II	:	Dr. Lindung Parningotan Manik, M.T.I	
Penguji III / Pembimbing I	:	Dr. Dwiza Riana, S.Si, MM, M.Kom	

UNIVERSITAS NUSA MANDIRI	**LEMBAR BIMBINGAN TESIS**
	UNIVERSITAS NUSA MANDIRI

NIM : 14002397
Nama Lengkap : Irwan Herliawan
Pembimbing : Dr. Dwiza Riana, S.Si, M.M, M.Kom
Judul Tesis : Klasifikasi *Face Mask Detection* Menggunakan *VGG-16* dengan *Transfer Learning*

No	Tanggal Bimbingan	Pokok Bahasan	Paraf Dosen Pembimbing
1.	10 April 2021	Bimbingan Perdana dan Pembahasan Judul	
2.	09 Mei 2021	Bimbingan Eksperimen	
3.	31 Mei 2021	Bimbingan Eskperimen Final	
4.	09 Juni 2021	Pengajuan BAB I	
5.	15 Juni 2021	Acc BAB I dan Pengajuan BAB II	
6.	05 Juli 2021	Acc BAB II	
7.	16 Juli 2021	Pengajuan BAB III	
8	22 Juli 2021	Acc BAB III dan Pengajuan BAB IV	
9.	26 Juli 2021	Acc BAB IV dan BAB V	
10.	31 Juli 2021	Revisi dan Acc BAB V	
11.	02 Agustus 2021	Acc Abstrak	
12.	03 Agutus 2021	Acc Keseluruhan	

Catatan untuk Dosen Pembimbing

Bimbingan Skripsi
- Dimulai pada tanggal : 10 April 2021
- Diakhiri pada tanggal : 03 Agustus 2021
- Jumlah pertemuan bimbingan : 12

Disetujui oleh,
Pembimbing

(Dr. Dwiza Riana, S.Si, M.M, M.Kom)

KATA PENGANTAR

Alhamdullillah puji syukur penulis panjatkan kehadirat Allah SWT, yang telah melimpahkan rahmat dan karunia-Nya, sehingga pada akhirnya penulis dapat menyelesaikan laporan Tesis ini tepat pada waktunya. Dimana laporan tesis ini penulis sajikan dalam bentuk buku yang sederhana.

Adapun judul tesis, yang penulis ambil sebagai berikut "Klasifikasi *Face Mask Detection* Menggunakan *VGG-16* dengan *Transfer Learning*". Tujuan penulisan laporan tesis ini dibuat sebagai salah satu untuk mendapatkan gelar Magister Ilmu Komputer (M.Kom) pada Program Studi Ilmu Komputer (S2) Universitas Nusa Mandiri.

Laporan Tesis ini diambil berdasarkan hasil penelitian atau riset mengenai penggunaan masker diera pandemic saat ini dan bagaimana teknologi terutama teknik image processing dapat membantunya dalam meningkatkan keamanan dan kesehatan masyarakat supaya terhindar dari Covid-19. Penulis juga lakukan mencari dan menganalisa berbagai macam sumber referensi, baik dalam bentuk jurnal ilmiah, buku-buku literatur, internet, dan lain-lain yang terkait dengan pembahasan pada laporan tesis ini.

Penulis menyadari bahwa tanpa bimbingan dan dukungan dari semua pihak dalam pembuatan laporan tesis ini, maka penulis tidak dapat menyelesaikan laporan tesis ini tepat pada waktunya. Untuk itu izinkanlah penulis kesempatan ini untuk mengucapkan terimakasih kepada:

1. Rektor Universitas Nusa Mandiri Ibu Dr. Dwiza Riana, S.Si, M.M, M.Kom.
2. Wakil Rektor I Bidang Akademik Universitas Nusa Mandiri.
3. Dekan Fakultas Teknologi Informasi Bapak Anton, M.Kom.
4. Ketua Program Studi Ilmu Komputer Universitas Nusa Mandiri Bapak Dr. Agus Subekti, S.T, M.T.
5. Ibu Dr. Dwiza Riana, S.Si, M.M, M.Kom selaku pembimbing yang selalu mengarahkan dan membimbing dalam melakukan penelitian.

6. Seluruh Dosen Program Studi Ilmu Komputer (S2) Universitas Nusa Mandiri yang telah memberikan pelajaran yang berarti bagi penulis selama menempuh studi.
7. Seluruh staf di lingkungan Universitas Nusa Mandiri yang telah melayani penulis dengan baik selama kuliah.
8. Ayahanda (Bapak Muhtar) dan Ibunda (Sri Hastuti) yang selalu memberikan motivasi dan do'a yang tak terhingga.
9. Winda Apriani S calon istri tercinta yang selalu support menuntaskan kuliah dari kelas XI SMKN 2 Ciamis (2014) sampai S2 Universitas Nusa Mandiri saat ini (2021).
10. Teman-teman satu kelas yang selalu memberikan support dan memberikan waktunya dalam berdiskusi untuk menuntaskan tesis saya.

Serta semua pihak yang terlalu banyak untuk penulis sebutkan satu persatu sehingga terwujudnya penulisan laporan tesis ini. Penulis menyadari bahwa penulisan laporan tesis ini masih jauh sekali dari sempurna, untuk itu penulis mohon kritik dan saran yang bersifat membangun demi kesempurnaan penulisan karya ilmiah yang penulis hasilkan untuk yang akan datang.

Akhir kata semoga laporan tesis ini dapat bermanfaat bagi penulis khususnya dan bagi para pembaca yang berminat pada umumnya.

Jakarta, 31 Juli 2021

Yang menyatakan

Irwan Herliawan

SURAT PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS

Nama	: Irwan Herliawan
NIM	: 14002397
Program Studi	: Ilmu Komputer
Jenjang	: Strata Dua (S2)
Konsentrasi	: *Image Processing*
Jenis Karya	: Tesis

Demi pengembangan ilmu pengetahuan, dengan ini menyetujui untuk memberikan ijin kepada pihak Program Studi Ilmu Komputer (S2) Universitas Nusa Mandiri Hak Bebas Royalti Non-Eksklusif (*Non-exclusive Royalti-Free Right*) atas karya ilmiah kami yang berjudul: "Klasifikasi *Face Mask Detection* Menggunakan *VGG-16* dengan *Transfer Learning*" beserta perangkat yang diperlukan (apabila ada).

Dengan Hak Bebas Royalti Non-Eksklusif ini pihak Universitas Nusa Mandiri berhak menyimpan, mengalih-media atau bentuk-kan, mengelolaannya dalam pangkalan data (*database*), mendistribusikannya dan menampilkan atau mempublikasikannya di internet atau media lain untuk kepentingan akademis tanpa perlu meminta ijin dari kami selama tetap mencantumkan nama kami sebagai penulis/pencipta karya ilmiah tersebut.

Saya bersedia untuk menanggung secara pribadi, tanpa melibatkan pihak Universitas Nusa Mandiri, segala bentuk tuntutan hukum yang timbul atas pelanggaran Hak Cipta dalam karya ilmiah saya ini .

Demikian pernyataan ini saya buat dengan sebenarnya.

Jakarta, 31 Juli 2021
Yang menyatakan,

Irwan Herliawan

ABSTRAK

Nama : Irwan Herliawan
NIM : 14002397
Program Studi : Ilmu Komputer
Jenjang : Strata Dua (S2)
Konsentrasi : *Image Processing*

Judul : "Klasifikasi *Face Mask Detection* Menggunakan *VGG* 16 dengan *Transfer Learning*"

Dunia saat ini sedang mengalami krisis kesehatan karena penyebaran penyakit *Corona Virus Disease* 2019 (Covid-19). Setiap harinya terjadi peningkatan data yang terpapar Covid-19 termasuk di Indonesia. Jika tidak secepatnya diatasi, hal ini akan mengakibatkan ketidakstabilan perekonomian di sebuah Negara. Virus ini menyerang anggota tubuh yaitu pada saluran pernapasan. Sesuai anjuran dari *World Health Organization* (WHO), salah satu perlindungan yang efektif adalah memakai masker ketika berada di kantor, sekolah, tempat belanja, rumah sakit dan tempat umum lainnya. Dalam penelitian ini bertujuan untuk membuat model *deep learning* pada *facemask detection dataset* menggunakan arsitektur *VGG* 16 dengan *transfer learning*. Model tersebut juga diuji dengan tiga *optimizer* yaitu *Root Mean Square Propagation* **(***RMSprop*), *Adaptive Moment Estimation* **(***Adam*) dan *Stochastic Gradient Descent* **(***SGD*) untuk mengetahui model *optimizer* mana yang bekerja dengan optimal. Dari hasil penelitian ini, dapat disimpulkan bahwa model *VGG* 16 dengan *transfer learning* menggunakan *optimizer RMSprop* mencapai akurasi tertinggi yaitu 99%. Model ini dapat di implementasikan di area gedung untuk mendeteksi penggunaan masker, sehingga orang yang terdekat atau pihak keamanan dapat menegur orang yang tidak mematuhi peraturan tersebut.

Kata kunci: *VGG-16, Face Mask, Transfer Learning*

ABSTRACT

Name	: Irwan Herliawan
NIM	: 14002397
Study of Program	: Ilmu Komputer
Levels	: Strata Dua (S2)
Concentration	: *Image Processing*
Titel	: "Klasifikasi *Face Mask Detection* Menggunakan VGG 16 dengan *Transfer Learning*"

The world is currently experiencing a health crisis due to the spread of the Corona Virus Disease 2019 (Covid-19). Every day there is an increase in data exposed to Covid-19, including in Indonesia. If not immediately addressed, this will result in economic instability in a country. This virus attacks the body's limbs, namely the respiratory tract. As recommended by the World Health Organization (WHO), one of the effective protections is to wear a mask when in offices, schools, shopping places, hospitals and other public places. This study aims to create a deep learning model on the facemask detection dataset using the VGG 16 architecture with transfer learning. The model is also tested with three optimizers, namely Root Mean Square Propagation (RMSprop), Adaptive Moment Estimation (Adam) and Stochastic Gradient Descent (SGD) to find out which optimizer model works optimally. From the results of this study, it can be concluded that the VGG 16 model with transfer learning using the RMSprop optimizer achieves the highest accuracy of 99%. This model can be implemented in the building area to detect the use of masks, so that the closest people or security forces can reprimand people who do not comply with these regulations.

Keywords: VGG-16, Face Mask, Transfer Learning

DAFTAR ISI

Halaman

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR LAMPIRAN

Halaman

BAB 1

PENDAHULUAN

1.1. Latar Belakang

Dunia saat ini sedang mengalami krisis kesahatan karena penyebaran penyakit *Corona Virus Disease* 2019 (Covid-19). Menurut situs resmi Covid-19 dari organisasi kesehatan *World Health Organization* (WHO), kasus Covid-19 di dunia (global) terkonfirmasi Covid-19 sebanyak 181.007.816 jiwa dan meninggal sebanyak 3.927.222 jiwa [1], Sedangkan penyebaran virus Covid-19 di Indonesia, menurut situs resmi Covid-19 yaitu Covid19.go.id menyatakan, bahwa di Indonesia sudah terdata sebanyak 2.156.465 jiwa positif Covid-19, sebanyak 1.717.370 terdata sembuh dan sebanyak 58.024 jiwa terdata meninggal karena terpapar Covid-19 [2].

Setiap harinya, terjadi peningkatan data yang terpapar sehingga apabila virus ini tidak cepat ditangani akan mengakibatkan ketidakstabilan perekenomian di sebuah Negara [3]. Pemerintah menganggap bahwa kesehatan masyarakat merupakan prioritas utama di suatu negara dalam kemajuan ekonomi dan pendidikan [4]. Virus Covid-19 ini menyerang anggota tubuh yaitu pada saluran pernapasan. Virus ini dapat menyebar dengan mudah di lingkungan yang ramai dan pada orang yang melakukan kontak dekat [5]. Untuk meminimalisir penyebaran virus Covid-19 ini, salah satunya yaitu dengan menggunakan masker ketika hendak melakukan aktivitas di luar rumah atau tempat umum [6].

Beberapa penelitian sebelumnya mengenai penggunaan masker menggunakan tahapan penelitian sebagai berikut: pengambilan dataset publik, *preprocessing* [7], [4], [8], [9], [10], *p*enggunaan masker sebanyak dua kelas yaitu kelas *with mask* dan *without mask* telah dilakukan oleh Abdellah Oumina [4], Arham, dkk [8], Mohamed Loey, dkk [7], Mohammad Farid G Naufal, dkk [10], Shay E Snider, dkk [6], Truong Quang Vinh [11].

Pembagian dataset telah dilakukan oleh [12]. Selanjutkan dataset dilakukan pelabelan seperti yang sudah dilakukan oleh Mohamed Loey, dkk [7], Gunasekaran Manogaran, dkk [12]. Penelitian menggunakan model *Deep Learning* telah dilakukan oleh [13]. Model *VGG 16* telah digunakan untuk klasifikasi penggunaan masker oleh Darmatasia [9], sedangkan penerapan *transfer*

learning telah digunakan oleh Darmatasia [13], Gunasekaran Manogaran, dkk [12]. Untuk perbandingan *optimizer* telah dilakukan oleh Prillianti, dkk [14].

Penelitian sebelumnya masih menggunakan objek yang hanya terdiri dari dua kelas saja. Selain itu, penelitian sebelumnya juga belum ada yang menggunakan arsitektur *VGG*-16 dengan membandingkan performa *optimizer*, sehingga belum diketahui keakuratan optimizer yang bisa mengklasifikasi penggunaan masker.

Melihat penelitian sebelumnya mengenai klasifikasi penggunaan masker dan tidak menggunakan masker yang hanya berfokus pada dua kelas, dan belum ada perbandingan antara optimizer, maka penelitian ini dilakukan untuk klasifikasi penggunaan masker pada dataset yang terdiri tiga kelas. Dataset ini diambil dari *Kaggle.com* [15]. Untuk itu, penelitian ini mengangkat judul "**Klasifikasi *Face Mask Detection* Menggunakan VGG-16 dengan Transfer Learning**".

1.2. Identifikasi Masalah

Beberapa identifikasi masalah yang terdapat pada penelitian ini berdasarkan uraian di atas:

1) Apakah *VGG 16* dengan *transfer learning* dapat mengklasifikasi dengan baik pada citra yang penggunaan masker.
2) Apakah algoritma *VGG 16* dengan *transfer learning* dapat mengklasifikasi dengan baik pada dataset tiga kelas penggunaan masker.
3) Bagaimana hasil uji menggunakan *optimizer RMSprop Adam* dan *SGD.*

1.3. Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

1) Melakukan klasifikasi menggunakan algoritma *VGG 16* dengan *transfer learning* pada dataset penggunaan masker.
2) Melakukan analisa efektivitas *VGG 16* dengan *transfer learning* pada citra penggunaan masker tiga kelas.
3) Melakukan optimalisasi menggunakan *optimizer Adam, SGD* dan *RMSprop* menggunakan algoritma *VGG 16* dengan *transfer learning.*

Selain tujuan penelitian di atas, maksud penulisan laporan tesis ini dibuat sebagai salah satu syarat untuk mendapatkan gelar Magister Ilmu Komputer (M.Kom) pada Program Studi Ilmu Komputer (S2) di Universitas Nusa Mandiri.

1.4. Ruang Lingkup Penelitian

Agar penelitian ini lebih terarah, maka ruang lingkup penelitian ini sesuai dengan tujuan dan fokus penelitian pada klasifikasi penggunaan masker menggunakan dataset yang diambil dari salah satu penyedia dataset *public* yaitu *Kaggle.com*[15]. Dataset ini bernama *facemask detector dataset* dengan memiliki tiga kelas yaitu kelas w*ith mask*, *without mask* dan *incorrect mask*. Algoritma yang digunakan dalam penelitian ini yaitu algoritma *VGG 16* dengan *transfer learning* dan op*timizer* yang digunakan yaitu *RMSprop , Adam dan SGD. Tools* yang digunakan untuk melakukan *running source code* yaitu *google colaboratory (*akun terbatas*)* [16].

1.5. Sistematika Penulisan

Sistematika penulisan dimaksudkan untuk memberikan gambaran yang jelas mengenai masalah yang akan dibahas. Berikut sistematika penulisan pada penelitian ini.

BAB I PENDAHULUAN

Bab ini berisi penjelasan tentang latar belakang penulisan, identifikasi masalah, tujuan penelitian, ruang lingkup penelitian dan hipotesis, serta sistematika penulisan dari penelitian yang akan disusun.

BAB II LANDASAN TEORI

Bab ini berisi tentang landasan teori yang digunakan dalam melakukan penelitian.

BAB III METODE PENELITIAN

Bab ini berisi tentang metode penelitian yang membahas tentang perancangan penelitian, tahap *computing approach*, serta pengembangan sistem.

BAB IV HASIL DAN PEMBAHASAN

Bab ini berisi tentang hasil dan pembahasan yang menguraikan tentang implementasi sistem pengukuran.

BAB V KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dari hasil penelitian dan saran-saran yang dibutuhkan untuk pengembangan sistem lebih lanjut.

BAB II
LANDASAN PEMIKIRAN

2.1. Tinjauan Pustaka

Penelitian ini dilakukan dengan menggunakan referensi dari berbagai sumber, seperti buku, jurnal, serta literatur yang lain yang digunakan sebagai landasan pemikiran untuk mendukung penelitian tentang penggunaan masker.

2.1.1. Dataset

Dataset merupakan kumpulan data yang berelasi atau berkaitan satu dengan lainnya dalam satu kesatuan yang biasanya bersifat spesifik terhadap suatu kasus tertentu. Dataset dipergunakan sebagai referensi data yang valid untuk suatu penelitian selanjutnya, misalnya untuk referensi data dalam pembelajaran sistem cerdas (sistem pemrosesan citra, pengenalan pola).

Pada penelitian ini, dataset citra yang digunakan merupakan dataset *public* yang diambil dari *kaggle.com* [1] yang bernama bernama *face mask fetector dataset*. Dataset ini terdiri dari tiga kelas yaitu kelas *with mask, kelas without mask,* kelas *incorrect mask*. Berikut tabel spesifikasi dataset citra *face mask detector* [2].

Tabel 2.1 Spesifikasi dataset *face mask detector*

Jenis Kelas	**Jumlah Data**
With mask	686
Without mask	690
Incorrect mask	703
Total	**2079**

Sumber [2]

Dataset citra pada Tabel 2.1 dibagi ke dalam dua bagian yaitu data *training* dan data *testing*. Data *training* adalah seperangkat data awal yang digunakan untuk membantu program memahami bagaimana menerapkan teknologi seperti jaringan saraf untuk belajar dan menghasilkan hasil yang canggih. Data tersebut mungkin dilengkapi dengan set data selanjutnya yang disebut data *validation* dan

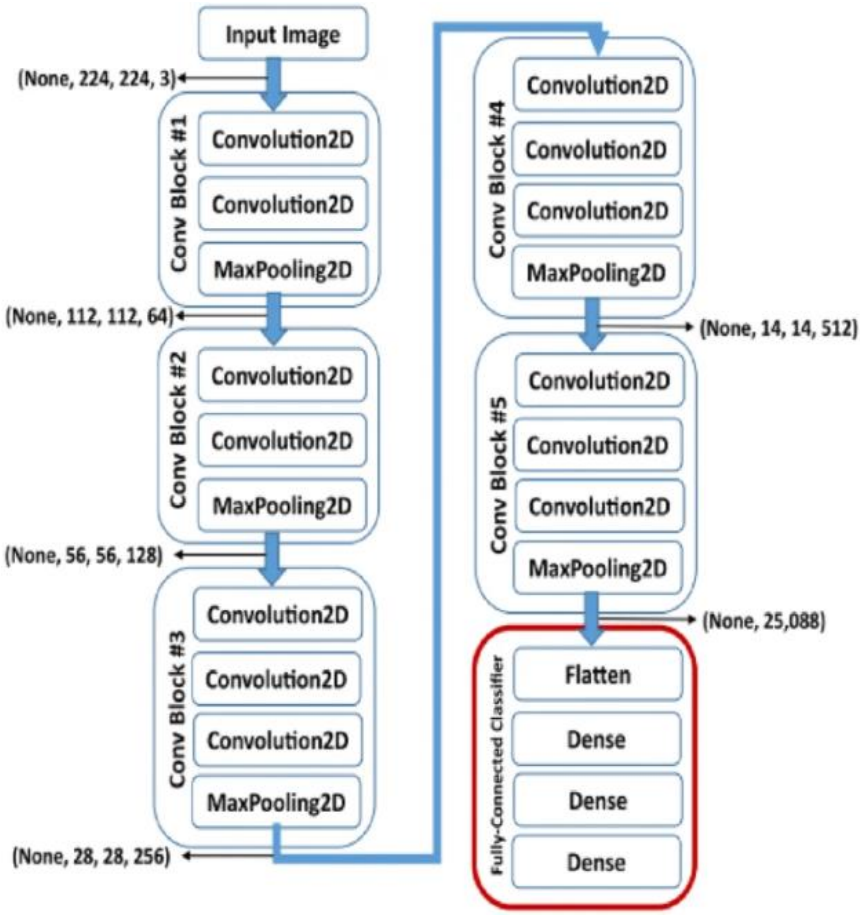
testing [3]. Sedangkan data *training* berisi data baru yang akan diklasifikasikan oleh model yang telah dibuat dan akurasi klasifikasi dievaluasi [4].

2.1.2. *Preprocessing*

Preprocessing citra bertujuan untuk memfasilitasi tahapan pemrosesan selanjutnya dengan memberikan input yang memenuhi asumsi penyederhanaan tertentu [5]. Tahapan pertama dalam *preprocessing* yaitu *resize*. *Resize* merupakan proses merubah luas bidang image menjadi lebih besar atau lebih kecil dari ukuran aslinya, dalam hal ini dengan mengubah ukuran dapat mengakibatkan pergeseran nilai warna dan LSB yang ada sehingga dengan berubahnya nilai parameter tersebut juga mengubah konten digital yang ada didalamnya [6]. Tahap kedua memabandingkan *preprocessing* dengan teknik *augmentasi* citra. *Augemntasi* merupakan suatu teknik untuk menghindari *over fitting* sekaligus meningkatkan akurasi pada penelitian dengan cara memutar citra uji secara *horizontal* maupun vertikal, memiringkan dan membuat bayangan pada citra uji [7].

2.1.3. Visual Geometry Group (VGG-16)

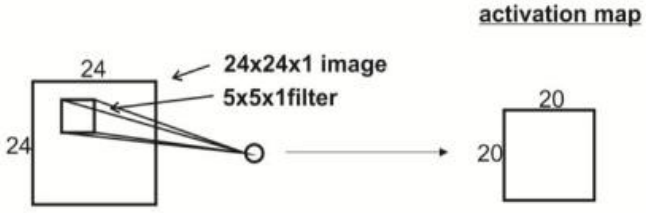
Jaringan *VGG-16* dilatih di database *ImageNet* [8] [9], Karena pelatihan ekstensi yang dilakukan *VGG-16* jaringan telah mengalami, bahkan memberikan akurasi yang sangat baik ketika kumpulan data citra dalam jumlah sedikit. Jaringan *VGG-16* terdiri dari 16 lapisan konvolusi dan memiliki bidang reseptif kecil 3x3. Ini memiliki kumpulan *Max* lapisan ukuran 2x2 dan memiliki total 5 lapisan tersebut. Ada 3 lapisan yang terhubung penuh setelah lapisan penyatuan *Max* terakhir. Ini diikuti oleh tiga lapisan yang terhubung penuh. Ini menggunakan *classifer softmax* sebagai lapisan akhir. Aktivasi *ReLu* adalah diterapkan ke semua lapisa tersembunyi. Skema *VGG-16* arsitektur ditunjukkan pada Gambar di bawah ini.



Gambar 2.1 Arsitektur *VGG16*
Sumber [10]

a. Convolutional Layer

Convolutional layer adalah lapisan pada CNN yang digunakan untuk memperoleh suatu piksel didasarkan oleh nilai pikselnya sendiri. Lapisan ini merupakan lapisan yang paling pertama menerima citra masukan. Lapisan ini terbentuk dari *filter* dengan panjang dan lebar (dalam piksel), serta ketebalan yang sesuai dengan *channel image* [11].Proses konvolusi pada CNN dapat dilihat pada Gambar 2.1.

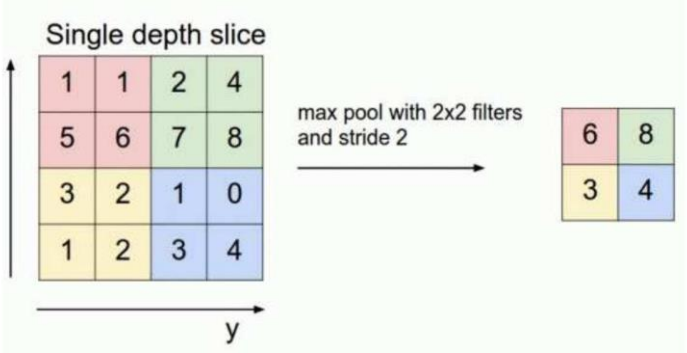


Sumber: Nurfita [12]

Gambar 2.2 Proses Konfolusi

b. Pooling Layer

Pooling layer merupakan lapisan pada CNN yang digunakan untuk downsampling atau mengurangi kompleksitas dari lapisan selanjutnya. Metode *pooling* yang paling umum digunakan adalah *maxpooling* yang akan membagi citra ke dalam wilayah-wilayah kecil. *Pooling* ini memiliki ukuran, biasanya adalah ukuran 2x2 24 [13]. *Pooling layer* digunakan untuk mengurangi dimensi *feature map*. Di dalam *network*, setiap *feature map* yang telah dimasukkan ke dalam *pooling layer* akan diambil sampelnya, jumlah keluaran *feature map* tidak berubah, tetapi ukurannya akan menjadi lebih kecil [14]



Sumber: Albawi [13]

Gambar 2.3. *Pooling Layer*

Pada Gambar 2.3, digambarkan sebuah *pooling* berukuran 2x2. Ketika *pooling* digerakkan, maka *pooling* tersebut akan bergerak ke kanan dan ke bawah untuk mencari nilai terbesar.

c. Fully Connected Layer

Fully connected layer merupakan lapisan akhir dari CNN yang mengambil *imput* dari hasil keluaran pada *pooling layer*. Jika pada *pooling layer* menghasilkan *feature map* dengan bentuk *array* multidimensi, maka pada *fully connected layer*, *array* tersebut akan diubah menjadi sebuah vektor [11]. Setiap simpul pada lapisan akan terhubung penuh secara langsung baik pada lapisan sebelum ataupun setelahnya [13]. *Fully connected layer* hanya dapat diaplikasikan pada akhir jaringan karena adanya transformasi data sehingga menyebabkan hilangnya informasi spasial dan tidak reversibel[15].

2.1.4. Fungsi Aktivasi

Input pada jaringan *neural network* merupakan bagian terpenting, dan akan diubah menjadi hasil *output* yang dikela sebagai aktivasi unit dengan menggunakan fungsi aktivasi. Fungsi aktivasi memiliki fungsi untuk mentransformasikan nilai skalar ke skalar [16]. Beberapa fungsi aktivasi yang sering digunakan adalah sebagai berikut:

***a. Rectified Linier Unit* (ReLU)**

ReLU merupakan fungsi aktivasi non linier yang banyak digunakan pada *neural network*. Keunggulan menggunakan fungsi aktivasi ReLU adalah dengan aktivasi ini neuron-neuron tidak diaktifkan secara bersamaan. Neuron akan dinonaktifkan ketika output dari transformasi linier adalan nol. Secara matematis, fungsi ReLU dapat dituliskan sebagai berikut:

$f(x) = maks\ (0, x)$…….. (1)

b. Sigmoid

Sigmoid merupakan fungsi aktivasi yang paling sering digunakan karena merupakan fungsi non-linear. Fungsi aktivasi *sigmoid* akan mengubah nilai ke dalam rentang 0 hingga 1. Fungsi *sigmoid* terdiferensialkan secara terus menerus dari fungsibentuk S yang halus [16]. Persamaan dari fungsi ini adalah:

$f(x) = \frac{1}{e^{-x}}$…… (2)

Dan diturunkan menjadi,

$f'^{(x)} = 1 - sigmoid(x)$……... (3)

Fungsi *sigmoid* ini dapat memperbaiki masalah pada kemungkinan nilai *output neuron*.

c. Softmax

Fungsi aktivasi *softmax* merupakan kombinasi dari beberapa fungsi *sigmoid*. Fungsi *softmax* dapat digunakan pada klasifikasi *multiclass*. Setiap titik data dari semua kelas akan mengembalikan nilai probabilitas [16]. Ketika kita membangun jaringan atau model untuk klasifikasi *multiclass*, maka lapisan keluaran jaringan akan memiliki jumlah neuron yang sama dengan jumlah kelas pada target.

2.1.5. Transfer Learning

Transfer learning digunakan di mana jaringan yang telah dilatih sebelumnya dilatih menggunakan data dari dataset baru. Manfaat utama *transfer learning* adalah mengurangi waktu yang dibutuhkan untuk mengembangkan dan melatih model dengan menggunakan kembali bobot dari model yang sudah dikembangkan [10].

2.1.6. *Optimizer*

Optimizer yang digunakan dalam penelitian ini yaitu Adam, *RMSprop* dan SGD. Berikut penjelasan mengenai *optimizer*.

a. *Adaptive Moment Estimation* (*Adam*)

Adam merupakan algoritma optimasi, sebagai pengganti klasik sistem penurunan gradien stokastik untuk memperbarui bobot jaringan dalam data pelatihan [17]. Hal Ini digunakan untuk melakukan optimasi dan merupakan salah satu optimasi terbaik saat ini. Adam berasal dari adagrad dan itu adalah pendekatan yang lebih dapat disesuaikan. ADAGRAD dan momentum secara kolektif dikenal sebagai Adam [18].

b. *Stochastic Gradient Descent* (SGD)

Dalam SGD, stokastik menceritakan tentang sistem atau tugas yang terkait dengan acak kemungkinan. Dalam proses ini, alih-alih seluruh kumpulan data, kami memilih beberapa sampel secara acak dari kumpulan data. SGD menghitung gradien parameter hanya menggunakan satu atau kurang[17].

Modifikasi dilakukan pada jumlah data yang digunakan untuk memperbarui parameter internal. Daripada menyertakan seluruh data pelatihan, SGD menggunakan subset data pelatihan (x(i); y(i)) secara bertahap. Ini menghitung satu pembaruan parameter berdasarkan satu subset kecil dari data pelatihan. Itu akan menghitung ulang gradien untuk himpunan bagian lain dan melakukan pembaruan parameter berikutnya. Setiap subset i dipilih secara acak. Oleh karena itu SGD biasanya berjalan lebih cepat dan bisa melakukan pembelajaran online. Pembaruan parameter pada waktu t + 1 dihitung menggunakan persamaan (1), adalah learning rate dan J adalah fungsi error [19]. Pembaruan yang sering di SGD dapat memberikan kemungkinan untuk menemukan minima lokal yang baru

dan lebih baik. Namun, dampak dari seringnya pembaruan adalah kesulitan dalam memperoleh konvergensi [18].

$$\theta_{t+1} = \theta_t - \eta . \nabla_\theta J(\theta; x^{(i)}; y^{(i)}) \quad \text{.....(4)}$$

c. *Root Mean Square Propagation* (*RMSprop*)

RMSprop mirip dengan algoritma penurunan gradien dengan momentum.[17]. RMSProp dikembangkan untuk menyelesaikan masalah Adagrad, hanya seperti yang dilakukan Adadelta. Ini membagi tingkat pembelajaran dengan rata-rata berjalan gradien kuadrat (E[g2]t) yang adalah meluruh secara eksponensial [18]

2.1.7. Confusion Metrix

Confusion matrix memberikan keputusan yang diperoleh dalam *traning* dan *testing*. *confusion matrix* memberikan penilaian *performance* klasifikasi berdasarkan objek dengan benar atau salah. *Confusion matrix* berisi informasi aktual (*actual*) dan informasi terprediksi (*predicted*) pada sistem klasifikasi. *Confusion Matrix* merupakan sebuah metode untuk evaluasi yang menggunakan tabel *matrix* seperti pada Tabel 2.2. Dapat kita lihat dari Tabel 2.2 bahwa jika dataset terdiri dari dua kelas, kelas yang satu dianggap sebagai positif dan yang lainnya negatif. Evaluasi dengan *confusion matrix* menghasilkan nilai *accuracy*, *preecision*, dan *recal* [20].

Tabel 2.2 *Confusion Matrix*

Correct Classification	*Classified as*	
	+	-
+	*True_Positive*	*False_Positive*
_	*False_Positive*	*True_Positive*

Sumber : [21]

True Positive (TP): Mengacu pada tupel positive yang benar diberi label oleh *classifier.*

True Negative (TN): Mengacu pada tupel negative yang benar kemudian diberi label oleh classifier.

False Positive (FP): Tupel negative yang salah diberi label sebagai positive.

False Negative (FN) : Tupel positive yang salah diberi label negative.

Pada tahap evaluasi menggunakan *confusion matrix* akan diperoleh nilai *accuracy*. Akurasi dapat dihitung menggunakan rumus.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad \text{..... (5)}$$

Dimana:

TP : Jumlah kasus positif yang diklasifikasikan sebagai positif

FP : Jumlah kasus negatif yang diklasifikasikan sebagai positif

TN : Jumlah kasus negatif yang diklasifikasikan sebagai negatif

FN : Jumlah kabsus positif yang diklasifikasikan sebagai negatif

2.2. Tinjauan Studi

Terdapat beberapa penelitian klasifikasi penggunaan masker baik penggunaan kelas dataset dan algoritma klasifikasi dan menjadi acuan pada penelitain ini.

1. Pada penelitian [22], menggabungkan 2 dataset penggunaaan masker yang diambil dari dataset public yaitu MMD dan PMK. Kumpulan data yang digabungkan berisi 1415 citra dengan menghapus yang buruk kualitas gambar dan redundansi citra yang buruk secara manual. Dataset memiliki kelas *with mask* dan *without mask*. Proses klasifikasi menggunakan algoritma ResNet50 dengan menerapkan *preporcessing* yaitu *resize* dan augemtasi citra untuk melatih dataset sehingga kemungkinan akurasinya dapat meningkat. *Optimizer* yang digunakan yaitu Adam dan SGDM. Setelah didapatkan akurasinya, peneliti juga mengimplementasikan pada deteksi mengguanakan *Yolo-V2*.

2. Pada penelitian [23], peneliti menggunakan tiga dataset *public* dari *RMFD, LFW* dan *SMFD*. Dataset tersebut memiliki kelas *with mask* dan *without mask*. Algoritma yang diterapkan yaitu model *hybride* atau menggabungkan *model Deep Learning ResNet50* dengan klasifikasi *SVM, Decision Tree* dan *Esemble*. Dari ketiga model klasifikasi tradisional, *SVM* mendapatkan akurasi terbaik.
3. Pada penelitian [24], peneliti menggunakan dataset publik dengan jumlah 1000 citra dan memiliki dua kelas yang terdiri dari kelas with mask (500) dan without mask (500). Dalam pengujiannya, peneliti menggunakan empat skenario pembagian dataset yaitu 80%:20% dan 50 epoch (data latih:data uji), 90%:10% 50 epoch, 80%:20 100 epoch, 90%:10% 100 *epoch*. Hasil penelitian ini, skenario pembagian dataset 90%:10% dengan jumlah epoch 50 mendapatkan nilai terbaik dalam akurasinya.
4. Pada penelitian [25], peneliti menggunakan dataset *public* yang terdiri dari dua kelas yaitu kelas *with mask* (3.725 citra) dan kelas *without mask* (3.828 citra). Tahap *Preprocessing* dilakukan *resize* kedalam ukuran 64x64 *pixels*. Algoritma yang digunakan yaitu *KNN, SVM* dan *CNN* dengan menguji data train dan data testing. Hasil penelitian ini KNN dan *SVM* mendapatkan waktu tercepat dibandingkan dengan *CNN*, akan tetapi *CNN* mendapatkan akaurasi yang terbaik dengan proses eksekusi yang lama dibandingkan dengan *KNN* dan *SVM*.
5. Pada penelitian [26], peneliti menggunakan dataset publik sebanyak 1376 citra yang memiliki 2 kelas yaitu *face with mask* (690 citra) dan *face without mask* (686 citra). *Preprocessing* yang dilakukan yaitu augmentasi yang terdiri dari tahap *cropping*, *padding* dan *horizontal flipping*. Algortima yng digunakan yaitu *MobileNet-V2, VGG19, Xception* dan *SVM, KNN* sebagai *classifier*. Hasil dari penelitian ini, model agloritma *MobileNetV2* dengan *SVM Classifier* mendapatakan angka *recall* dan *presicion* yang tertinggi.
6. Pada penelitian [27], peneliti menggunakan dataset *private* yang dikumpulkan dari *google* yang terdiri dari 3 kelas yaitu menggunakan masker, tidak menggunakan masker dan menggunakan masker secara tidak benar. Setiap kelas mempunyai 125 citra. *Preprocessing* yang dilakukan

dalam penelitian ini meliputi ekstrak Region of Interest (ROI) yang dilakukan secara manual dengan mengambil area wajah dan *resize* citra untuk mengurangi dimensi data tanpa menghilangkan informasi penting. Algoritma yang digunakan yaitu *VGG 16, ResNet101V2, Inception V3, MobileNet, DenseNet201,NASNetLarge.* Peneliti mengatur learning rate 0.001, *batch size* 8, dan jumlah *epoch* 25. *Ativation* yang digunakan *ReLu*, *optimizer* Adam dan *loss fuction* menggunakan *categorical_crossentropy*. Untuk menghindari *over fitting*, diguanakn dropout 0.5. Model dilatih dengan pembagian dataset 80:20 dan diimplementasikan menggunakan *library* Keras. Hasil dari penelitian ini bahwa inception V3 memiliki performa yang baik berdasarkan nilai akurasi, *presisi*, *recall*, dan F1 *score* pada dataset yang digunakan.

7. Pada penelitian [28], peneliti menggunakan dataset yang diperoleh dari google. citra terdiri dari dua kelas yaitu kelas menggunakan masker, tidak menggunakan masker dan menggunakan masker tidak benar. Preprocessing yang dilakukan yaitu perolehan *ROI* pada area wajah secara manual dan *resize* ukuran citra. Ukuran citra diatur menjadi 224x224. Model yang digunakan yaitu *Xception Transfer Learning*. Hasil dari penelitian ini model mampu membedakan yang menggunakan masker dan tidak menggunakan masker dilihat dari *confusion matrix*.

Tabel 2.3 Penelitian Sebelumnya

No	Judul penelitian	Peneliti	Akreditasi	Algoritma	Perbedaan
1	*Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection*	[22]	Q1 H-Index 61	Melakukan klasifikasi menggunakan metode ResNet 50 dan Yolo V2 sebagai detector. (2021)	Menggunakan 2 kelas, yaitu menggunakan masker dan tidak menggunakan masker dengan total data 1415 citra.
2	*A Hybrid Deep Transfer Leaning Model with Machine Learning Methods for Face Mask Detection in the Era of the Covid Pandemic*	[23]	Q1 H-Index 762	Pada penelitian ini menggunkana algoritma ResNet50 dan SVM, Decision Tree,	Penelitian ini menggunakan dataset public dari RMFD,LFW dan SMFD. Masing-masing

				dan Esenmble. (2021)	dataset terdiri dari dua kelas yaitu face mask dan without mask.
3	*Convolutional Neural Network (CNN)* untuk Klasifiikasi Penggunaan Masker	[24]	S3 H-Index 10	Melakukan klasifikasi menggunakan model CNN (2020)	Menggunakan dataset yang berjumlah 2 kelas yaitu menggunakan masker dan tidak menggunakan masker, dataset yang digunakan sebanyak 1000 citra
4	*Comparative Analysis of Image Classification Algorithms for face mask detection*	[25]	S2 H Index 13	Menggunakan kalsifikasi dengan algoritma KNN, SVM dan CNN. (2021)	Penelitian ini menggunakan dataset private yang memiliki dua kelas yaitu menggunakan masker dan tidak menggunakan masker.
5	*Control the Covid-19 Pandemic: Face Mask Detection Using Transfer Learning*	[26]	International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS)	Penelitian ini menggunakan algoritma VGG19, MobileNetV2 dan Xception dan menerapkan detection (2020)	Penelitian ini menggunakan dataset yang berjumlah 1376 yang terdiri dari 2 kelas yaitu menggunakan masker dan tidak menggunakan masker.
6	Analisa Perbandingan Performa Model Deep Learning untuk Mendeteksi Penggunaan Masker	[27]		Melakukan klasifikasi menggunakan algoritma ResNet101V2, InceptionV3, MobileNet, DesNet201, VGG 16 dan NASNetLarge. (2020)	Menggunakan dataset yang berjumlah 3 kelas yaitu menggunakan masker, tidak menggunakan masker, menggunakan masker tidak benar. Dataset yang diuji berjumlah 125 citra.
7	Deteksi Penggunaan Masker Menggunakan *Xception Trnasfer Learning*	[28]	S5 H-Index 8	Melakukan klasifikasi menggunakan algortima Xception transfer	Penelirian ini menggunakan dataset yang berjumlah 125 citra dengan kelas

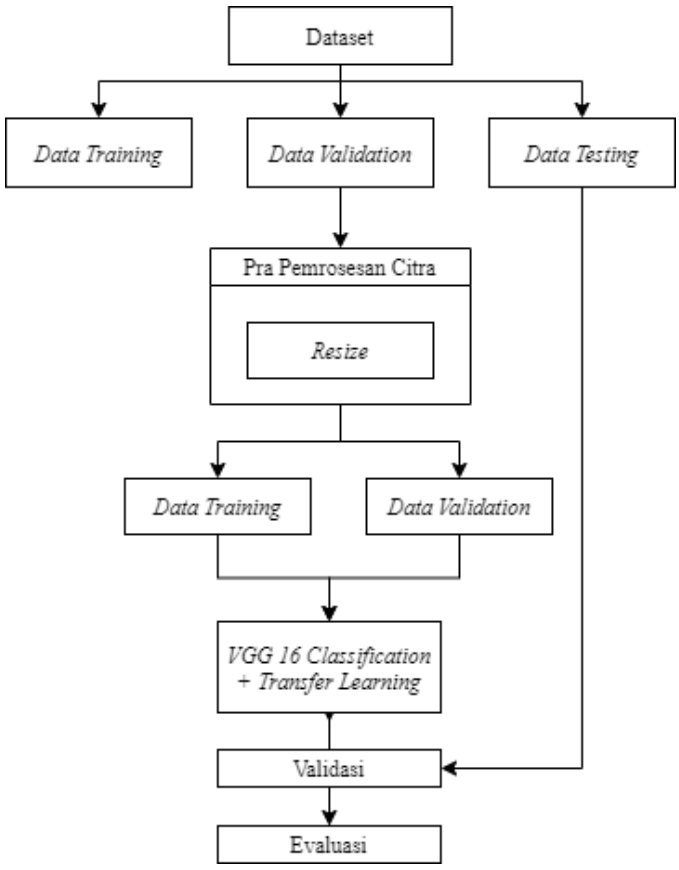
				Learning (2020)	menggunakan masker dan tidak menggunakan masker.

BAB III
METODOLOGI PENELITIAN

Pada bab ini menjelasakan mulai dari perancangan penelitian, dataset, *preprocessing*, metode yang diusulkan, eksperimen dan pengujian model dan evaluasi dan validasi.

3.1. Perancangan Penelitian

Penelitian memiliki fungsi dalam mencarikan penjelasan dan jawaban terhadap permasalahan serta memberikan alternatif bagi kemungkinan yang dapat digunakan untuk pemecahan masalah [1]. Dalam melakukan penelitian diperlukan model penelitian untuk menghasilkan pengetahuan baru yang ingin dicapai. Berikut pada Gambar 3.1 adalah model penelitian yang diterapkan.

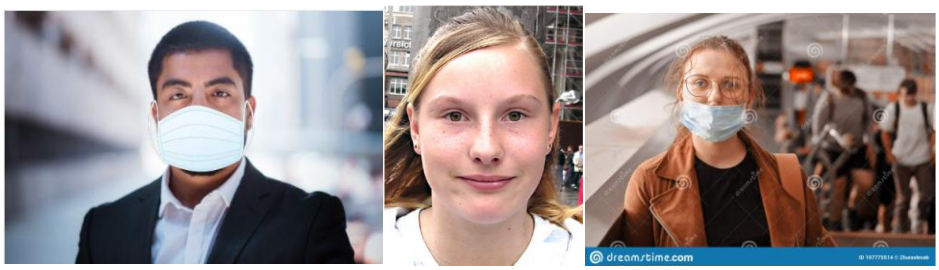


Sumber: Hasil Penelitian (2021)

Gambar 3.1 Tahapan Penelitian

3.1.1. Dataset

Objek pada penelitian ini adalah citra penggunaan masker yang diambil dari *kaggle.com* yang bernama *face mask detector*. Dataset ini memiliki tiga kelas yaitu kelas *with mask, without mask* dan *incorrect mask* [2]. Dataset ini sebelumnya telah digunakan dalam penelitian Mira M. Bpulos [3] dengan hasil akurasi sebesar 97%. Detail dataset ini memiliki tiga kelas yaitu menggunakan masker (with mask) sebanyak 690 citra, tidak menggunakan masker (*without mask*) sebanyak 686 citra dan menggunakan masker tidak benar (*incorrect mask*) sebanyak 703 citra. Total keseluruhan citra yaitu 2079 citra. Berikut contoh citra dari setiap kelas.



a b c

Sumber : [2]

Gambar 3.2 (a) *with mask*, (b) *without mask*, (c) *incorrect mask*

Pada Gambar 3.2 menunjukan contoh citra *face mask detector dataset* yang terdiri dari masing-masing kelas yaitu kelas *with mask* (Gambar 3.2.a), kelas *without mask* (Gambar 3.2.b) dan *kelas incorrect mask* (Gambar 3.2.c).

Pengolahan data awal pada penelitian ini adalah melakukan pembagian dataset dengan perbandingan 90:10. Pembagian data ini bertujuan untuk meningkatkan akurasi [4].

3.1.2. *Preprocessing*

Pre-processing yang dilakukan dalam penelitian ini terdiri yaitu *resize* yang bertujuan untuk mengurangi dimensi data tanpa menghilangkan informasi penting dari data [5]. Ukuran citra yang digunakan adalah 224x224.

3.1.3. Metode yang Diusulkan

Pada penelitian ini dilakukan klasifikasi data penggunaan masker dengan menggunakan arsitektur *VGG 16* dengan *transfer learning*. Proses pada penelitian ini diimplementasikan menggunakan *google colaboratory* [6] yang merupakan fasilitas dari *google* untuk keperluan penelitian. *Google colab*oratory ini hanya bisa diakses secara *online* dan waktu *running* penelitian terbatas untuk akun bukan premium.

3.1.4. Eksperimen dan Pengujian Model

Pada tahap eksperimen dan pengujian model melalui beberapa tahapan diantaranya, *resize*, mengatur *batch size*, membuat model *VGG 16* dengan *transfer learning*, mengatur *optimizer*, menajalankan source code dan menampilkan penelitian dalam bentuk akurasi, grafik, *confusion matrix* dan *classification report*.

1. Melakukan ***resize*** seluruh data ke ukuran 224x224.
2. Pelabelan citra.
3. Mengatur *batch size* adalah banyaknya citra yang dimasukan dalam setiap *steps* training dengban nilai 254.
4. Tahap kelima membuat model *VGG 16* dengan *Transfer Learning* dengan detail berikut:
    a. Mengatur *Weight* adalah bobot dari masing-masing layer yang sudah di training berdasarkan bobot *imagenet*.
    b. *Include top* adalah perintah untuk menyertakan apakah model yang digunakan akan disertakan dengan *top layer* dari arsitektur *network* tersebut atau misalkan top *network* tersebut memiliki 3 node yaitu : *flatten layer*, *layer* dengan 1024 node dan fungsi aktivasinya, lalu *prediction* node dimana jumlah node sesuai dengan banyaknya jumlah kelas. Dikarenakan jumlah kelas pada *imagenet* adalah 1000 maka ini tidak cocok dengan jumlah kelas yang kita miliki yaitu hanya 4. sehingga

include top di sini adalah *false.*

c. Mengatur *activation ReLu*

d. Mengatur *activation* dengan *softmax.*

e. Mengatur *epoch* dengan nilai 50 *epoch*.

f. Mengatur *optimizer* dengan *Adam, SGD* dan *RMSprop.*

5. Menjalankan seluruh *soruce code.*

6. Menampilkan hasil penelitian dalam bentuk akurasi, grafik, *confusion matrix* dan melakukan *test* atau pengujian model.

3.1.4. Evaluasi dan Validasi

Pada tahap evaluasi dan validasi ini dilakukan pengujian dan pemrosesan yang sama seperti pada data *training* terhadap citra baru atau data *testing* untuk mengetahui seberapa besar keakuratan aplikasi yang telah dibuat menggunakan *google colaboratory* dalam klasifikasi penggunaan masker sesuai dengan eksperimen yang dilakukan dan algoritma yang dipilih.

3.2. Perangkat Pendukung

Dalam melakukan penelitian ini dibutuhkan *tools* atau alat penunjang baik perangkat keras maupun perangkat lunak untuk melakukan akuisisi citra dan pemrosesan citra hingga menghasilkan tujuan dari penelitian ini. Berikut adalah spesifikasi *hardware* yang dibutuhkan untuk melakukan penelitian ini.

1. Spesifikasi Perangkat Keras

a. *CPU* / Komputer

1) Processor Intel® Pentium® Dual Core

2) *RAM* DDR III 4 GB

3) *Hard Disk* 500 GB

4) *VGA* 1,5 GB

b. Monitor dengan resolusi layar minimum 1024x768

2. Spesifikasi Perangkat Lunak

a. Sistem operasi yang umum digunakan seperti: *Microsoft Windows* atau *Linux* (*Ubuntu, Fedora*, dan lain-lain), dianjurkan *Microsoft Windows.*

b. Goolge Collaboratory

Google collaboratory merupakan sebuah e*xecutable document* yang dapat digunakan untuk menyimpan, menulis, serta membagikan program yang telah ditulis melalui *Google Drive.*

Pada dasarnya, *software* ini dibangun serupa dengan Jupyter Notebook gratis *cloud* yang dijalankan menggunakan *browser,* seperti *Mozilla Firefox* dan *Google Chrome.*

c. Goolgle Drive

Google drive merupakan layanan penyimpanan data tersinkronisasi yang dikembangkan oleh Google. Diluncurkan pada tanggal 24 April 2012, Google Drive memungkinkan penggunanya untuk menyimpan data di server mereka, mensinkronisasi data di perangkat yang berbeda, dan saling berbagi berkas [7].

BAB IV

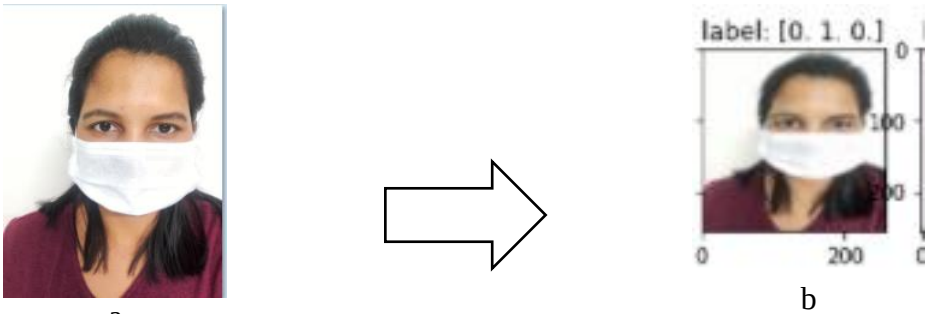
HASIL PENELITIAN

Pada bagian ini membahas hasil penelitian mulai dari hasil *resize*, hasil pelabelan data, dan membandingkan model *VGG 16* dengan *transfer learning* menggunakan tiga *optimizer* yaitu *RMSprop, Adam* dan *SGD.*

a. *VGG 16* dengan *transfer learning* menggunakan *optimizer RMSprop.*
b. *VGG 16* dengan *transfer learning* menggunakan *optimizer Adam.*
c. *VGG 16* dengan *transfer learning* menggunakan *optimizer SGD.*

4.1. Citra Hasil *Resize*

Pada tahap pertama dilakukan *resize* atau merubah ukuran cittra kedalam ukuran 224x224. Berikut citra hasil *resize* dibawah ini.

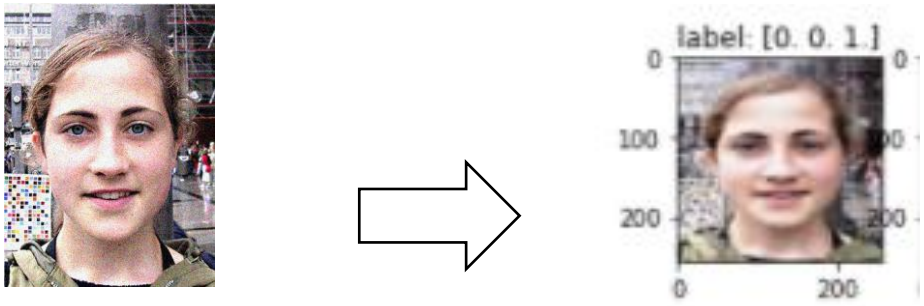


a
Sumber [1]

b
Sumber: Penelitian (2021)

Gambar 4.1 Citra asli (a) dan citra hasil *resize* (b) pada kelas *with mask*

Pada gambar 4.1 menunjukan bahwa Gambar 4.1. (a) merupakan contoh citra asli dari kelas *with mask* dan Gambar 4.1. (b) merupakan contoh citra hasil *resize* dengan ukuran 224x224 pada kelas *with mask*. Citra hasil *resize* menunjukan adanya pengurangan dimensi dari citra asli. Citra yang ditampilkan dipilih secara acak oleh sistem.



a

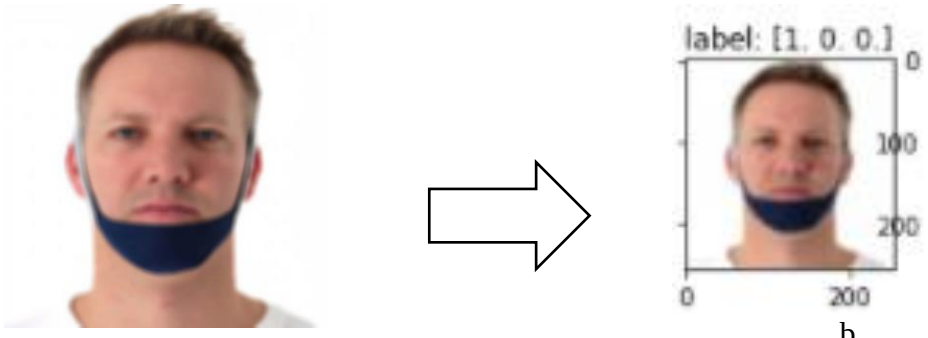
Sumber [1]

b

Sumber: Penelitian (2021)

Gambar 4.2 Citra asli (a) dan citra hasil *resize* (b) pada kelas *without mask*

Pada Gambar 4.2 menunjukan bahwa Gambar 4.2.a merupakan contoh citra asli dari kelas *without mask* dan Gambar 4.2.b merupakan contoh citra hasil *resize* dengan ukuran 224x224 dari kelas *without mask*. Citra hasil *resize* menunjukan adanya pengurangan dimensti citra. Citra yang ditampilkan dipilih secara acak oleh sistem.



a

Sumber [1]

b

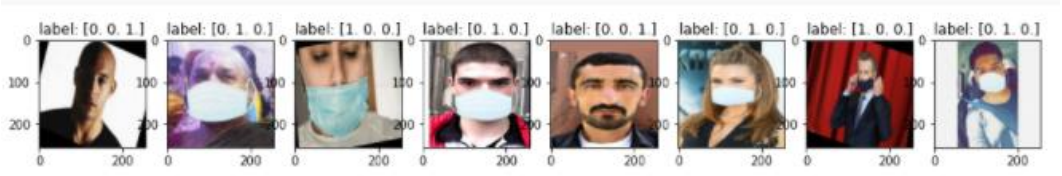
Sumber: Penelitian (2021)

Gambar 4.3 Citra asli (a) dan citra hasil *resize* (b) pada kelas *incorrect mask*

Pada Gambar 4.3 menunjukan bahwa Gambar 4.3.a merupakan contoh citra asli dari kelas *incorrect mask* dan gambar 4.3.b merupakan contoh citra hasil *resize* dengan ukuran 224x224 dari kelas *incorrect mask*. Citra hasil *resize* menunjukan adanya pengurangan dimensi citra. Citra yang ditampilkan dipilih secara acak oleh sistem.

4.2. Citra Hasil Pelabelan

Citra Hasil pelabelan dataset ditampilkan sesuai dengan kelasnya diantranya *with mask*, *without mask* dan *incorrect mask*. Berikut contoh citra hasil pelabelan yang dipilih secara acak oleh sistem dengan menampilkan empat baris dan delapan kolom.

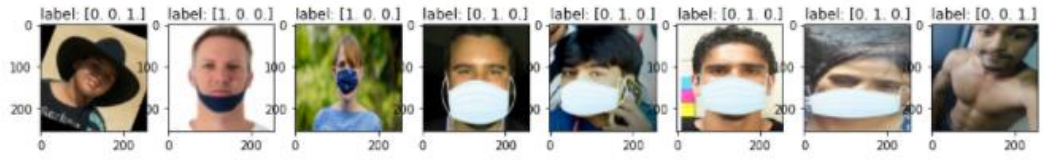


a b c d e f g h

Sumber: Penelitian (2021)

Gambar 4.4 Citra hasil pelabelan baris ke satu. Citra (a,e) kelas *without mask*, citra (b,d,f,h) kelas *with mask* dan citra (c,g) kelas *incorrect mask*

Pada Gambar 4.4 menunjukan hasil pelabelan citra pada setiap kelas yang berada di baris ke satu yang dipilih secara acak. Citra (a,e) menunjukan kelas *without mask* dengan label 0.0.1. Citra (b, d, f, h) menunjukan kelas *with mask* dengan lebel 0.1.0. Citra (c, g) menunjukan kelas *incorrect mask* dengan label 1.0.0. Pada baris pertama ini, citra yang paling banyak ditampilkan yaitu citra kelas *with mask* degan label 0.1.0.



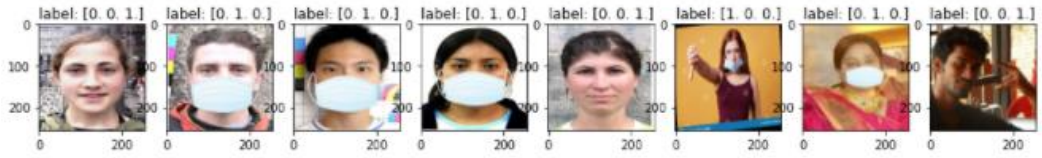
a b c d e f g h

Sumber: Penelitian (2021)

Gambar 4.5 Citra hasil pelabelan baris kedua. Citra (a, h) kelas *without mask*, citra (b) kelas *incorrect mask* dan citra *(c,d,e,f,g)* kelas *with mask*

Pada Gambar 4.5 menunjukan hasil pelabelan citra pada setiap kelas yang berada pada baris kedua yang dipilih secara acak. Citra (a, h) menunjukan kelas *without mask* dengan label 0.0.1. Citra (b) menunjukan kelas *incorrect mask*

dengan label 1.0.0. Citra (c, d, e, f, g) menunjukan kelas *with mask* dengan label 0.1.0. Pada baris kedua ini, citra yang paling banyak ditampilkan yaitu citra kelas *with mask* dengan kelas 0.1.0.



a b c d e f g h

Sumber: Penelitian (2021)

Gambar 4.6 Citra hasil pelabelan baris ketiga. Citra (a,e,h) kelas *without mask*, citra (b, c, d, g) kelas *with mask* dan citra (f) kelas *incorrect mask*.

Pada Gambar 4.6 menunjukan hasil pelabelan citra pada setiap kelas yang berada pada baris ketiga yang dipilih secara acak. Citra (a, e, h) menunjukan kelas *without mask* dengan label 0.0.1. Citra (b,c,d,g) menunjukan kelas *with mask* dengan label 0.1.0. Citra (f) menunjukan kelas *incorrect mask* dengan label 1.0.0. Pada baris ketiga ini, citra yang paling banyak ditampilkan yaitu citra kelas *with mask* dengan kelas 0.1.0.

a b c d e f g h

Sumber: Penelitian (2021)

Gambar 4.7 Citra hasil pelabelan baris keempat. Citra (a, b, g) kelas *with mask*, citra (c, e, f) kelas *incorrect mask* dan citra (d,h) kelas *without mask*

Pada Gambar 4.7 menunjukan pelabelan citra pada setiap kelas yang berada pada baris keempat yang dipilih secara acak. Citra (a, b, g) menunukan citra dengan kelas *with mask* dengan label 0.1.0. Citra (c, e, f) menunjukan gambar dengan kelas *incorrect mask* dengan label 1.0.0. Citra (d, h) menunjukan kelas *without mask* dengan label 0.0.1. Pada baris keempat ini, citra yang paling banyak

ditampilkan yaitu citra dengan kelas *with mask* dengan label 0.1.0 dan citra kelas *incorrect mask* dengan label 1.0.0.

4.3. Hasil Perbandingan *VGG 16 Transfer Learning* dengan *Optimizer* (*RMSprop, Adam, SGD*)

Hasil perbandingan arsitektur *VGG 16 transfer learning* dengan *optimizer RMSprop, Adam dan SGD,* masing-masing ditampilkan dalam bentuk akurasi, grafik akurasi, grafik *loss*, *classification report, confusion metrix* dan *test* model atau pengujian:

4.3.1. Hasil Eksperimen *VGG 16 Transfer Learning* Menggunakan *Optimizer RMSprop*

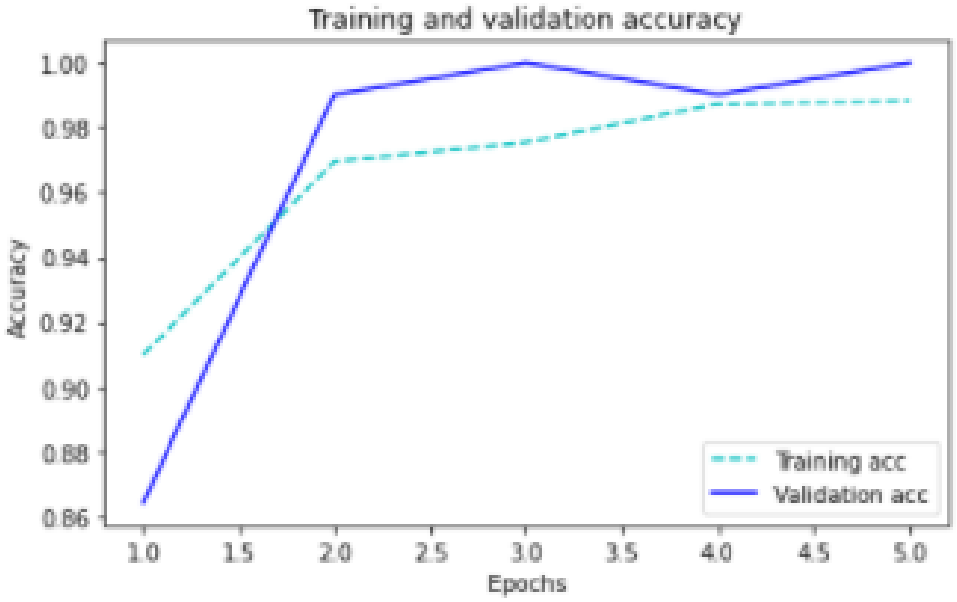
Hasil penelitian menggunakan *optimizer RMSprop* ini ditampikan dalam bentuk akurasi, grafik akurasi dan *loss*, *classification report, confusion metrix*, dan *test* model atau pengujian:

A. Akurasi

   Dengan model *VGG 16 trasnfer learning* menggunakan *optimizer RMSprop* dan diatur 50 *epoch* menghasilkan rata-rata akurasi sebesar 99% dan *loss* 0.06%.

B. Grafik

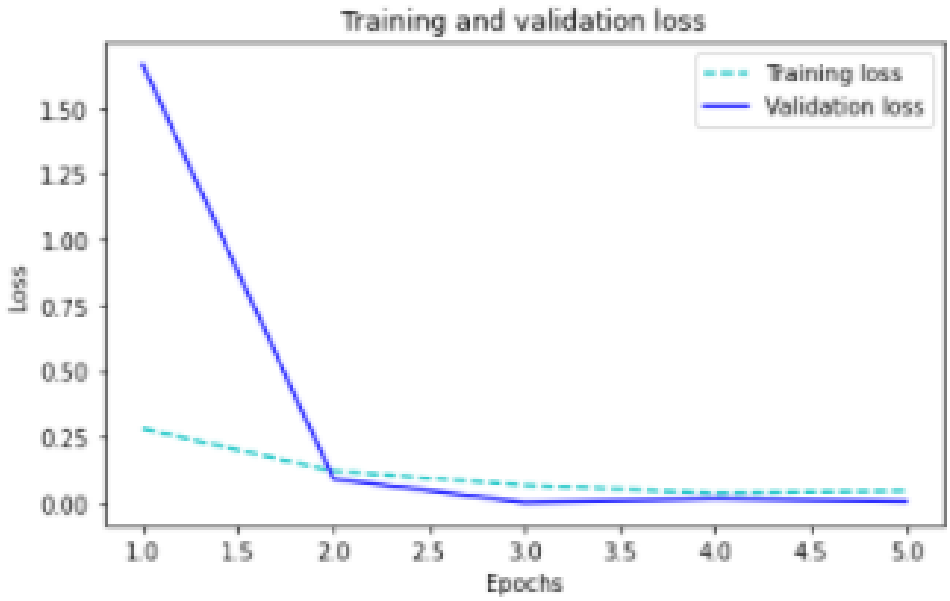
   Berikut grafik akurasi dan *loss* yang dihasilkan oleh model *VGG 16* dengan *transfer learning* menggunakan *optimizer RMSprop.*



Sumber: Penelitian (2021)

Gambar 4.8 Grafik akurasi model *VGG 16 transfer learning* menggunakan *optimizer RMSprop*

Dari Gambar 4.8 menunjukan bahwa hasil grafik akurasi yang diuji terhadap data *training* dan data *validation*. Garis yang berwarna biru merupakan tanda grafik akurasi data *validation* yang menunjukan kenaikan mulai dari *epoch* 10-50. Sedangkan garis berwarna hijau merupakan data *train* yang menunjukan terjadinya kenaikan yang konsisten. Dari grafik tersebut dapat disimpulkan tidak ada tanda-tanda *over fitting.*



Sumber: Penelitian (2021)

Gambar 4.9 Grafik *loss* akurasi model *VGG 16 transfer learning* menggunakan *optimizer RMSprop*

Dari Gambar 4.9 menunjukan bahwa hasil grafik akurasi yang diuji terhadap data *training* dan data *validation.* Garis yang berwarna biru merupakan data *validation* yang terjadinya *loss* dari yang tertinggi yaitu loss 1.50 di *epoch* 10 kemudian menurun sampai *epoch* 50. Sedangkan garis yang berwarna hijau merupakan data *validation* yang menunjukan *loss* mulai dari *epoch* satu yaitu 0.25 dan terus menurun sampai *epoch* 50. Dapat disimpulkan dari grafik *loss* akurasi model ini tidak ada tanda-tanda *over fitting.*

C. *Classification Report*

Berikut *classification report* yang dihasilkan dari model *VGG 16 tansfer learning* menggunakan *optimizer RMSprop*.

	precision	recall	f1-score	support
with_mask	0.97	1.00	0.99	35
withouth_mask	1.00	0.97	0.99	34
incorrect_mask	1.00	1.00	1.00	34
accuracy			0.99	103
macro avg	0.99	0.99	0.99	103
weighted avg	0.99	0.99	0.99	103

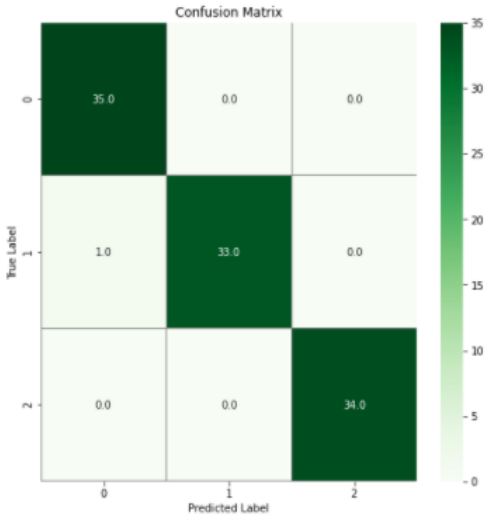
Sumber: Penelitian (2021)

Gambar 4.10 *Classification Report VGG 16 transfer learning* menggunakan *optimizer RMSprop*

Dari Gambar 4.10 menunjukan bahwa kelas *with mask* mendapatkan nilai *precision* 0.97, *recall* 1.00, dan F1 *score* 0.99. Kelas *without mask* mendapatkan nilai *precission* 1.00, *recall* 0.97, dan f1 *score* 0.99. kelas incorrect mask mendapatkan nilai *precision* 1.00, *recall* 1.00, dan f1 *score* 1.00. Akurasi yang didapat yaitu 99%.

D. *Confusion Metrix*

Berikut *confusion metrix* yang dihasilkan dari model *VGG 16 tansfer learning* menggunakan *optimizer RMSprop*.



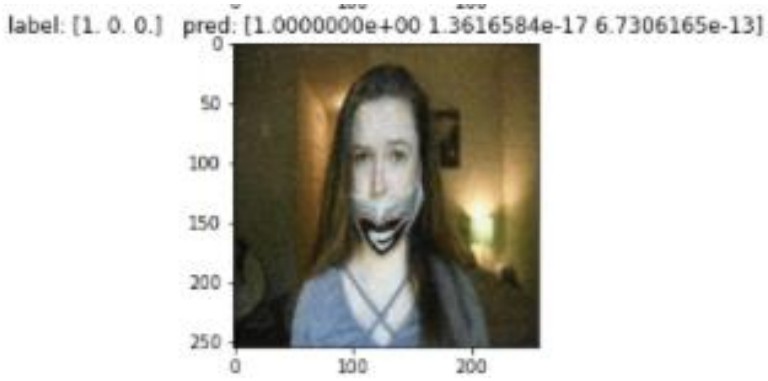
Sumber: Penelitian (2021)

Gambar 4.11 *Confusion matrix* model *VGG 16 transfer learning* menggunakan *optimizer RMSprop*

Dari Gambar 4.11 diatas menunjukan 0 sebagai kelas *with mask*, 1 sebagai kelas *without mask* dan 2 sebagai kelas *incorrect mask*. Dari *confusion matrix* diatas menunjukan model memprediksi benar untuk kelas "*with mask*" sebanyak 35. Model memprediksi benar untuk kelas "*without mask*" sebanyak 33. Dan model memprediksi benar untuk kelas "*incorrect mask"* sebanyak 34. Sedangkan model memprediksi salah pada kelas "*incorrect mask"* sebagai kelas "*with mask*" sebanyak 1.

E. *Test* Model atau Pengujian Model

Berdasasarkan model yang sudah dibuat, model tersebut dijuji terhadap dataset yang sama untuk mendapatkan keakuratan model. Model ini membaca dataset secara *random*. Kelas yang terpilh yang ditampilkan oleh model ini yaitu kelas *incorrect mask*. Berikut gambar hasil *test* model atau pengujian model.



Sumber: Penelitian (2021)

Gambar 4.12 Pengujian *VGG 16 transfer learning* menggunakan *optimizer RMSprop*

Gambar 4.12 Menunjukan citra hasil prediksi model atau *test* model kelas *incorrect mask* dengan label nilai 1.0.0 dan nilai prediksi 1.00.

4.3.2. Hasil Eksperimen *VGG 16 Transfer Learning* Menggunakan *Optimizer Adam*

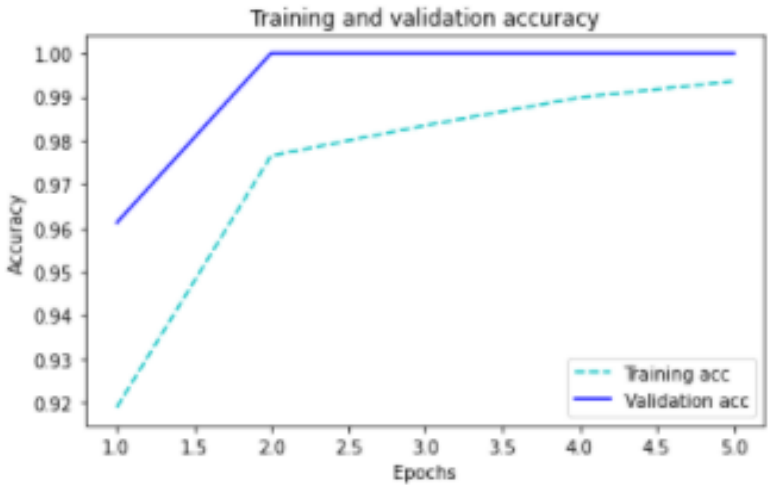
Hasil penelitian menggunakan *optimizer Adam* ini ditampikan dalam bentuk akurasi, grafik akurasi dan *loss, classifiacation report, confusion metrix*, dan *test* model atau pengujian:

A. Hasil Akurasi

Dengan model *VGG 16* menggunakan *optimizer Adam* dan diatur 50 *epoch* menghasilkan rata-rata akurasi sebesar 98% dan *loss* 0.04%.

B. Grafik Akurasi dan *Loss*

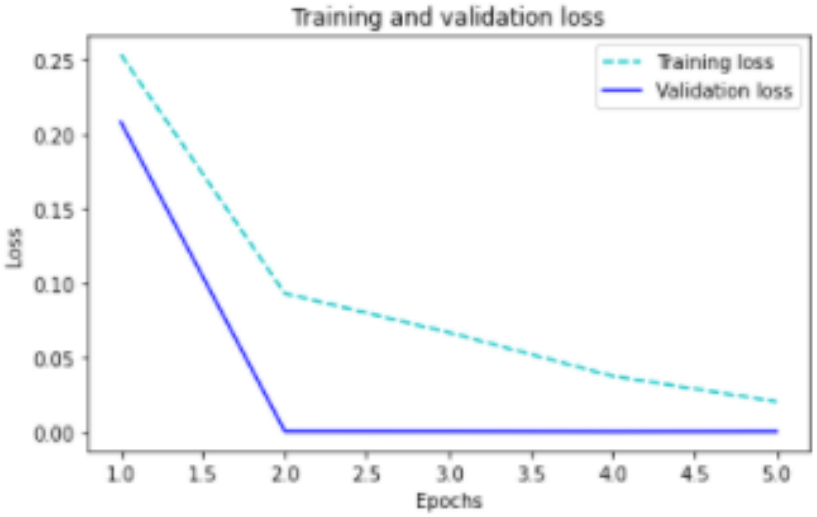
Berikut akurasi dan *loos* yang dihasilkan oleh model *VGG 16* dengan *transfer learning* menggunakan *optimizer Adam*.



Sumber: Penelitian (2021)

Gambar 4.13 Grafik akurasi data *train* dan data *validation* dengan model *VGG 16 – transfer learning* menggunakan *optimizer Adam*

Pada Gambar 4.13 menunjukan hasil grafik akurasi yang diuji terhadap data *training* dan data *validation*. Garis yang berwarna biru merupakan tanda akurasi data *validation*. Garis yang berwarna hijau merupakan tanda grafik data *train*. Kedua garis tersebut menunjukan kenaikan yang searah dan konsisten. Dari grafik tersebut dapat disimpulkan bahwa tidak ada tanda-tanda *over fitting*.



Sumber: Penelitian (2021)

Gambar 4.14 Grafik loss akurasi data *train* dan data *validation* dengan model *VGG 16 – transfer learning* menggunakan *optimizer Adam*

Pada gambar 4.14 menunjukan hasil grafik loss yang diuji terhadap data *training* dan data *validation*. Garis yang berwarna biru merupakan tanda grafik loss data *validation.* Garis yang berwarna hijau merupakan tanda grafik *loss* data *train*. Kedua garis tersebut menunjukan penurunan *loss* akurasi yang searah dan konsisten. Dari grafik tersebut dapat disimpulkan bahwa tidak ada tanda-tanda *under fitting.*

C. *Classification Report*

Berikut *confusion matrix* yang dihasilkan oleh model *VGG 16* dengan *transfer learning* menggunakan *optimizer Adam*.

	precision	recall	f1-score	support
with_mask	0.95	1.00	0.97	35
withouth_mask	1.00	0.97	0.99	34
incorrect_mask	1.00	0.97	0.99	34
accuracy			0.98	103
macro avg	0.98	0.98	0.98	103
weighted avg	0.98	0.98	0.98	103

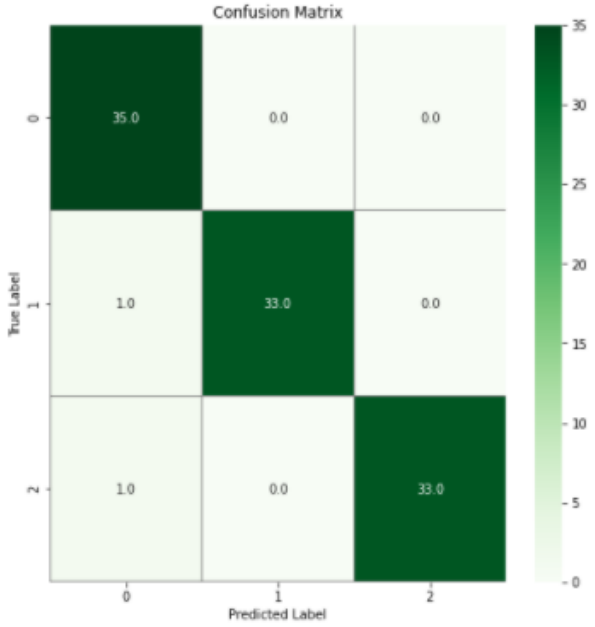
Sumber: Penelitian (2021)

Gambar 4.15 *Classification report* dengan model *VGG 16 – transfer learning* menggunakan *optimizer Adam*

Dari gambar 4.15 menunjukan bahwa precision kelas *with mask* mendapatkan nilai *precision* 0.95, *recall* 1.00, dan *f1 score* 97. Kelas *without mask* mendapatkan nilai precission 1.00, *recall* 0.97, dan *f1 score* 0.99. kelas *incorrect mask* mendapatkan nilai *precision* 1.00, *recall* 0.97, dan *f1 score* 0.99. Akurasi yang didapatkan oleh model ini yaitu 98%.

D. Confusion Matrix

Berikut *confusion matrix* arsitektur *VGG 16 transfer learning* menggunakan menggunakan *optimizer Adam.*



Sumber: Penelitian (2021)

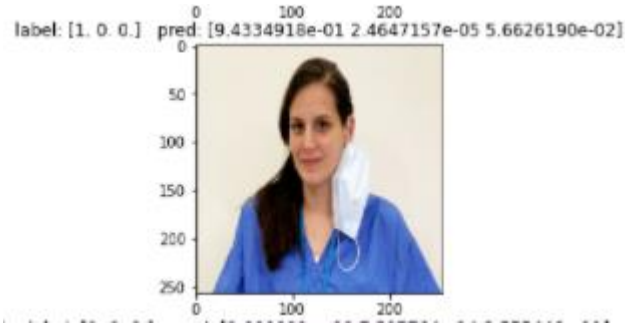
Gambar 4.16 *Confusion Matrix* dengan model *VGG 16 – transfer learning* menggunakan *optimizer Adam*

Dari gambar 4.16 diatas menunjukan 0 sebagai kelas *with mask*, 1 sebagai kelas *without mask* dan 2 sebagai kelas *incorrect mask*. Dari *confusion matrix* diatas menunjukan model memprediksi benar untuk kelas "*with mask*" sebanyak 35. Selanjutnya model memprediksi benar untuk kelas "*without mask*" sebanyak 33. Dan model memprediksi benar untuk kelas "*incorrect mask"* sebanyak 33. Sedangkan model memprediksi salah pada kelas "*incorrect mask"* sebagai kelas "*with mask*" sebanyak 1. Dan model

memprediksi salah kelas "*without mask*" sebagai kelas "*with mask"* sebanyak 1.

E. *Test* Model atau Pengujian Model

Berdasasarkan model yang sudah dibuat, model tersebut dijuji terhadap dataset yang sama untuk mendapatkan hasil keakuratan model. Model ini membaca dataset secara random. Kelas yang terpilh yang ditampilkan oleh model ini yaitu kelas *incorrect mask*. Berikut gambar hasil *test* modelataupengujian model.



Sumber: Hasil penelitian

Gambar 4.17 Hasil *test* model kelas *incorrect mask* model *VGG 16 – transfer learning* menggunakan *optimizer Adam*

Pada gambar 4.17 menunjukan bahwa model memilih *kelas incorrect mask* dengan label kelas 1.0.0 dan nilai prediksi 9.43.

4.3.4. Hasil Penelitian *VGG 16 transfer learning* menggunakan *Optimizer SGD*

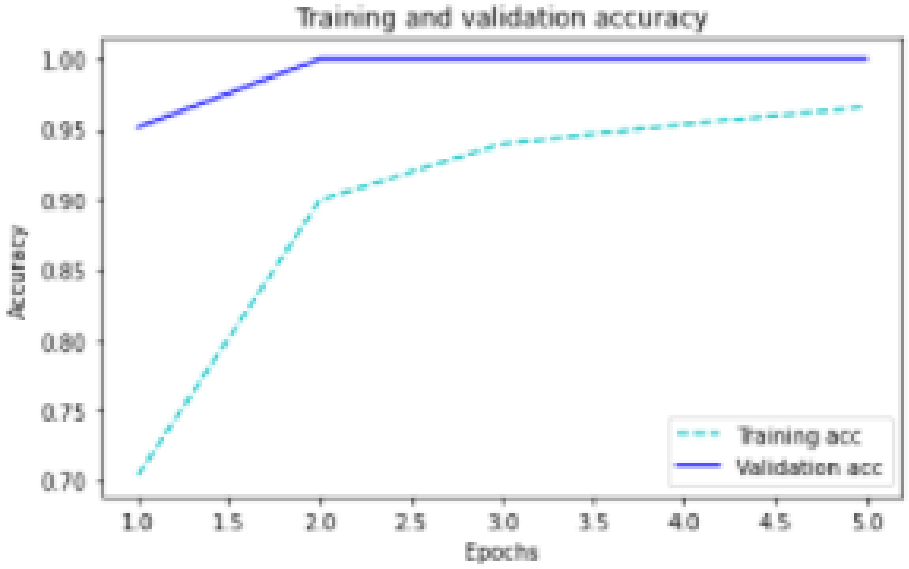
Hasil penelitianVGG 16 transfer learning menggunakan *optimizer SGD* ini ditampikan dalam bentuk akurasi, grafik akurasi dan *loss*, *classification repot*, *confusion metrix*, dan *test* model atau pengujian.

A. Akurasi

Dengan arsitektur *VGG 16* menggunakan *optimizer SGD* dan diatur 50 *epoch* menghasilkan rata-rata akurasi sebesar 97% dan *loss* 0.08.

B. Grafik Akurasi dan *Loss*

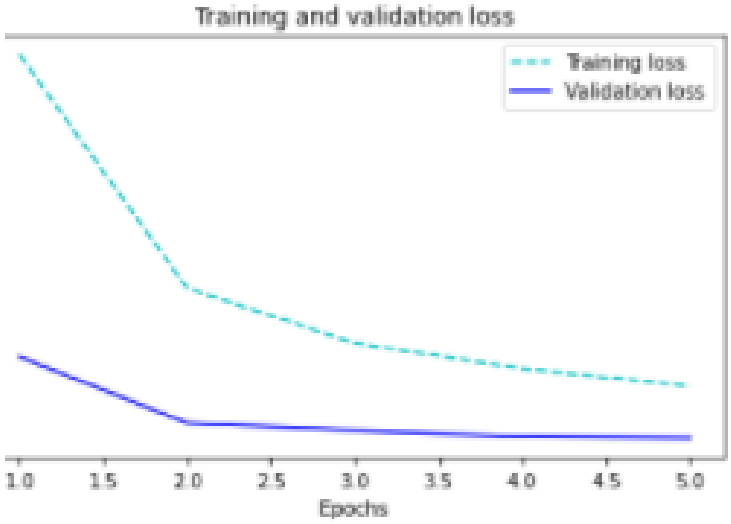
Berikut grafik akurasi dan *loss* yang dihasilkan oleh model *VGG 16* dengan *transfer learning* menggunakan *optimizer SGD.*



Sumber: Penelitian (2021)

Gambar 4.18 Grafik akurasi *VGG 16 transfer learning* menggunakan *optimizer SGD*

Pada gambar 4.18 menunjukan hasil grafik akurasi yang diuji terhadap data *training* dan data *validation*. Garis yang berwarna biru merupakan akurasi data *validation*. Garis yang berwarna hijau merupakan grafik data *train*. Kedua garis tersebut menunjukan kenaikan yang searah dan konsisten. Dari grafik tersebut dapat disimpulkan bahwa tidak ada tanda-tanda *over fitting*.



Sumber: Penelitian (2021)

Gambar 4.19 Grafik *loss* akurasi *VGG 16 transfer learning* menggunakan *optimizer SGD*

Pada gambar 4.19 menunjukan hasil grafik *loss* akurasi yang diuji terhadap data *training* dan data *validation*. Garis yang berwarna biru merupakan tanda akurasi data *validation*. Garis yang berwarna hijau merupakan tanda grafik data *train*. Kedua garis tersebut menunjukan kenaikan yang searah dan konsisten. Dari grafik tersebut dapat disimpulkan bahwa tidak ada tanda-tanda *over fitting*.

C. Classification Report

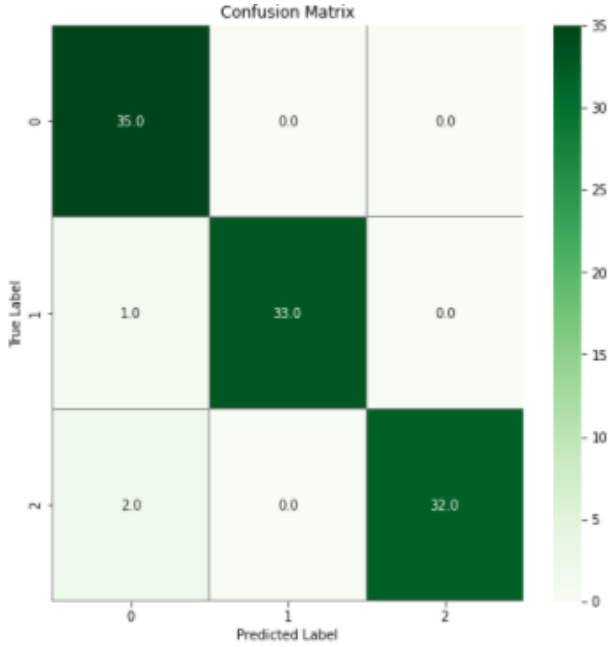
Berikut *classification report* yang dihasilkan oleh model VGG 16 *transfer learning* menggunakan *optimizer SGD.*

	precision	recall	f1-score	support
with_mask	0.92	1.00	0.96	35
withouth_mask	1.00	0.97	0.99	34
incorrect_mask	1.00	0.94	0.97	34
accuracy			0.97	103
macro avg	0.97	0.97	0.97	103
weighted avg	0.97	0.97	0.97	103

Sumber: Penelitian (2021)

Gambar 4.20 *Classification Report* model *VGG 16 transfer learning* menggunakan *optimizer SGD*

Dari gambar 4.20 menunjukan bahwa *precision* kelas *with mask* mendapatkan nilai *precision* 0.92, recall 1.00, dan F1 score 0.96. Kelas *without mask* mendapatkan nilai *precission* 1.00, *recall* 0.97, dan *f1 score* 0.99. Kelas *incorrect mask* mendapatkan nilai *precision* 1.00, *recall* 0.94, dan f1-*score* 0.97. Akurasi yang didapatkan dari model ini sebesar 97%.



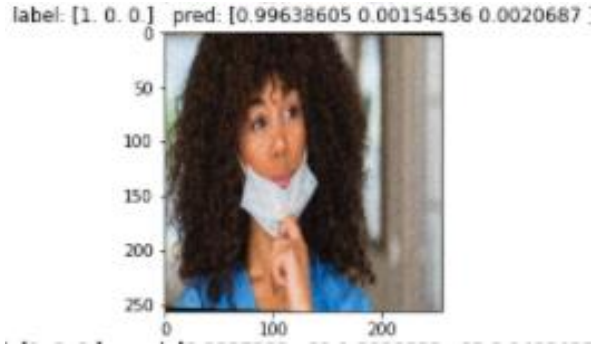
Sumber: Penelitian (2021)

Gambar 4.21 *Confusion matrix VGG 16 transfer learning* menggunakan *optimizer SGD*

Dari gambar 4.21 diatas menunjukan 0 sebagai kelas *with mask*, 1 sebagai kelas *without mask* dan 2 sebagai kelas *incorrect mask*. Dari *confusion matrix* diatas menunjukan model memprediksi benar untuk kelas "*with mask*" sebanyak 35. Model memprediksi benar untuk kelas "*without mask*" sebanyak 33. Dan model memprediksi benar untuk kelas "*incorrect mask"* sebanyak 32. Sedangkan model memprediksi salah pada kelas "*incorrect mask"* sebagai kelas "*with mask*" sebanyak 2. Model memprediksi salah pada kelas "*without mask"* pada kelas "*with mask"* sebanyak 1.

D. *Test* Model atau Pengujian Model

Berdasasarkan model yang sudah dibuat, model tersebut dijuji terhadap dataset yang sama untuk mendapatkan keakuratan model. Model ini membaca dataset secara *random*. Kelas yang terpilh yang ditampilkan oleh model ini yaitu kelas *incorrect mask*. Berikut gambar hasil *test* model atau pengujian model.



Sumber: Penelitian (2021)

Gambar 4.22 Pengujian *VGG 16 transfer learning* menggunakan *optimizer SGD*

Dari gambar 4.22 menunjukan bahwa, model memilih kelas *incorrect mask* dengan label kelas 1.0.0 dan prediksi 0.99.

Tabel 4.1 *Confusion Matrix Result*

No	Nama	Kelas	Precision	Recall	F1 Score	Akurasi
1	*VGG 16* dengan *transfer learning menggunakan optimizer* ***RMSprop***	With mask	0.97	1.00	0.99	99%*
		Without mask	1.00	0.97	0.99	
		Incorrect mask	1.00	1.00	1.00	
2	*VGG 16* dengan *transfer learning menggunakan optimizer* ***Adam***	With mask	0.95	1.00	0.97	98%
		Without mask	1.00	0.97	0.99	
		Incorrect mask	1.00	0.97	0.99	
3	*VGG 16* dengan *transfer learning menggunakan optimizer* ***SGD***	With mask	0.92	1.00	0.96	97%
		Without mask	1.00	0.97	0.99	
		Incorrect mask	1.00	0.4	0.97	

Sumber: Penelitian (2021)

Pada Tabel 4.1 diatas menunjukan bahwa, arsitektur *VGG 16* dengan *trasnfer learning* menggunakan *optimizer RMSprop* mendapatkan nilai *precission* 0,97, recall 1.00, *F1Score* 0.99 pada kelas *with mask.* Nilai *precission* 1.00, *recall* 97,

dan F1 *score* 0.99 pada kelas *without mask.* Nilai *precission* 1.00, *recal* 1.00, F1 *score* 1.00. Akurasi yang dihasilkan dari optimizer *RMSprop* sebesar 99%.

Untuk arsitektur *VGG 16* dengan *trasnfer learning* menggunakan *optimizer Adam* mendapatkan nilai *precission* 0.95, *recall* 1.00 F1 *score* 0.97 pada kelas *with mask*. Nilai *precission* 1.00, recall 0,97. F1 *score* 0.99. Akurasi yang dihasilkan *VGG 16* dengan *trasnfer learning* menggunakan *optimizer Adam yaitu 98%*.

Sedangkan arsitektur *VGG 16* dengan *trasnfer learning* menggunakan *optimizer SGD* dengan nilai *precission* 0.92, recall 1.00, *F1 Score* 0.96 pada kelas *with mask.* Nilai *precission* 1.00, *recall* 0.97, F1 score 0.99 pada kelas *without mask*. Nilai *precission 0.92, recall 1.00, F1 Score, 0.96.* Akurasi yang dihasilkan *VGG 16* dengan *trasnfer learning* menggunakan *optimizer SGD yait sebesar yaitu 97%*.

BAB V

PENUTUP

Pada bagian ini dijelaskan mengenai kesimpulan secara ringkas dari hasil penelitian yang telah dilakukan dan beberapa saran yang dapat diaplikasikan untuk penelitian selanjutnya.

5.1. Kesimpulan

Berdasarkan permasalahan yang telah dipaparkan dan diidentifikasi sebelumnya, maka penulis dapat mengambil kesimpulan dari penelitian yang telah dilakukan bahwa:

1. Dengan menggunakan model *VGG 16* dengan *transfer learning*, model dapat melakukan klasifikasi dengan baik pada dataset penggunaan masker dengan jumlah data sebanyak 2.079 citra.

2. Model *VGG 16* dengan transfer learning dapat melakukan klasifikasi dengan baik pada dataset citra tiga kelas *with mask, without mask dan incorrect mask*.

3. Dari ketiga *optimizer* yang digunakan yaitu *RMSprop, Adam,* dan *SGD* tidak semuanya dapat melakukan klasifikasi dengan baik. Hasil akurasi yang terbaik dari ketiga optimizer yaitu *optimizer RMSprop yaitu 99%*. Hal ini terbukti dari akurasi, *classification report* dan *confusion matrix* yang dihasilkan.

5.2. Saran

Berikut adalah beberapa saran yang dapat diaplikasikan untuk penelitian selanjutnya agar menghasilkan penelitian yang lebih baik dalam segmentasi, identifikasi atau proses lainnya:

1. Dengan model yang sudah dibuat di penelitian ini, peneliti selanjutnya dapat menerapkan identifikasi langsung melalui *video* secara *realtime* apakah orang tersebut menggunakan masker, tidak menggunakan masker atau menggunakan masker tidak benar.

2. Melakukan pengujian algoritma yang sudah diterapkan pada citra tunggal, dicoba terapkan pada citra berkelompok.

DAFTAR REFERENSI

[1] who.int, "Jumlah Perkembanagan Covid-19 di Dunia Menurut WHO," *https://covid19.who.int/ diakses pada 29 Juni 2021 Pukul 22.32*, 2021. .

[2] "Data Sebaran Covid-19 di Indonesia," *Https://covid19.go.id/ diakses pada 29 Juni 2021 Pukul 22.32*, 2021. .

[3] C. I. Burhanuddin and M. N. Abdi, "Ancaman Krisis Ekonomi Global Dari Dampak Penyebaran Virus Corona (Covid-19)," *AkMen*, vol. 17, pp. 90–98, 2020.

[4] A. Oumina, N. El Makhfi, and M. Hamdi, "Control the COVID-19 Pandemic: Face Mask Detection Using Transfer Learning," *2020 IEEE 2nd Int. Conf. Electron. Control. Optim. Comput. Sci. ICECOCS 2020*, 2020.

[5] D. M. Altmann, D. C. Douek, and R. J. Boyton, "What policy makers need to know about COVID-19 protective immunity," *Lancet*, vol. 395, no. 10236, pp. 1527–1529, 2020.

[6] S. E. Snyder and G. Husari, "Thor: A deep learning approach for face mask detection to prevent the COVID-19 pandemic," *Conf. Proc. - IEEE SOUTHEASTCON*, vol. 2021-March, 2021.

[7] M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, "A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic," *Meas. J. Int. Meas. Confed.*, vol. 167, no. July 2020, p. 108288, 2021.

[8] A. Rahim, K. Kusrini, and E. T. Luthfi, "Convolutional Neural Network untuk Kalasifikasi Penggunaan Masker," *Inspir. J. Teknol. Inf. dan Komun.*, vol. 10, no. 2, p. 109, 2020.

[9] Darmatasia, "Analisis Perbandingan Performa Model Deep Learning untuk Mendeteksi Penggunaan Masker," *J. IT*, vol. 11, no. 2, pp. 101–107, 2020.

[10] R. Kanotra, A. Analyst, N. Wadhwa, and N. Jeyanthi*, "Comparative Analysis of Object Detection Algorithms for Face Mask Detection," *Int. J. Eng. Adv. Technol.*, vol. 10, no. 4, pp. 148–151, 2021.

[11] T. Q. Vinh and N. T. N. Anh, "Real-Time Face Mask Detector Using YOLOv3 Algorithm and Haar Cascade Classifier," *Proc. - 2020 Int. Conf. Adv. Comput. Appl. ACOMP 2020*, pp. 146–149, 2020.

[12] M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, "Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection," *Sustain. Cities Soc.*, vol. 65, no. November 2020, p. 102600, 2021.

[13] Darmasita, "Deteksi Penggunaan Masker Menggunakan Xception Transfer Learning," *J. INSTEK (Informatika Sains dan Teknol.*, vol. 5, pp. 279–288, 2020.

[14] K. R. Prilianti, T. H. P. Brotosudarmo, S. Anam, and A. Suryanto, "Performance comparison of the convolutional neural network optimizer for photosynthetic pigments prediction on plant digital image," *AIP Conf. Proc.*, vol. 2084, 2019.

[15] "Kaggle.com," 2021. [Online]. Available: https://www.kaggle.com/.

[16] "Google Collaboratory," 2021. [Online]. Available: https://colab.research.google.com/.

[17] "Kaggle.com dataset," 2021. [Online]. Available: https://www.kaggle.com/spandanpatnaik09/face-mask-detectormask-not-mask-incorrect-mask. [Accessed: 27-Apr-2021].
[18] "Training Dataset," *2020*. [Online]. Available: https://www.techopedia.com/definition/33181/training-data, 2020.
[19] I. Herliawan *et al.*, "Classification of Liver Disease By Applying Random Forest," vol. 6, no. 1, pp. 89–94, 2020.
[20] J. Chen, D. Lopresti, and G. Nagy, "Conservative preprocessing of document images," *Int. J. Doc. Anal. Recognit.*, vol. 19, no. 4, pp. 321–333, 2016.
[21] E. S. Wijaya and Y. Prayudi, "Integrasi Metode Steganografi DCS Pada Image Dengan Kriptografi Blowfish Sebagai Model Anti Forensik Untuk Keamanan Ganda Konten Digital," *SNATI (Seminar Nas. Apl. Teknol. Informasi)*, pp. 11–17, 2015.
[22] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work.*, vol. 2020-June, pp. 3008–3017, 2020.
[23] L. Fei-Fei, J. Deng, and K. Li, "ImageNet: Constructing a large-scale image database," *J. Vis.*, vol. 9, no. 8, pp. 1037–1037, 2010.
[24] Sujitpal, "Transfer learning and fne tuning for cross domain image classifcation with Keras. GitHub: transfer learning and fne tuning for cross domain image classifcation with Keras," 2021. [Online]. Available: https://github.com/sujitpal/fttl-with-keras.
[25] D. Theckedath and R. R. Sedamkar, "Detecting Affect States Using VGG16, ResNet50 and SE-ResNet50 Networks," *SN Comput. Sci.*, vol. 1, no. 2, pp. 1–7, 2020.
[26] R. D. Nurfita and G. Ariyanto, "Implementasi Deep Learning Berbasis Tensorflow Untuk Pengenalan Sidik Jari," *Emit. J. Tek. Elektro*, vol. 18, no. 01, pp. 22–27, 2018.
[27] R. D. Nurfita and G. Ariyanto, "Implementasi Deep Learning Berbasis Tensorflow Untuk Pengenalan Sidik Jari," *Emit. J. Tek. Elektro*, vol. 18, no. 1, pp. 22–27, 2018.
[28] S. Albawi, T. A. M. Mohammed, and S. Alzawi, "Layers of a Convolutional Neural Network," *Ieee*, 2017.
[29] K. Yan, S. Huang, Y. Song, W. Liu, and N. Fan, "Face recognition based on convolution neural network," *Chinese Control Conf. CCC*, pp. 4077–4081, 2017.
[30] S. Ilahiyah and A. Nilogiri, "Implementasi Deep Learning Pada Identifikasi Jenis Tumbuhan Berdasarkan Citra Daun Menggunakan Convolutional Neural Network," *JUSTINDO (Jurnal Sist. dan Teknol. Inf. Indones.*, vol. 3, no. 2, pp. 49–56, 2018.
[31] S. Sharma, S. Sharma, and A. Athaiya, "Activation Functions in Neural Networks," *Int. J. Eng. Appl. Sci. Technol.*, vol. 04, no. 12, pp. 310–316, 2020.
[32] S. Khedkar, P. Gandhi, G. Shinde, and V. Subramanian, *Deep Learning Techniques for Biomedical and Health Informatics*. 2020.
[33] P. Toulis, T. Horel, and E. M. Airoldi, "The proximal Robbins–Monro

method," *J. R. Stat. Soc. Ser. B Stat. Methodol.*, vol. 83, no. 1, pp. 188–212,2021.
[34] D. R. Ente, S. A. Thamrin, S. Arifin, H. Kuswanto, and A. Andreza, "Klasifikasi Faktor-Faktor Penyebab Penyakit Diabetes Melitus Di Rumah Sakit Unhas Menggunakan Algoritma C4.5," *Indones. J. Stat. Its Appl.*, vol.4, no. 1, pp. 80–88, 2020.
[35] M. Yusa and W. Sindu, "Evaluasimodel Decision Tree C4.5 Guna Prediksi Posibilitas Resiko Obesitas," *Semin. Nas.*, pp. 147–152, 2015.
[36] N. Alfianika, *Buku Ajar Metode Penelitian Pengajaran Bahasa Indonesia*. Yogyakarta: Deepublish, 2018.
[37] M. M. Boulos, "Montclair State University Digital Commons Facial Recognition and Face Mask Detection Using Machine Learning Techniques," 2021.
[38] A. Gholamy, V. Kreinovich, and O. Kosheleva, "Why 70/30 or 80/20 Relation Between Training and Testing Sets : A Pedagogical Explanation," *Dep. Tech. Reports*, pp. 1–6, 2018.
[39] "Google Drive," 2021. [Online]. Available:https://www.google.com/drive/.

DAFTAR RIWAYAT HIDUP

A. Biodata Mahasiswa

NIM	: 14002397
Nama	: Irwan Herliawan
Tempat & Tanggal Lahir	: Ciamis, 7 Desember 1995
Jenis Kelamin	: Laki-laki
Alamat Lengkap	: Dusun Karangkamiri RT 07 RW 01 Desa Karangkamiri Kec. Langkaplancar Kab. Pangandaran (46391)

B. Riwayat Pendidikan Formal dan Non-Formal

1) MIS Ciparingga, 2007
2) SMP Negeri 2 Langkaplancar, 2012
3) SMK Negeri 2 Ciamis, 2015
4) AMIK BSI Tasikmalaya, 2018
5) Universitas BSI Bandung, 2019
6) Universitas Nusa Mandiri (Sedang Berjalan)

C. Riawayat Pengalaman Organisasi dan Pekerjaan

1. Ketua Osis SMP Negeri 2 Langkaplancar
2. Purna Paskibraka Indonesia Tahun 2012
3. Pengurus Purna Paskibraka Kabupaten Ciamis 2013-2019
4. Ketua HIMMI AMIK BSI Tasikmalaya, 2017-2018
5. Pengurus di Club Motor Vixion Rider Ciamis
6. Pengurus Karang Taruna, 2017-2019
7. Kepala Program Studi Administrasi Perkantoran di SMK Samudera Buana, 2017-2019
8. Marketing di Show Room Jaya Motor Ciamis
9. Marketing and Communication AMIK BSI Tasikmalaya 2016-2018

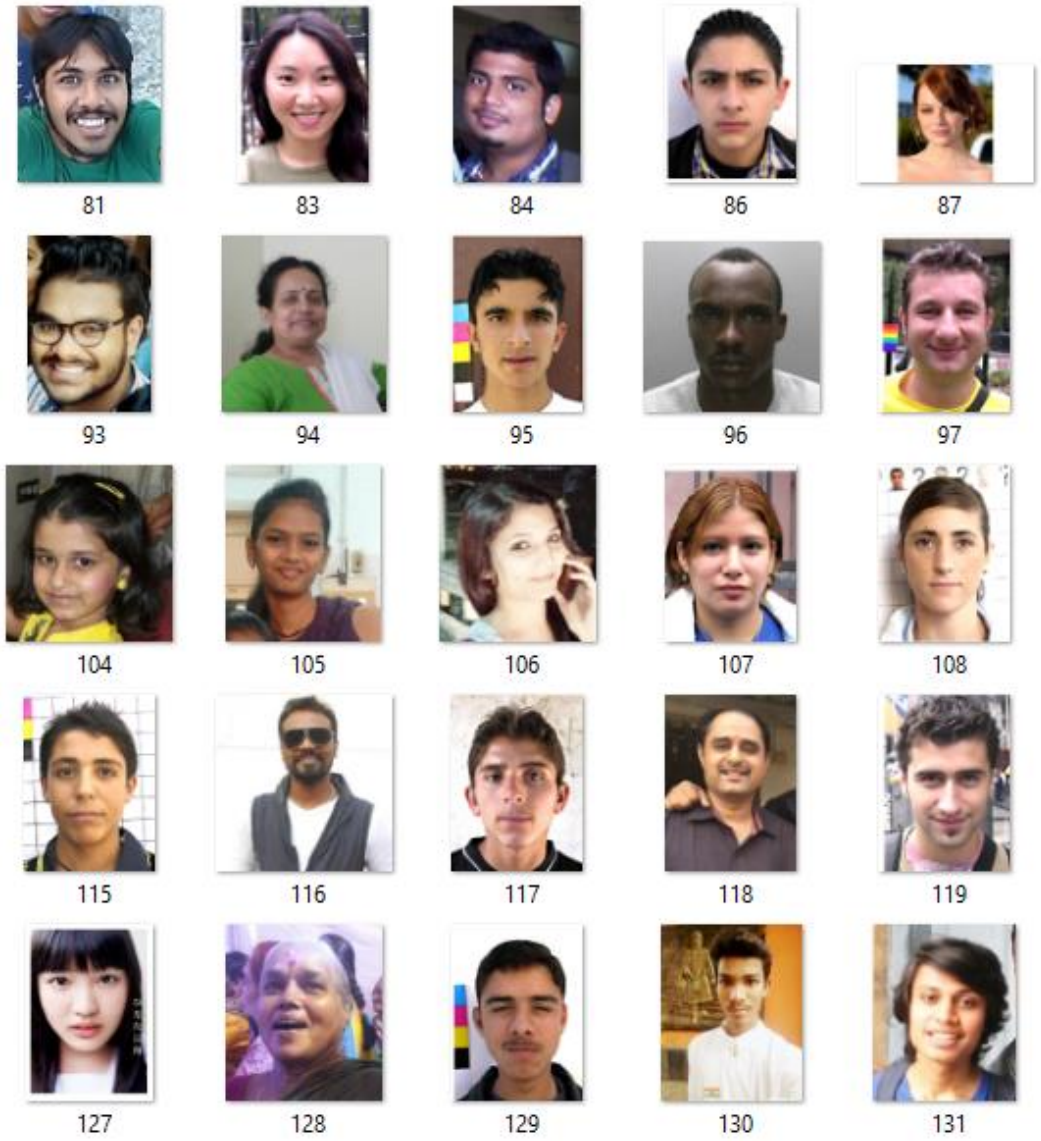
10. Marketing and Communicatiion Universitas BSI Cibitung 2019 sampai sekarang
11. *Owner* Grup CV Mitra Digital Persada Indonesia, 2021

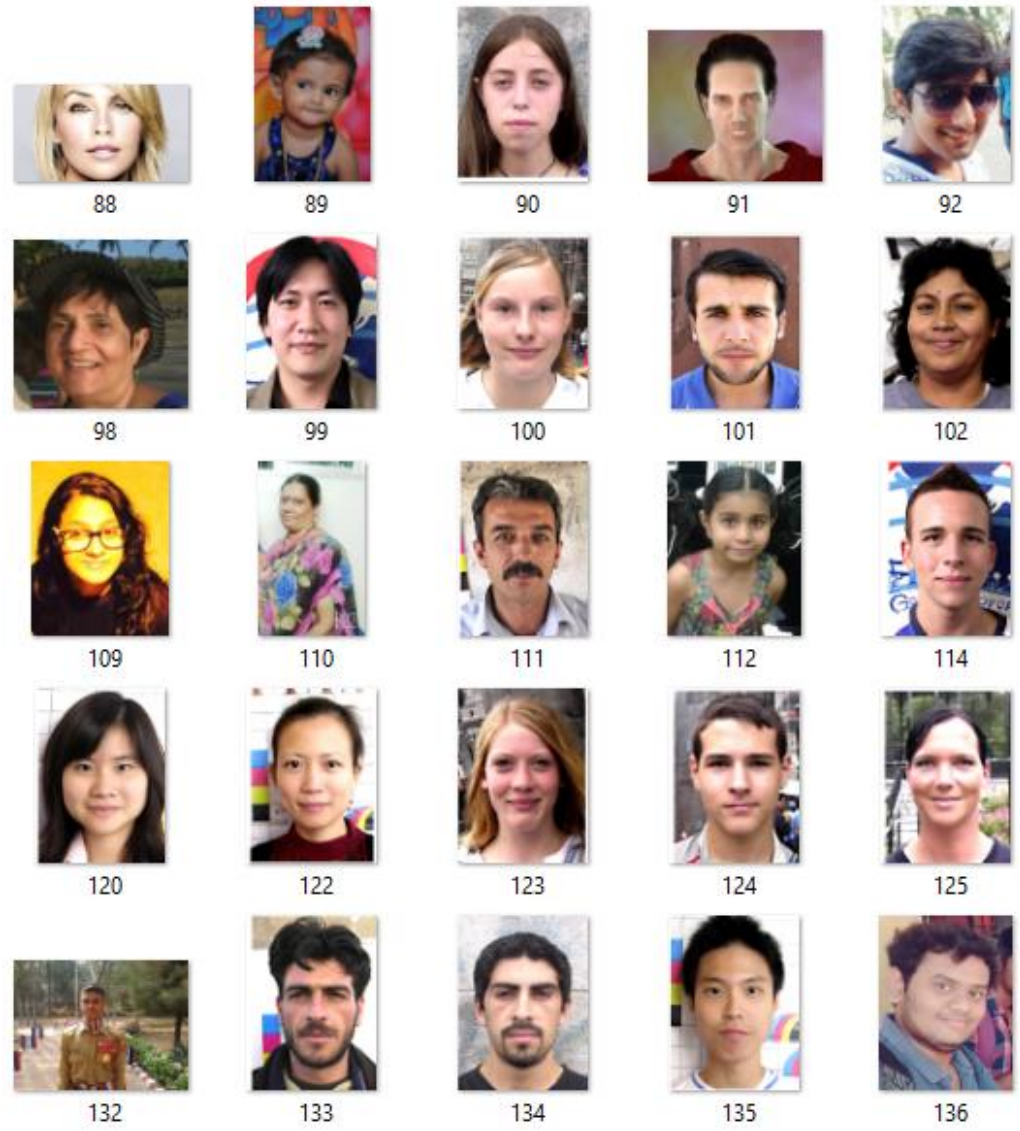
Cibitung, 31 Juli 2021

Irwan Herliawan, S.Kom

LAMPIRAN-LAMPIRAN

Lampiran 1. Citra Asli Dataset Kelas *Without Mask*




88
89
90
91
92
98
99
100
101
102
109
110
111
112
114
120
122
123
124
125
132
133
134
135
136

Lampiran 2. Citra Asli Dataset Kelas *With Mask*


84-with-mask
85-with-mask
86-with-mask
88-with-mask
89-with-mask
97-with-mask
98-with-mask
99-with-mask
100-with-mask
101-with-mask
108-with-mask
109-with-mask
110-with-mask
111-with-mask
112-with-mask
118-with-mask
119-with-mask
120-with-mask
121-with-mask
122-with-mask
128-with-mask
129-with-mask
131-with-mask
132-with-mask
133-with-mask

Lampiran 3 Citra Asli Dataset Kelas *Incorrect Mask*


aug_87


aug_88


aug_91


aug_93


aug_94


aug_100


aug_101


aug_102


aug_103


aug_104


aug_111


aug_113


aug_114


aug_115


aug_116


aug_122


aug_123


aug_124


aug_125


aug_126


aug_132


aug_133


aug_134


aug_135


aug_136

Lampiran 4 Hasil pelabelan pada setiap kelas


label: [0. 0. 1.]
label: [0. 0. 1.]
label: [0. 1. 0.]
label: [0. 1. 0.]
label: [1. 0. 0.]
label: [1. 0. 0.]
label: [1. 0. 0.]
label: [1. 0. 0.]
label: [0. 1. 0.]
label: [1. 0. 0.]
label: [1. 0. 0.]
label: [0. 1. 0.]
label: [1. 0. 0.]
label: [0. 0. 1.]
label: [1. 0. 0.]
label: [0. 1. 0.]
label: [0. 1. 0.]
label: [0. 1. 0.]
label: [0. 1. 0.]
label: [0. 1. 0.]
label: [0. 0. 1.]
label: [0. 0. 1.]
label: [0. 0. 1.]
label: [1. 0. 0.]
label: [0. 1. 0.]
label: [0. 1. 0.]
label: [0. 0. 1.]
label: [0. 1. 0.]
label: [0. 0. 1.]
label: [0. 1. 0.]
label: [0. 1. 0.]
label: [0. 1. 0.]
label: [0. 0. 1.]
label: [0. 1. 0.]
label: [1. 0. 0.]
label: [1. 0. 0.]
label: [0. 1. 0.]
label: [0. 1. 0.]
label: [1. 0. 0.]
label: [0. 0. 1.]
label: [1. 0. 0.]
label: [0. 0. 1.]
label: [1. 0. 0.]
label: [0. 1. 0.]
label: [0. 1. 0.]
label: [1. 0. 0.]
label: [0. 0. 1.]
label: [0. 1. 0.]

Lampiran 5. ***Source Code VGG 16 Transfer Learning dengan Optimzer RMSprop***

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

▾ IMPORT LIBRARIES

```
[ ] import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.metrics import confusion_matrix
    from sklearn.metrics import classification_report
    from sklearn.metrics import accuracy_score

    from tensorflow.keras.preprocessing.image import ImageDataGenerator
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import InputLayer, BatchNormalization, Dropout, Flatten, Dense, Activation, MaxPool2D, Conv2D
    from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
    from tensorflow.keras.applications.vgg16 import VGG16
    from tensorflow.keras.optimizers import Adam
    from tensorflow.keras.optimizers import SGD
    from tensorflow.keras.optimizers import RMSprop
    from tensorflow.keras.metrics import AUC
```

```
[ ] # define a function to plot the result from training step
    def show_result(history):

        # Print the result from the last epoch
        print('Last train accuracy: %s'%history.history['accuracy'][-1])
        print('Last validation accuracy: %s'%history.history['val_accuracy'][-1])

        loss = history.history['loss']
        val_loss = history.history['val_loss']

        acc = history.history['accuracy']
        val_acc = history.history['val_accuracy']
```

```
val_loss = history.history['val_loss']

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

epochs = range(1, len(loss) + 1)

# Define a subplot
fig, axs = plt.subplots(1,2,figsize=(15,4))

# Plot loss
loss_plot = axs[0]

loss_plot.plot(epochs, loss, 'c--', label='Training loss')
loss_plot.plot(epochs, val_loss, 'b', label='Validation loss')
loss_plot.set_title('Training and validation loss')
loss_plot.set_xlabel('Epochs')
loss_plot.set_ylabel('Loss')
loss_plot.legend()

# Plot accuracy
acc_plot = axs[1]

acc_plot.plot(epochs, acc, 'c--', label='Training acc')
acc_plot.plot(epochs, val_acc, 'b', label='Validation acc')
acc_plot.set_title('Training and validation accuracy')
acc_plot.set_xlabel('Epochs')
acc_plot.set_ylabel('Accuracy')
acc_plot.legend()
```

DATA AUGMENTATION

```
[ ] from tensorflow.keras.preprocessing import image_dataset_from_directory
    import numpy as np

    # Define paths
    ROOT_DIR = '/content/drive/MyDrive/FaceMaskIncorrect'
```

```
TRAIN_DIR = '/train'
TEST_DIR = '/test'
VAL_DIR = '/valid'

WITH_MASK = '/with_mask'
WITH_OUT_MASK = '/without_mask'
INCORRECT_MASK = '/incorrect_mask'

# Load images
train_data = image_dataset_from_directory(
    ROOT_DIR + TRAIN_DIR,
    label_mode= 'categorical',
)

val_data = image_dataset_from_directory(
    ROOT_DIR + VAL_DIR,
    label_mode= 'categorical',
)

test_data = image_dataset_from_directory(
    ROOT_DIR + TEST_DIR,
    label_mode= 'categorical',
    shuffle= False
)
```

```
Found 1873 files belonging to 3 classes.
Found 103 files belonging to 3 classes.
Found 103 files belonging to 3 classes.
```

```
[ ] # Define subplots
fig, axs = plt.subplots(4, 8, figsize=(15, 15))

# Convert batch to numpy iterator and retrieve the first batch
[image_batch, image_labels] = iter(train_data).next()
```

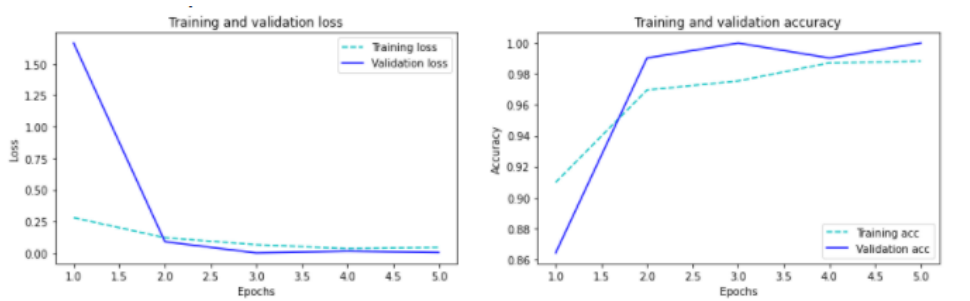
```
# Show images
for i, ax in enumerate(axs.ravel()):
    ax.imshow(image_batch[i]/255)
    ax.set_title("label: {}".format(image_labels[i]))
```


```
[ ] batch_size = 254
    base_model = VGG16(input_shape=(256,256,3),
                       include_top=False,
                       weights="imagenet")

    # Freezing Layers
    for layer in base_model.layers:
        layer.trainable=False

    # Building Model
    model=Sequential()
    model.add(base_model)
    model.add(Dropout(0.5))
    model.add(Flatten())
    model.add(BatchNormalization())
    model.add(Dense(2048,kernel_initializer='he_uniform'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
    model.add(Dense(1024,kernel_initializer='he_uniform'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
    model.add(Dense(3,activation='softmax'))

    model.compile(optimizer= 'RMSprop',
                  loss= 'categorical_crossentropy',
                  metrics= 'accuracy')

    # Model Summary
```

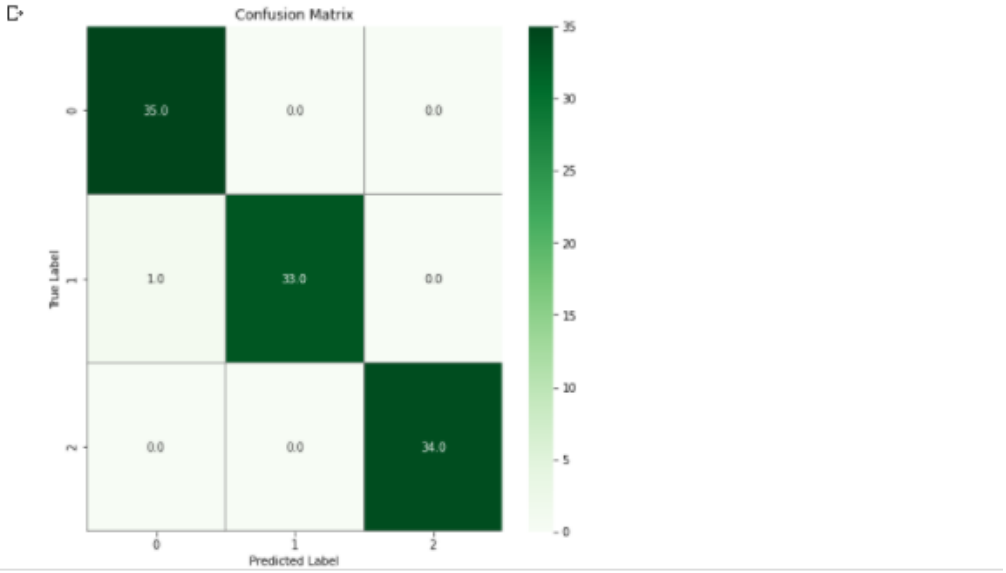
```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 1s 0us/step
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 8, 8, 512)         14714688
_________________________________________________________________
dropout (Dropout)            (None, 8, 8, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 32768)             0
_________________________________________________________________
batch_normalization (BatchNo (None, 32768)             131072
_________________________________________________________________
dense (Dense)                (None, 2048)              67110912
_________________________________________________________________
batch_normalization_1 (Batch (None, 2048)              8192
_________________________________________________________________
activation (Activation)      (None, 2048)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              2098176
_________________________________________________________________
batch_normalization_2 (Batch (None, 1024)              4096
_________________________________________________________________
activation_1 (Activation)    (None, 1024)              0
_________________________________________________________________
dropout_2 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 3075
=================================================================
Total params: 84,070,211
Trainable params: 69,283,843
Non-trainable params: 14,786,368
```

```
model_history=model.fit(train_data,
                        validation_data=val_data,
                        epochs = 5,
                        # callbacks = callback_list,
                        verbose = 1)
# Save model
model.save('face_mask_detection_model.h5')

# Evaluate model][=]
print(model.evaluate(test_data))

show_result(model_history)
```

```
Epoch 1/5
59/59 [==============================] - 1652s 28s/step - loss: 0.2798 - accuracy: 0.9098 - val_loss: 1.6643 - val_accuracy: 0.8641
Epoch 2/5
59/59 [==============================] - 1476s 25s/step - loss: 0.1208 - accuracy: 0.9696 - val_loss: 0.0890 - val_accuracy: 0.9903
Epoch 3/5
59/59 [==============================] - 1501s 25s/step - loss: 0.0654 - accuracy: 0.9754 - val_loss: 4.5386e-06 - val_accuracy: 1.0000
Epoch 4/5
59/59 [==============================] - 1470s 25s/step - loss: 0.0345 - accuracy: 0.9872 - val_loss: 0.0144 - val_accuracy: 0.9903
Epoch 5/5
59/59 [==============================] - 1504s 25s/step - loss: 0.0445 - accuracy: 0.9883 - val_loss: 0.0038 - val_accuracy: 1.0000
4/4 [==============================] - 91s 20s/step - loss: 0.0674 - accuracy: 0.9903
[0.06741753220558167, 0.9902912378311157]
Last train accuracy: 0.9882541298866272
Last validation accuracy: 1.0
```



```
# Get label of each image from batched dataset
y_con = np.concatenate([ y for x, y in test_data], axis=0)

# Predict each image
pred = model.predict(test_data)

# Convert from one-hot encoding format to integer
pred = np.argmax(pred, axis=-1)
y_pred = np.argmax(y_con, axis=-1)

# compute the confusion matrix
confusion_mtx = confusion_matrix(y_pred, pred)
# plot the confusion matrix
f,ax = plt.subplots(figsize=(8, 8))
sns.heatmap(confusion_mtx, annot=True, linewidths=0.01,cmap="Greens",linecolor="gray", fmt= '.1f',ax=ax)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```



```
[ ]  from sklearn.metrics import classification_report

     # Get label of each image from batched dataset
     y = np.concatenate([ y for x, y in test_data], axis=0)

     # Predict each image
     pred = model.predict(test_data)

     # Convert from one-hot encoding format to integer (for using in classification report)
     pred = np.argmax(pred, axis=-1)
     y = np.argmax(y, axis=-1)

     # Show model performance on test dataset
     print(classification_report(y, pred,  target_names=['with_mask', 'withouth_mask','incorrect_mask']))
```

```
                precision    recall  f1-score   support

     with_mask       0.97      1.00      0.99        35
  withouth_mask      1.00      0.97      0.99        34
incorrect_mask       1.00      1.00      1.00        34

      accuracy                           0.99       103
     macro avg       0.99      0.99      0.99       103
  weighted avg       0.99      0.99      0.99       103
```

```
[ ]  # Get first batch from test dataset
     [test_data_batch, test_label_batch] = iter(test_data).next()

     # Predict that batch
     test_pred = model.predict(test_data_batch)

     # Define subplots
     fig, axs = plt.subplots(8, 4, figsize=(30,30))

     # Show result
     for i, ax in enumerate(axs.ravel()):
         ax.imshow(test_data_batch[i]/255)
         ax.set_title('label: {}   pred: {}'.format(test_label_batch[i], test_pred[i]))
```

Lampiran 6. ***Source Code VGG 16 Transfer Learning dengan Optimzer Adam***

VGG16_TL_FaceMaskIncorrect_50_Epoch_Adam.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at August 2

+ Code + Text

```
[ ] from google.colab import drive
    drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

▾ IMPORT LIBRARIES

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import InputLayer, BatchNormalization, Dropout, Flatten, Dense, Activation, MaxPool2D, Conv2D
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import AUC
```

```
[ ] # define a function to plot the result from training step
    def show_result(history):

        # Print the result from the last epoch
        print('Last train accuracy: %s'%history.history['accuracy'][-1])
        print('Last validation accuracy: %s'%history.history['val_accuracy'][-1])

        loss = history.history['loss']
        val_loss = history.history['val_loss']

        acc = history.history['accuracy']
        val_acc = history.history['val_accuracy']

        epochs = range(1, len(loss) + 1)
```

```
val_acc = history.history['val_accuracy']

epochs = range(1, len(loss) + 1)

# Define a subplot
fig, axs = plt.subplots(1,2,figsize=(15,4))

# Plot loss
loss_plot = axs[0]

loss_plot.plot(epochs, loss, 'c--', label='Training loss')
loss_plot.plot(epochs, val_loss, 'b', label='Validation loss')
loss_plot.set_title('Training and validation loss')
loss_plot.set_xlabel('Epochs')
loss_plot.set_ylabel('Loss')
loss_plot.legend()

# Plot accuracy
acc_plot = axs[1]

acc_plot.plot(epochs, acc, 'c--', label='Training acc')
acc_plot.plot(epochs, val_acc, 'b', label='Validation acc')
acc_plot.set_title('Training and validation accuracy')
acc_plot.set_xlabel('Epochs')
acc_plot.set_ylabel('Accuracy')
acc_plot.legend()
```

Double-click (or enter) to edit

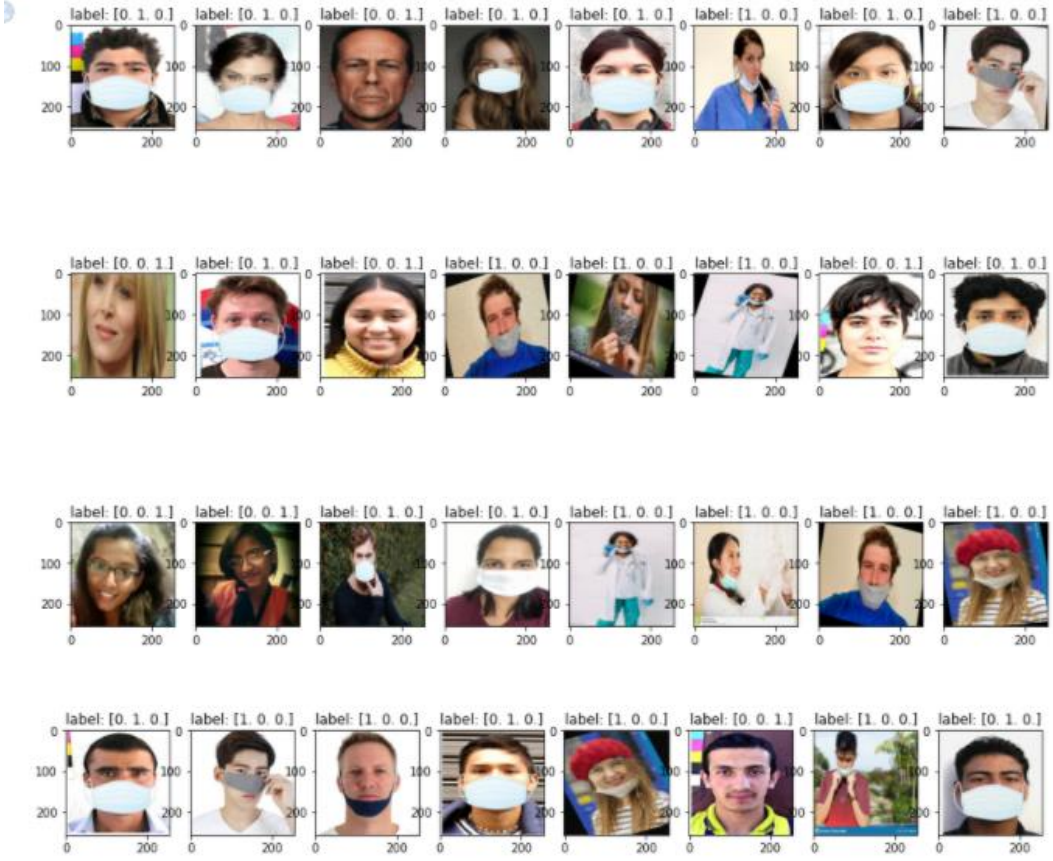
```
[ ] from tensorflow.keras.preprocessing import image_dataset_from_directory
    import numpy as np

    # Define paths
    ROOT_DIR = '/content/drive/MyDrive/FaceMaskIncorrect'
    TRAIN_DIR = '/train'
    TEST_DIR = '/test'
    VAL_DIR = '/valid'
```

```
WITH_MASK = '/with_mask'
WITH_OUT_MASK = '/without_mask'
INCORRECT_MASK = '/incorrect_mask'

# Load images
train_data = image_dataset_from_directory(
    ROOT_DIR + TRAIN_DIR,
    label_mode= 'categorical',
)

val_data = image_dataset_from_directory(
    ROOT_DIR + VAL_DIR,
    label_mode= 'categorical',
)

test_data = image_dataset_from_directory(
    ROOT_DIR + TEST_DIR,
    label_mode= 'categorical',
    shuffle= False
)
```

```
Found 1873 files belonging to 3 classes.
Found 103 files belonging to 3 classes.
Found 103 files belonging to 3 classes.
```

```
# Define subplots
fig, axs = plt.subplots(4, 8, figsize=(15, 15))

# Convert batch to numpy iterator and retrieve the first batch
[image_batch, image_labels] = iter(train_data).next()

# Show images
for i, ax in enumerate(axs.ravel()):
    ax.imshow(image_batch[i]/255)
    ax.set_title("label: {}".format(image_labels[i]))
```

```
# Show images
for i, ax in enumerate(axs.ravel()):
    ax.imshow(image_batch[i]/255)
    ax.set_title("label: {}".format(image_labels[i]))
```

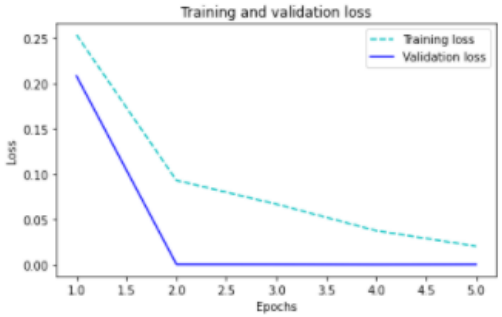


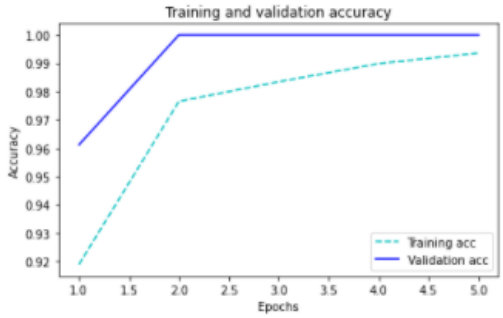
```
batch_size = 254
base_model = VGG16(input_shape=(256,256,3),
                   include_top=False,
                   weights="imagenet")

# Freezing Layers
for layer in base_model.layers:
    layer.trainable=False

# Building Model
model=Sequential()
model.add(base_model)
model.add(Dropout(0.5))
model.add(Flatten())
model.add(BatchNormalization())
model.add(Dense(2048,kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1024,kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(3,activation='softmax'))

model.compile(optimizer= 'adam',
              loss= 'categorical_crossentropy',
              metrics= 'accuracy')

# Model Summary
model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 0s 0us/step
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 8, 8, 512)         14714688
_________________________________________________________________
dropout (Dropout)            (None, 8, 8, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 32768)             0
_________________________________________________________________
batch_normalization (BatchNo (None, 32768)             131072
_________________________________________________________________
dense (Dense)                (None, 2048)              67110912
_________________________________________________________________
batch_normalization_1 (Batch (None, 2048)              8192
_________________________________________________________________
activation (Activation)      (None, 2048)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              2098176
_________________________________________________________________
batch_normalization_2 (Batch (None, 1024)              4096
_________________________________________________________________
activation_1 (Activation)    (None, 1024)              0
_________________________________________________________________
dropout_2 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 3075
=================================================================
Total params: 84,070,211
Trainable params: 69,283,843
Non-trainable params: 14,786,368
```

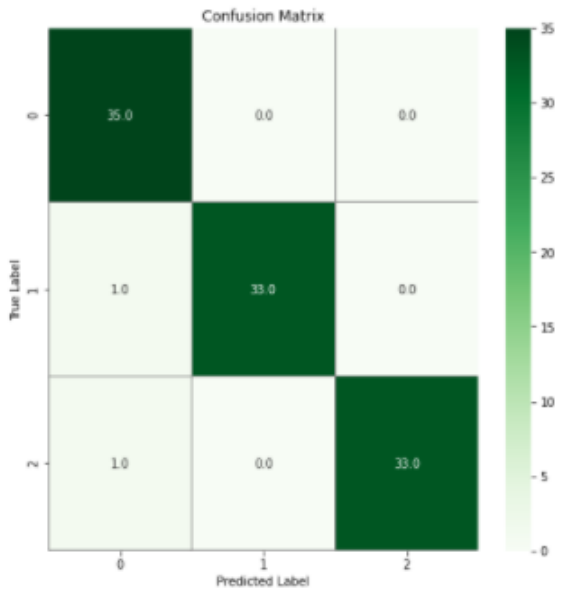
```
model_history=model.fit(train_data,
                        validation_data=val_data,
                        epochs = 5,
                        # callbacks = callback_list,
                        verbose = 1)
# Save model
model.save('face_mask_detection_model.h5')

# Evaluate model][=]
print(model.evaluate(test_data))

show_result(model_history)
```

```
Epoch 1/5
59/59 [==============================] - 1640s 28s/step - loss: 0.2539 - accuracy: 0.9188 - val_loss: 0.2083 - val_accuracy: 0.9612
Epoch 2/5
59/59 [==============================] - 1357s 23s/step - loss: 0.0931 - accuracy: 0.9765 - val_loss: 2.3876e-04 - val_accuracy: 1.0000
Epoch 3/5
59/59 [==============================] - 1353s 23s/step - loss: 0.0669 - accuracy: 0.9834 - val_loss: 1.9465e-04 - val_accuracy: 1.0000
Epoch 4/5
59/59 [==============================] - 1355s 23s/step - loss: 0.0375 - accuracy: 0.9899 - val_loss: 7.4160e-05 - val_accuracy: 1.0000
Epoch 5/5
59/59 [==============================] - 1350s 23s/step - loss: 0.0203 - accuracy: 0.9936 - val_loss: 1.6451e-04 - val_accuracy: 1.0000
4/4 [==============================] - 90s 20s/step - loss: 0.0458 - accuracy: 0.9806
[0.04575182497501373, 0.9805825352668762]
Last train accuracy: 0.993593156337738
Last validation accuracy: 1.0
```





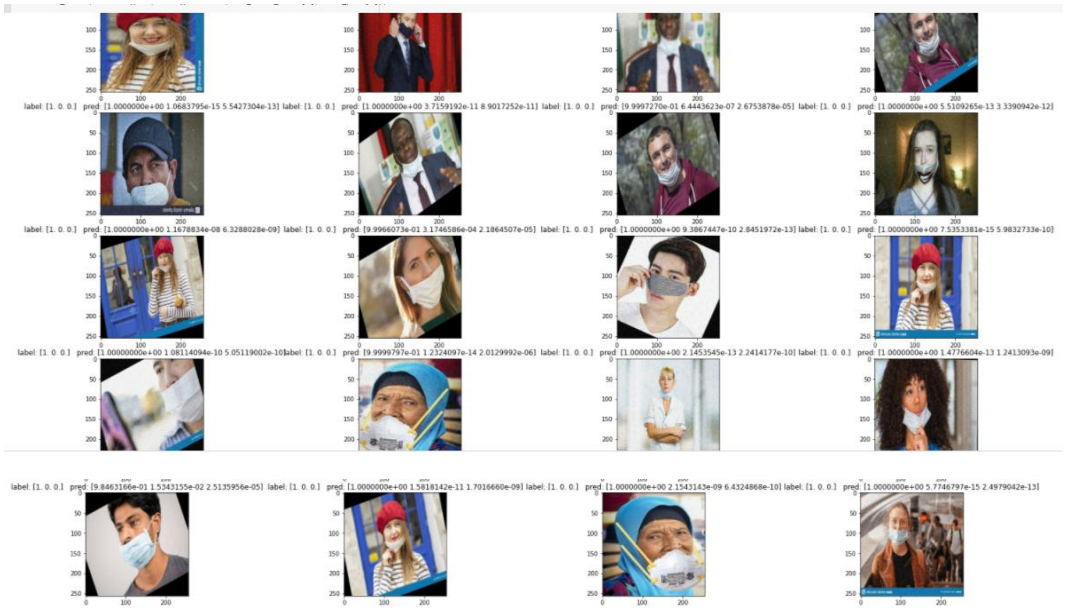
```
y_con = np.concatenate([ y for x, y in test_data], axis=0)

# Predict each image
pred = model.predict(test_data)

# Convert from one-hot encoding format to integer
pred = np.argmax(pred, axis=-1)
y_pred = np.argmax(y_con, axis=-1)

# compute the confusion matrix
confusion_mtx = confusion_matrix(y_pred, pred)
# plot the confusion matrix
f,ax = plt.subplots(figsize=(8, 8))
sns.heatmap(confusion_mtx, annot=True, linewidths=0.01,cmap="Greens",linecolor="gray", fmt= '.1f',ax=ax)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```



```
[ ] from sklearn.metrics import classification_report

    # Get label of each image from batched dataset
    y = np.concatenate([ y for x, y in test_data], axis=0)

    # Predict each image
    pred = model.predict(test_data)

    # Convert from one-hot encoding format to integer (for using in classification report)
    pred = np.argmax(pred, axis=-1)
    y = np.argmax(y, axis=-1)

    # Show model performance on test dataset
    print(classification_report(y, pred,  target_names=['with_mask', 'withouth_mask','incorrect_mask']))
```

```
                precision    recall  f1-score   support

     with_mask       0.95      1.00      0.97        35
 withouth_mask       1.00      0.97      0.99        34
incorrect_mask       1.00      0.97      0.99        34

      accuracy                           0.98       103
     macro avg       0.98      0.98      0.98       103
  weighted avg       0.98      0.98      0.98       103
```

```
[ ] # Get first batch from test dataset
    [test_data_batch, test_label_batch] = iter(test_data).next()

    # Predict that batch
    test_pred = model.predict(test_data_batch)

    # Define subplots
    fig, axs = plt.subplots(8, 4, figsize=(30,30))

    # Show result
    for i, ax in enumerate(axs.ravel()):
        ax.imshow(test_data_batch[i]/255)
        ax.set_title('label: {}   pred: {}'.format(test_label_batch[i], test_pred[i]))
```



Lampiran 7. ***Source Code VGG 16 Transfer Learning dengan Optimzer Adam***

VGG16_TL_FaceMaskIncorrect - 50_ Epoch - SGD.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at August 2

+ Code + Text

```
from google.colab import drive-
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

IMPORT LIBRARIES

```
[ ] import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.metrics import confusion_matrix
    from sklearn.metrics import classification_report
    from sklearn.metrics import accuracy_score

    from tensorflow.keras.preprocessing.image import ImageDataGenerator
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import InputLayer, BatchNormalization, Dropout, Flatten, Dense, Activation, MaxPool2D, Conv2D
    from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
    from tensorflow.keras.applications.vgg16 import VGG16
    from tensorflow.keras.optimizers import Adam
    from tensorflow.keras.optimizers import SGD
    from tensorflow.keras.metrics import AUC
```

```
[ ] # define a function to plot the result from training step
    def show_result(history):

        # Print the result from the last epoch
        print('Last train accuracy: %s'%history.history['accuracy'][-1])
        print('Last validation accuracy: %s'%history.history['val_accuracy'][-1])

        loss = history.history['loss']
        val_loss = history.history['val_loss']

        acc = history.history['accuracy']
        val_acc = history.history['val_accuracy']
```

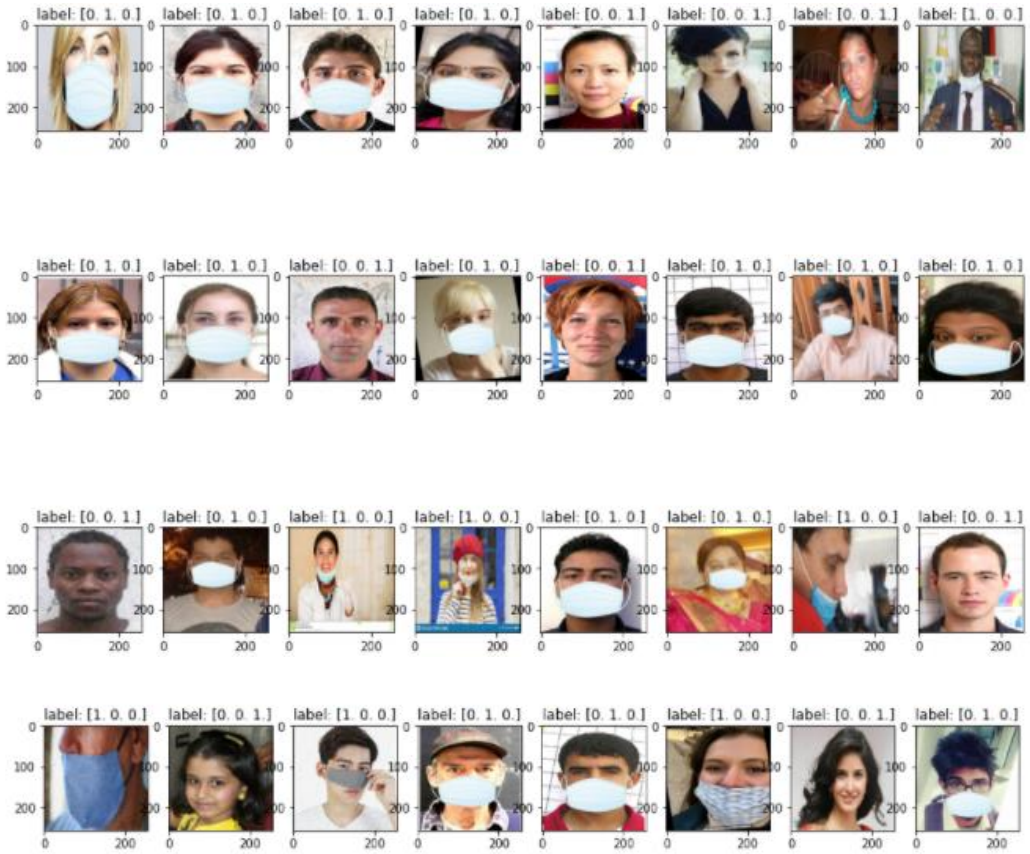
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

epochs = range(1, len(loss) + 1)

# Define a subplot
fig, axs = plt.subplots(1,2,figsize=(15,4))

# Plot loss
loss_plot = axs[0]

loss_plot.plot(epochs, loss, 'c--', label='Training loss')
loss_plot.plot(epochs, val_loss, 'b', label='Validation loss')
loss_plot.set_title('Training and validation loss')
loss_plot.set_xlabel('Epochs')
loss_plot.set_ylabel('Loss')
loss_plot.legend()

# Plot accuracy
acc_plot = axs[1]

acc_plot.plot(epochs, acc, 'c--', label='Training acc')
acc_plot.plot(epochs, val_acc, 'b', label='Validation acc')
acc_plot.set_title('Training and validation accuracy')
acc_plot.set_xlabel('Epochs')
acc_plot.set_ylabel('Accuracy')
acc_plot.legend()
```

```
[ ] from tensorflow.keras.preprocessing import image_dataset_from_directory
    import numpy as np

    # Define paths
    ROOT_DIR = '/content/drive/MyDrive/FaceMaskIncorrect'
    TRAIN_DIR = '/train'
    TEST_DIR = '/test'

    WITH_MASK = '/with_mask'
    WITH_OUT_MASK = '/without_mask'
```

```
[ ]  # Load images
     train_data = image_dataset_from_directory(
         ROOT_DIR + TRAIN_DIR,
         label_mode= 'categorical',
     )

     test_data = image_dataset_from_directory(
         ROOT_DIR + TEST_DIR,
         label_mode= 'categorical',
         shuffle= False
     )
```

```
Found 1873 files belonging to 3 classes.
Found 103 files belonging to 3 classes.
Found 103 files belonging to 3 classes.
```

```
[ ]  # Define subplots
     fig, axs = plt.subplots(4, 8, figsize=(15, 15))

     # Convert batch to numpy iterator and retrieve the first batch
     [image_batch, image_labels] = iter(train_data).next()

     # Show images
     for i, ax in enumerate(axs.ravel()):
         ax.imshow(image_batch[i]/255)
         ax.set_title("label: {}".format(image_labels[i]))
```

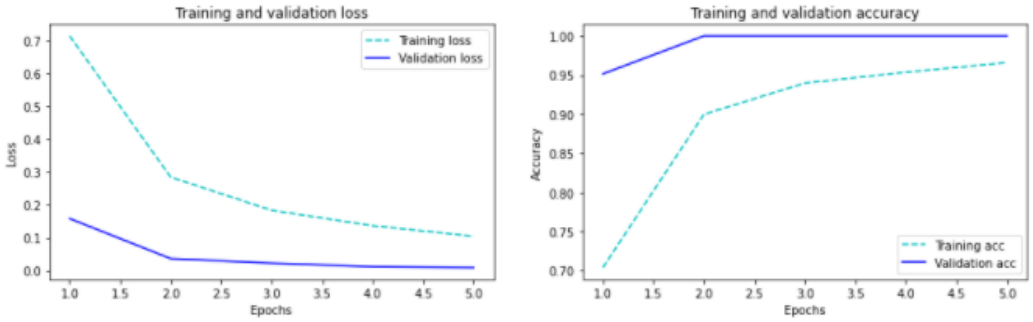


```
batch_size = 254
base_model = VGG16(input_shape=(256,256,3),
                   include_top=False,
                   weights="imagenet")

# Freezing Layers
for layer in base_model.layers:
    layer.trainable=False

# Building Model
model=Sequential()
model.add(base_model)
model.add(Dropout(0.5))
model.add(Flatten())
model.add(BatchNormalization())
model.add(Dense(2048,kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1024,kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(3,activation='softmax'))

model.compile(optimizer= 'SGD',
              loss= 'categorical_crossentropy',
              metrics= 'accuracy')

# Model Summary
model.summary()
```

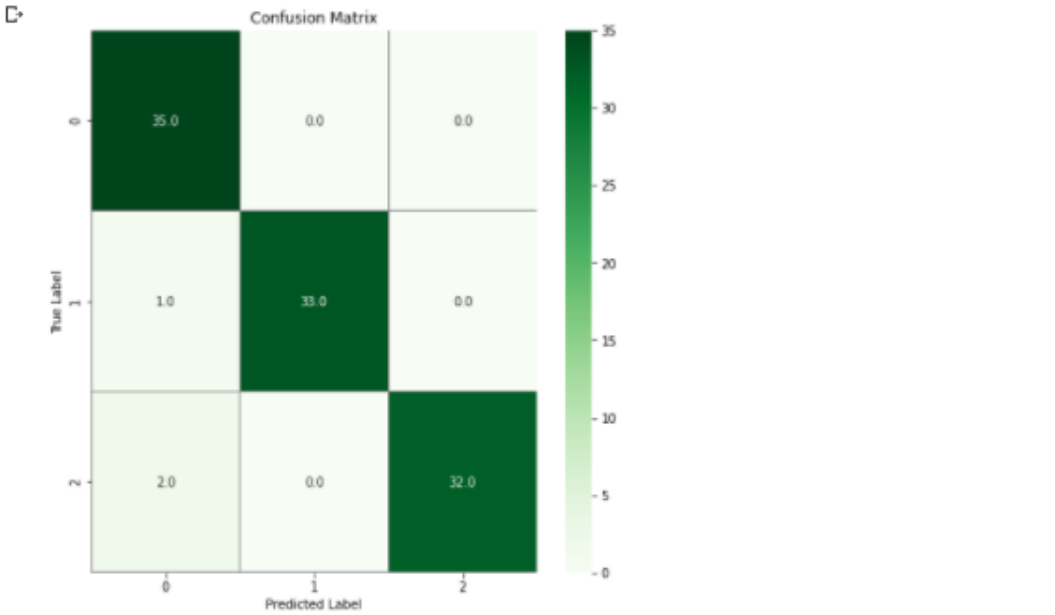
```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 0s 0us/step
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 8, 8, 512)         14714688
_________________________________________________________________
dropout (Dropout)            (None, 8, 8, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 32768)             0
_________________________________________________________________
batch_normalization (BatchNo (None, 32768)             131072
_________________________________________________________________
dense (Dense)                (None, 2048)              67110912
_________________________________________________________________
batch_normalization_1 (Batch (None, 2048)              8192
_________________________________________________________________
activation (Activation)      (None, 2048)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              2098176
_________________________________________________________________
batch_normalization_2 (Batch (None, 1024)              4096
_________________________________________________________________
activation_1 (Activation)    (None, 1024)              0
_________________________________________________________________
dropout_2 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 3075
=================================================================
Total params: 84,070,211
Trainable params: 69,283,843
Non-trainable params: 14,786,368
```

```
model_history=model.fit(train_data,
                        validation_data=val_data,
                        epochs = 5,
                        # callbacks = callback_list,
                        verbose = 1)
# Save model
model.save('face_mask_detection_model.h5')

# Evaluate model][=]
print(model.evaluate(test_data))

show_result(model_history)
```

```
Epoch 1/5
59/59 [==============================] - 1110s 19s/step - loss: 0.7139 - accuracy: 0.7037 - val_loss: 0.1586 - val_accuracy: 0.9515
Epoch 2/5
59/59 [==============================] - 769s 13s/step - loss: 0.2848 - accuracy: 0.8996 - val_loss: 0.0369 - val_accuracy: 1.0000
Epoch 3/5
59/59 [==============================] - 770s 13s/step - loss: 0.1837 - accuracy: 0.9397 - val_loss: 0.0230 - val_accuracy: 1.0000
Epoch 4/5
59/59 [==============================] - 770s 13s/step - loss: 0.1369 - accuracy: 0.9536 - val_loss: 0.0123 - val_accuracy: 1.0000
Epoch 5/5
59/59 [==============================] - 770s 13s/step - loss: 0.1049 - accuracy: 0.9658 - val_loss: 0.0096 - val_accuracy: 1.0000
4/4 [==============================] - 60s 13s/step - loss: 0.0873 - accuracy: 0.9709
[0.08730564266443253, 0.9708737730979919]
Last train accuracy: 0.9658302068710327
Last validation accuracy: 1.0
```



```
# Get label of each image from batched dataset
y_con = np.concatenate([ y for x, y in test_data], axis=0)

# Predict each image
pred = model.predict(test_data)

# Convert from one-hot encoding format to integer
pred = np.argmax(pred, axis=-1)
y_pred = np.argmax(y_con, axis=-1)

# compute the confusion matrix
confusion_mtx = confusion_matrix(y_pred, pred)
# plot the confusion matrix
f,ax = plt.subplots(figsize=(8, 8))
sns.heatmap(confusion_mtx, annot=True, linewidths=0.01,cmap="Greens",linecolor="gray", fmt= '.1f',ax=ax)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```



```
[ ]
```

```
from sklearn.metrics import classification_report

# Get label of each image from batched dataset
y = np.concatenate([ y for x, y in test_data], axis=0)

# Predict each image
pred = model.predict(test_data)

# Convert from one-hot encoding format to integer (for using in classification report)
pred = np.argmax(pred, axis=-1)
y = np.argmax(y, axis=-1)

# Show model performance on test dataset
print(classification_report(y, pred,  target_names=['with_mask', 'withouth_mask','incorrect_mask']))
```

```
                precision    recall  f1-score   support

     with_mask       0.92      1.00      0.96        35
 withouth_mask       1.00      0.97      0.99        34
incorrect_mask       1.00      0.94      0.97        34

      accuracy                           0.97       103
     macro avg       0.97      0.97      0.97       103
  weighted avg       0.97      0.97      0.97       103
```

```
[ ]
```

```
# Get first batch from test dataset
[test_data_batch, test_label_batch] = iter(test_data).next()

# Predict that batch
test_pred = model.predict(test_data_batch)

# Define subplots
fig, axs = plt.subplots(8, 4, figsize=(30,30))

# Show result
for i, ax in enumerate(axs.ravel()):
    ax.imshow(test_data_batch[i]/255)
    ax.set_title('label: {}   pred: {}'.format(test_label_batch[i], test_pred[i]))
```


label: [1. 0. 0.] pred: [9.9835336e-01 4.8137334e-04 1.1652575e-03]
label: [1. 0. 0.] pred: [9.9996459e-01 1.7507571e-05 1.7856231e-05]
label: [1. 0. 0.] pred: [9.9990106e-01 4.9887291e-05 4.8937782e-05]
label: [1. 0. 0.] pred: [0.99638605 0.00154536 0.0020687]
label: [1. 0. 0.] pred: [0.959305 0.03886582 0.00182915]
label: [1. 0. 0.] pred: [9.9723166e-01 6.2645192e-04 2.1418044e-03]
label: [1. 0. 0.] pred: [9.6206254e-01 3.7371349e-02 5.6613155e-04]
label: [1. 0. 0.] pred: [9.9997282e-01 1.8030822e-05 9.1408438e-06]