ABSTRAK

Semakin banyak bermunculan berbagai macam penyakit yang ada didunia salah satunya ialah *Coronavirus Disease* 2019 atau dengan kata lain yaitu *virus Covid-19*, penyakit menular yang disebabkan SARS-CoV-2. Penularan *virus* ini dapat dengan cepat menular dengan cara melalui cairan yang berasal dari hidung maupun mulut pada saat batuk,bersin maupun saat berbicara,karena hal tersebut diharapkan setiap orang harus melindungi diri maupun menjaga diri dengan berbagai macam protokol kesehatan, salah satunya adalah dengan menggunakan masker pada saat melakukan aktifitas diluar rumah. Pemerintah Indonesia membuat aturan penggunaan masker muka sehingga menjadi suatu kewajiban dan diperlukan sebuah sistem untuk mendeteksi penggunaan masker muka sehingga orang-orang lebih disiplin dan peduli terhadap penggunaan masker muka. Pada penelitian tugas akhir ini membuah suatu sistem yang memiliki tujuan untuk mendeteksi apakah orang yang ada didepan kamera memakai masker atau tidak secara *real time*. Perancangan sistem pendeteksi masker dilakukan dengan menggunakan metode *Convolutional Neural Network* (CNN) dan arsitektur *MobileNetV2* serta SSD *Resnet10* untuk pendeteksian muka. Pelatihan dan pengujian model penelitian dilakukan pada *Google Colaboratory* yang kemudian hasil dari pelatihan *model* dikonversi kedalam format h5 kemudian di*import* kedalam *Software Spyder* untuk dapat diimplementasikan kedalam sistem. Hasil dari pengujian menghasilkan akurasi bernilai 0,992, *recall* bernilai 0.994, *presisi* dengan nilai 0.990, *f1-score* dengan 0.992. Faktor yang memengaruhi perbedaan dari hasil pengujian antara lain spesifikasi kamera yang digunakan untuk pengujian, motif masker yang dipakai, tingkat pencahayaan dan jarak saat pengujian.

Kata Kunci: Covid 19, *Deep Learning*, CNN, *MobilenetV2*, *Google Colaboratory*, SSD *Resnet10*.

BAB 1

PENDAHULUAN

1.1 LATAR BELAKANG

Munculnya berbagai macam *virus* yang ada di dunia salah satunya adalah muncul *virus Coronavirus Disease* 2019 atau yang lebih dikenal dengan nama Covid-19 merupakan penyakit menular penyebabnya dari SARS-CoV2 yang merupakan salah satu dari jenis *virus corona*, ada dua jenis *virus corona* yang diketahui dapat menyebabkan dengan gejala berat seperti *Middle East Respiratory Syndrome* (MERS) dan *Severe Acute Respiratory Syndrome* (SARS) [1]. Persebaran *virus covid-19* dapat menyebar dengan sangat cepat pada saat seseorang sudah terpapar oleh *virus covid-19*, ditandai dengan tubuhnya akan mengalami beberapa gejala antara lain pernapasan terganggu, demam, batuk, dan bahkan sampai sesak napas. Kemunculan *virus* ini diduga berkaitan dengan Pasar Ikan di Wuhan, China pada akhir bulan Desember 2019 [2].

Berdasarkan dari beberapa studi membuktikan bahwa, *covid-19* ditularkan dari orang yang bergejala (*simptomatik*) ke orang lain yang berdekatan melalui *droplet* atau cairan yang dikeluarkan oleh seseorang pada saat orang bersin,batuk maupun pada saat hanya berbicara [2]. Penularan *droplet* terjadi ketika seseorang berada pada jarak yang dekat sekitar 1(satu) meter dengan seseorang yang memiliki gejala pernapasan seperti batuk atau bersin, sehingga *droplet* berisiko akan mengenai mulut dan hidung. Maka dari itu diperlukan pencegahan *covid-19* dengan beberapacara antara lain memakai masker, menjaga jarak, rajin mencuci tangan, dan selalu menerapkan protokol kesehatan.

Penerapan protokol kesehatan salah satunya yaitu dengan menggunakanmasker pada saat melakukan aktifitas diluar rumah baik kegiatan yang dilakukan perorangan maupun kegiatan yang dilakukan dengan bertemu orang banyak. Menggunakan masker merupakan salah satu cara perlindungan diri dari *virus covid-19* dan untuk saat ini semua orang baik orang sehat maupun sakit diwajibkan menggunakan masker. Orang sehat dapat menggunakan masker kain

saat berada diluar rumah sedangkan untuk orang yang memiliki gejala infeksi pernapasan seperti batuk atau bersin,yang dicurigai memiliki gejala *covid-19* dan juga petugas kesehatan memakai masker bedah [3]. Dengan banyaknya dampak yang disebabkan oleh *virus covid-19* saat berada diluar rumah ternyata masih banyak sekali orang-orang yang mengabaikan penggunaan masker ketika berada di keramaian. Akibatnya diperlukan sebuah terobosan untuk pendeteksian masker pada suatu tempat seperti ketika memasuki lingkungan kampus atau pun tempat-tempat umum lainnya agar penggunaan masker dapat lebih terpantau.

Pendeteksian masker wajah *covid-19* dilakukan menggunakan salah satu pembelajaran dari *Machine Learning* yaitu *Deep Learning. Machine Learning* sendiri merupakan sebuah cabang ilmu kecerdasan buatan yang mempunyai tujuan untuk memungkinkan sebuah mesin dapat melakukan pekerjaannya secara terampil dengan menggunakan perangkat lunak yang cerdas. *Machine Learning* dapat melakukan tugas-tugas tertentu dengan cara mempelajari algoritma dan model statistika yang ada. Proses dari pendeteksian masker wajah *covid-19* menggunakan dasar dari deep learning yaitu menggunakan representasi sederhana tetapi dapat menghasilkan sebuah sistem yang kompleks sehingga deep learning dapat dilatih untuk mendeteksi suatu objek dan juga melakukan klasifikasi objek seperti yang dilakukan oleh penulis yaitu pendeteksian masker wajah [4].

Penelitian dilakukan untuk mendapatkan hasil yang penulis inginkan dan dibutuhkan digunakanlah teknik pelatihan untuk klasifikasi objek dengan menggunakan *Convolutional Neural Network*. Setelah dilakukan pelatihan *model* dengan CNN maka selanjutnya *model* dikombinasikan dengan modul SSD *Resnet* untuk mendeteksi suatu objek atau masker wajah. Membentuk suatu sistem pendeteksi objek dapat menggunakan dua kombinasi algoritma diantaranya dengan melakukan klasifikasi kemudian mendeteksi objek tersebut. [5].

1.2 RUMUSAN MASALAH

Rumusan masalah dari penelitian ini adalah:

1) Bagaimana mengimplementasikan model arsitektur *mobilenetv2* dan *ssd resnet10* untuk pendeteksian masker muka?
2) Bagaimana kinerja model arsitektur untuk pendeteksian masker muka?

1.3 BATASAN MASALAH

Batasan masalah dari penelitian ini adalah:

Metode *deep learning* yang digunakan untuk pendeteksi masker muka

1) Terdiri dari algoritma pendeteksi muka dengan SSD *ResNet,* serta algoritma klasifikasi wajah dengan CNN (*MobileNetV2*).
2) Sistem dirancang untuk mengidentifikasi pengguna tersebut memakai masker atau tidak memakai masker.
3) Sistem pendeteksi masker muka menggunakan bahasa pemrograman *Python* dengan antarmuka dan infrastruktur *Google Colaboratory*.
4) Pembuatan *model deep learning* pada deteksi masker muka dengan citra menggunakan *Framework Keras* dan *Tensorflow*.

1.4 TUJUAN

Tujuan dari penelitian ini adalah:

1. Melakukan implementasi prosedur metode CNN dengan menggunakan model arsitektur *mobilenetv2* dan *ssd resnet10* untuk pendeteksian masker muka.
2. Menguji kinerja model arsitektur *mobilenetv2* dan *ssd resnet10* untuk pendeteksian masker muka.

1.5 MANFAAT

Penelitian ini diharapkan dapat membantu mencegah penyebaran *Covid-19* ditempat umum seperti kampus dan instansi lainnya. Diharapkan juga penelitian ini dapat mempermudah petugas dalam pemeriksaan dalam penggunaan masker setiap orang yang berada diluar rumah apalagi ditempat yang ramai.

1.6 SISTEMATIKA PENULISAN

Sistematika penulisan penelitian ini terbagi menjadi beberapa bab berdasarkan pengelompokkan pokok-pokok pikiran yang tercantum dengan bab- bab sebagai berikut :

BAB I PENDAHULUAN

Bab ini berisikan tentang latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, dan sistematika penulisan.

BAB II DASAR TEORI

Bab ini berisikan tentang kajian pustaka yang dijadikan sebagairujukkan dalam tugas akhir ini dan berisikan tentang landasan- landasan teori pendukung yang digunakan pada tugas akhir ini.

BAB III METODOLOGI PENELITAN

Bab ini berisikan tentang metode penelitian yang menjelaskan bagaimana perancangan sistem, pengujian sistem, alat yang akandigunakan, dan alur penelitian

BAB IV HASIL DAN PEMBAHASAN

Bab ini berisikan tentang pembahasan dan analisa berdasarkan hasil penelitian yang telah didapatkan melalui sistem yang telah dibuat.

BAB V PENUTUP

Bab ini berisikan tentang kesimpulan berdasarkan analisis yang telah dijelaskan pada bab sebelumnya dan saran yang ditunjukkan untuk penelitian selanjutnya.

BAB 2

DASAR TEORI

2.1 KAJIAN PUSTAKA

Penelitian Ivan Hartono, Agustinus Noertjahyana, Leo Willyanto Santoso pada tahun 2022 yang berjudul " Deteksi Masker Wajah dengan Metode *Convolutional Neural Network*" meneliti tentang pembuatan sistem untuk menerapkan protokol kesehatan yang sudah dibuat oleh pemerintah yang bertujuan dalam membantu kinerjadari petugas keamanan fasilitas umum untuk memeriksa apakah masyarakat sudah menggunakan masker dengan benar atau tidak dengan menggunakan arsitektur VGG16Net dan untuk mendeteksi wajah menggunakan SSD*Resnet10*. *Penelitian ini menggunakan dataset* dengan total data sebanyak 2.095 *data* gambar yang diperoleh dari *github* /chandrikadeb7 "*Face Mask Detection*" dengan perbandingan data *training*sebesar 75% dari jumlah keseluruhan *dataset* yang dimiliki dan data *testing* dengan ukuran sebesar 25% dari semua *dataset* yang ada. Hasil performa dari VGG16Net dan SSD*Resnet* 98% untuk akurasi dan *f1-score* sebesar 98%. Saran dalam penelitian ini agar dapat menambah jumlah *dataset* yang lebih bervariasi dan menguji konfigurasi pada VGG16Net untuk meningkatkan akurasi[6].

Parmonang R. Togatorop, Ahmad Fauzi pada tahun 2019 dengan penelitiannya yang berjudul "Klasifikasi Penggunaan Masker Wajah Menggunakan *Squeeznet*" membahas tentang banyaknya penelitian terkait pendeteksian penggunaan masker yang sudah banyak dilakukan. Penelitian ini menggunakan *deep transfer learning model* menggunakan algoritma YoloV3 untuk mendeteksi wajah dan Darknet-53 untuk *backbone.* Penulis menggunakan *single shot multibox detector* dan *MobileNetV2* serta melakukan pendeteksian secara *realtime* menggunakan *Transfer Learning* penggunaan *Naïve Bayes* (NB) dan *Support Vector Machine* (SVM) pada saat proses klasifikasi untuk melihat performa *Squeeznet*. Hasil yang didapat dengan menggunakan *Naïve Bayes* (NB) akurasi 0,958, presisi 0,981, *recall* 0,938, sedangkan dengan menggunakan *Support Vector Machine* (SVM) akurasi sebesar 0,992, presisi 0,994 dan *recall* 0,990 [7].

Menurut penelitian yang dilakukan oleh Tri Septiana Nadia Puspita Putri, Mohamad Al Fikih, Novendra Setyawan melakukan penelitian pada tahun 2020 tentang "*Face Mask Detection Covid19 Using Convolutional Neural Network*" yang membahas tentang penerapan sistem deteksi masker dengan menggunakan pengolahan citra. Perancangan sistem dalam penelitian ini menggunakan kombinasi klasifikasi dari pendeteksian objek, gambar dan pelacakan objek untuk mengembangkan sistem deteksimasker dalam bentuk gambar maupun *video*. *Dataset* diambil dengan berbagai macam variasi antara lain gambar yang memakai hijab, topi maupun yang tidak memakai atribut apapun, dan juga gambar yang berasal dari berbagai negara antara lain asia, eropa, dan amerika. Penelitian ini menghasilkan nilai akurasi sebesar 0,9933% dan *training loss* 0,0213%[8].

Penelitian yang dilakukan oleh Muhammad Aminullah pada tahun 2021 dengan judul "Alat Deteksi Masker Dengan Metode *Convolutional Neural Network* Untuk Tunanetra Pada *Era New Normal*" melakukan penelitian pembuatan alat untuk mendukung penyandang tunanetra dalam menghadapi *covid-19*. Dalam perancangan ini penelitian dilakukan dengan menggunakan metode CNN dan penggunaan *tensorflow* yang berfungsi untuk *framework deep learning* bertujuan untuk mengenali dan mengklasifikasikan suatu objek. Sistem deteksi masker dan jarak di implementasikan pada Rasberry Pi dan mendapat FPS sebesar 0,33. Hasil pengujian dapat diterima dengan bantuan *earphone* dan pengasuh dapat melakukan pemantauan melalui sistem pengawasan yang ada pada alat ini secara *real-time* antara lain dapat diipantau suhu badan, keadaan dan lokasi keberadaan penyandang tunanetra dengan mengguunakan aplikasi *smartphone* pada penelitian ini juga dilengkapi dengan API *Google Maps* untuk menampilkan lokasi penyandang tunanetra[16].

Nyoman Purnama, Putu Kusuma Negara melakukan penelitian tentang "Deteksi Masker Pencegahan *Covid19* Menggunakan *Convolutional Neural Network* Berbasis Android" pada tahun 2021, membuat sebuah perbandingan dua metode optimasi pada *deep learning* yaitu *adam* dan *gradient descent* dan tujuan dari dilakukan proses pengujian untuk mengetahui nilai *recall*, *presisi* dan akurasi setiap *optimizer*. Pengujian diproses menggunakan perangkat dengan berbasis *android* dan penggunaan bahasa pemograman *java* serta *framework mobilenetv2.* Penelitian yang

dilakukan ini menghasilkan akurasi sebesar 90% pada saat menggunakan *adam* dan saat menggunakan optimasi *gradient descent* menghasilkan angka sebesar 80% [17].

Penelitian dengan judul "*Convolutional Neural Network* Arsitektur *MobileNet-V2* Untuk Mendeteksi Tumor Otak" yang dilakukan oleh Widi Hastomo, Sugiyanto dan Sudjiran pada tahun 2021 melakukan sebuah penelitian untuk memprediksi penyakit tumor otak yang diderita oleh pasien yang diprediksi dari kemampuan pembacaan *image* dari peralatan CT *Scanner* dengan menggunakan metode CNN. Penggunaan arsitektur *MobilenetV2* sebagai arsitektur yang digunakan pada penelitian ini memiliki fungsi untuk digunakan sebagai *training* data dan *testing* data dengan total 2.870 *image*tumor otak. Penelitian yang dilakukan menghasilkan sebuah nilai *training* akurasi sebesar 97% dan nilai 94% untuk nilai *testing* serta nilai disetiap akurasi pada setiap klasifikasi yang dilakukan antara lain *glioma* sebesar 99%, *meningioma* sebesar 85%, *no tumor* sebesar 99% dan *pituaty* sebesar 96%. Hasil akurasi yang dilakukan dalam penelitian kali ini mendapatkan hasil yang sangat baik dan menghasilkan sebuah modelyang memiliki manfaat untuk mendiagnosa pasien dengan cepat, murah dan akurat[18].

Penelitian tentang pendeteksian masker wajah *covid-19* menggunakan metode CNN yang akan dilakukan penulis tidak terlepas dari hasil penelitian yang telah disebutkan yang selanjutnya dipaparkan pada tabel 2.1.

Tabel 2.1 Tinjauan Pustaka Penelitian Terdahulu

No	Jurnal	Keterangan
1	Mohammad Ivan Hartono, Agustinus Noertjahyana, Leo Willyanto Santoso yang berjudul "*Deteksi Masker Wajah dengan Metode Convolutional Neural Network*" tahun 2022.	Pembuatan sistem deteksi masker wajah menggunakan metode *Convolutinal neural network* menggunakan arsitektur VGG16Net dan pendeteksi wajah menggunakan ssd *resnet10*.

No	Jurnal	Keterangan
2	Parmonangan R. Togatorop, Ahmad Fauzi dengan penelitiannya yang ber judul ”Klasifikasi Penggu naan Masker Wajah Menggunakan *Squeezenet*” pada tahun 2019.	Klasifikasi Penggunaaan masker wajah menggunakan *Squeezenet*. Pe*model*an klasifikasi dilakukan dengan menggu- nakan *SqueezeNet* untuk proses ekstraksi fiturdan *Naïve Bayes, Support Vector Machine* untuk proses klasifikasi.
3	Tri Septiana Nadia Puspita Putri, Mohamad Al Fikih, Novendra Setyawan tentang “*Face Mask Detection Covid19 Using Convolutional Neural Network*” tahun 2020	Membahas tentangpenerapan sistem deteksi masker dengan menggunakan pengolahan citra. Perancangan sistemdalam penelitian ini menggunakan kombinasi klasifikasi dari pendeteksian objek, gambar dan pelacakan objek untuk mengembangkan sistem deteksi masker dalam bentuk gambar maupun video. *Dataset* diambil dengan berbagai macamvariasi antara lain gambaryang memakai hijab, topi maupun yang tidakmemakai atribut apapun, dan juga gambar yangberasal dari berbagai negaraantara lain asia, eropa, dan amerika. Penelitian ini menghasilkan nilai akurasi sebesar 0,9933% dan training loss 0,0213%.
4	Penelitian yang dilakukan oleh Muhammad Aminullah dengan judul“Alat Deteksi Masker	Pembuatan alat untuk mendukung penyandang tunanetra dalam menghadapi *covid-19*. Penelitian ini menggunakan CNN dengan meng-

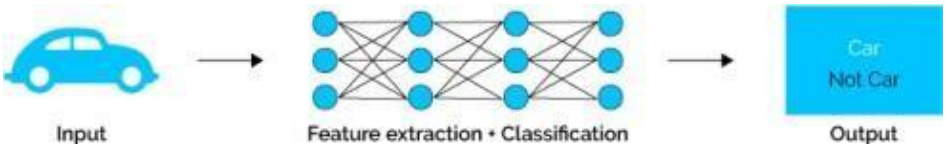
No	Jurnal	Keterangan
	Dengan Metode *Convolutional Neural Network* Untuk Tunanetra Pada Era New Normal" tahun 2021	gunakan *tensorflow* sebagai *framework deep learning* yang dapat mengenali dan mengklasifikasikan suatu objek. Sistem deteksi masker dan jarak di implementasikan pada Rasberry Pi dan mendapat FPS sebesar 0,33. Hasil pengujian dapat diterima melalui *earphone* dan sistem pengawasan pada alat ini antara lain pengasuh dapat melakukan monitoring secara real- time suhu badan, keadaan dan lokasi keberadaan penyandang tunanetra melalui aplikasi smartphone dan pada aplikasi dilengkapi API *Google Maps* untuk menampilkan lokasi penyandang tunanetra.
5	Nyoman Purnama, Putu Kusuma Negara melakukan penelitian tentang "Deteksi Masker Pencegahan *Covid19* Menggunakan *Convolutional Neural Network* Berbasis *Android*" tahun 2021	Membuat sebuah perbandingan dua metode optimasi pada deep learning yaitu *adam* dan *gradient descent* dan tujuandari dilakukan proses penguji untukmengetahui nilai *recall,* presisi dan akurasi setiap *optimizer*. Pengujian diproses menggunakan perangkat dengan berbasis *android* dan penggunaanbahasa pemograman *java* serta *framework mobilenetv2.* Penelitian yang dilakukan ini meng hasilkan akurasi sebesar 90% pada

No	Jurnal	Keterangan
		saat menggunakan *adam* dan saat menggunakan optimasi *gradient descent* menghasilkan angka sebesar 80%
6	Penelitian dengan judul "*Convolutional Neural Network* Arsitektur *MobileNet-V2* Untuk Mendeteksi Tumor Otak" yang dilakukan oleh Widi Hastomo, Sugiyanto dan Sudjiran tahun 2021	Melakukan penelitian untuk memprediksi penyakit tumor otak yang diderita oleh pasien yang diprediksi dari kemampuan pembacaan *image* dari peralatan CT Scanner dengan menggunakan metode CNN. Penggunaan arsitektur MobilenetV2 memiliki fungsi untuk digunakan sebagai *training* data dan *testing* data dengan total 2.870 *image* tumor otak. Penelitian yang dilakukan menghasilkan sebuah nilai *training* akurasi sebesar 97% dan nilai 94% untuk nilai *testing* serta nilai disetiap akurasi pada setiap klasifikasi yang dilakukan antara lain *glioma* sebesar 99%, *meningioma* sebesar 85%, *no tumor* sebesar 99% dan *pituaty* sebesar 96%. Hasil akurasi yang dilakukan mendapatkan hasil yang sangat baik dan menghasilkan sebuah model yang memiliki manfaat untuk mendiagnosa pasien dengan cepat, murah dan akurat.

2.2 DASAR TEORI

2.2.1 Deep Learning

Salah satu cabang dari *Machine Learning* yang tersusun dari algoritma pemodelan abstraksi tingkat tinggi pada *data* dengan menggunakan sekumpulan fungsi transformasi *non-linear* dengan berlapis-lapis dan mendalam yaitu pengertian dari *Deep Learning*. Pada pengaplikasiannya *deep learning* banyak digunakan untuk *speech recognition* pada ponsel pintar, analisa *video* dan citra, 12(dua belas) klasifikasi teks dan sebagainya. Teknik dan algoritma dalam *deeplearning* dapat digunakan untuk kebutuhan *supervised learning* (pembelajaran terarah), *unsupervised learning* (pembelajaran tak terarah), dan s*emi-supervised learning* (semi-terarah). Struktur dan jumlah jaringan saraf pada algoritmanya sangatbanyak bisa mencapai ratusan lapisan. Terdapat dua istilah penting dalam pembangunan *model* yaitu pelatihan dan pengujian. Pelatihan atau *training* adalah proses konstruksi *model* sedangkan pengujian atau *testing* merupakan proses menguji kinerja dari *model* hasil pembelajaran. Pelatihan biasanya menggunakan sebuah kumpulan *data* atau biasa disebut *dataset* yang berupa sampel *data* dalam statistika, dapat berupa citra [9].



Gambar 2.1 Arsitektur *Deep Learning*[15]

Seperti yang terlihat pada Gambar 2.1 Arsitektur Deep Learning[15] pada deeplearning tahap *feature extraction* dan *classification* berada dalam satu tahapan [15] .

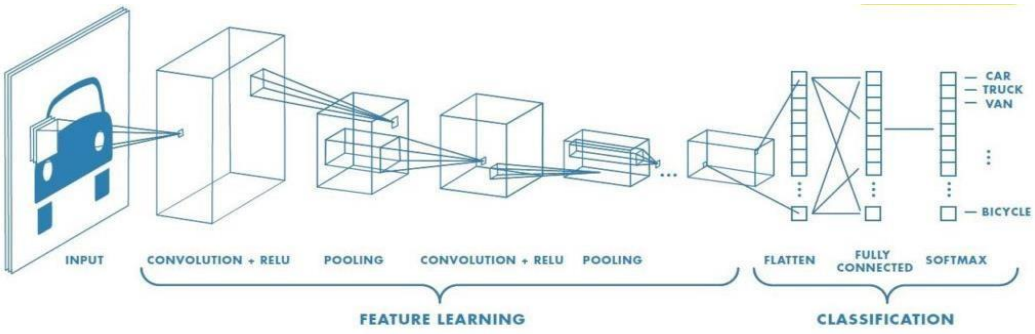
2.2.2 Coronavirus Disease 2019 (Covid-19)

Coronavirus Disease 2019 (Covid-19) mempunyai pengertian sebuah penyakit dapat menular dengan penyebabnya yaitu *Severe Acute Respiratory Syndrome Coronavirus* 2 atau dengan singkatan SARS-CoV2 [1]. *Virus* jenis baru yang biasa dikenal dengan nama *covid-19* yaitu *virus* corona yang tidak pernah ter

identifikasi sebelumnya dalam tubuh manusia. Ciri-ciri kemunculan ataupertanda dan gejala saat terpapar infeksi *virus* ini antara lain sakit batuk, demam,dan parahnya hingga sesak napas. *Virus* ini juga dapat menimbulkan penyakit berat diantaranya sindrom pernapasan akut , gagal ginjal, pneumonia, dan bahkan dapat menyebabkan kematian. Informasi tentang *virus covid-19* pertamakalinya diketahui berasal dari Kota Wuhan, Provinsi Hubei, China pada 31 Desember 2019. Pada tanggal 30 Januari 2020, *World Health Organization* (WHO) menetapkan bahwa kejadian tersebut sebagai Kedaruratan Kesehatan Masyarakat yang Meresahkan Dunia (KKMMD) dan pada tanggal 11 Maret 2020 *Covid-19* ditetapkan sebagai pandemi, sebab *virus* ini dapat menular sangatcepat dan sangat berbahaya [1].

2.2.3 Convolutional Neural Network (CNN)

Pengembangan dari *MultilayerPerceptron* (MLP) yang di *desain* untuk mengolah *data* dua dimensi merupakan pengertian dari *Convolutional Neural Network (CNN)*. CNN dapat mengetahui informasi dari suatu objek seperti citra,teks, potongan suara dan sebagainya berupa *data*. Namun paling banyak digunakan pada bidang pemrosesan citra.



Gambar 2.2 Alur Kerja CNN [6]

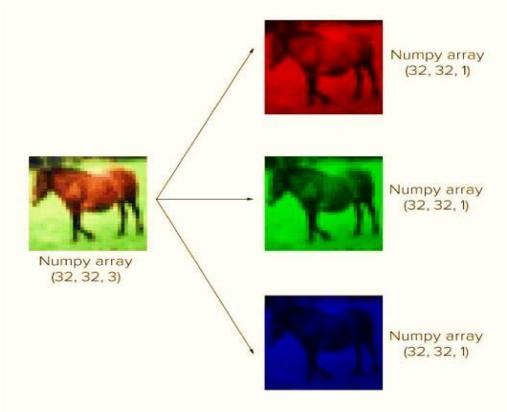
Gambar 2.2 memperlihatkan sebuah alur dari proses CNN ketika melakukanpemrosesan citra dengan *feature learning* menggunakan *convolutionrelu* kemudian *pooling* hingga proses klasifikasi citra dengan menggunakan*flatten, fully connected* dan *softmax* sehingga citra dapat diklasifikasikan ke kategori tertentu berdasarkan nilai keluarannya.

Alur Kerja CNN seperti pada Gambar 2.2 Alur Kerja CNN dapat

dikategorikan memiliki lima komponen utamapada *layer* atau lapisannya yaitu sebagai berikut :

1. *Input Layer*

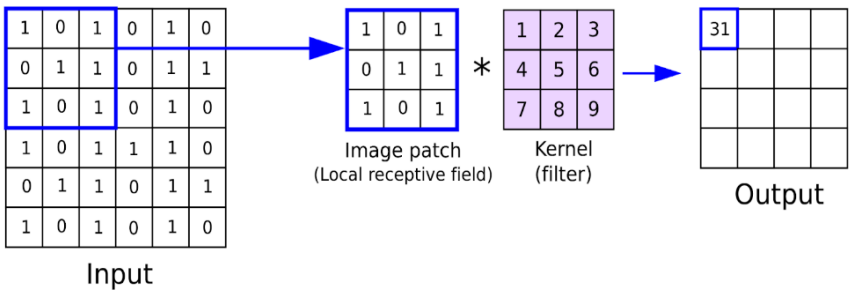
Lapisan masukkan dapat berupa sebuah citra RGB (*Red, Green, Blue*) dengan ukuran32x32 piksel yang sebenarnya merupakan sebuah *multidimensional array* dengan ukuran 32x32x3. Nilai 3 terakhir merupakan jumlah dari kanal. Contoh dari citra tersebut ditunjukkan pada Gambar 2.3.



Gambar 2.3 *Citra Input Layer* [6]

2. *Convolutional Layer*

Lapisan ini merupakan lapisan yang pertama kali menerima masukkan citra langsung pada arsitektur.Pada lapisan ini juga melakukan kombinasi *linier filter* terhadap daerah lokal seperti operasi konvolusi. Seperti layaknya *citra, filter* lapisan pada proses konvolusi memiliki ukuran tinggi, lebar dan tebal tertentu. Alur padalapisan konvolusi dapat digambarkan pada gambar 2.4.



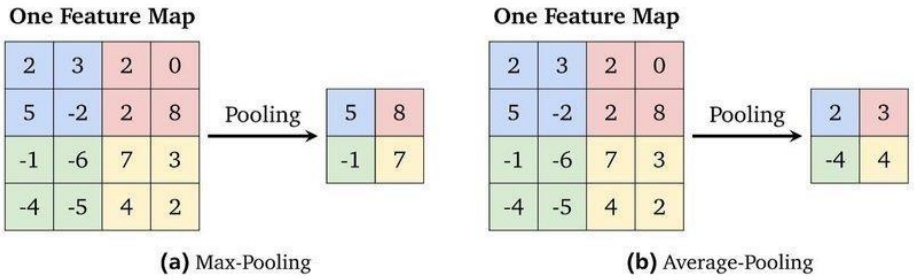
Gambar 2.4 Alur *Convolutional Layer* [6]

3. Activation Layer

Lapisan aktivasi mempunyai pengertian lapisan dimana *feature map* dimasukan ke dalam fungsi aktivasi. mengubah nilai yang ada pada *feature map* pada *range* tertentu sesuai dengan fungsi aktivasi yang digunakan adalah fungsi dari lapisan aktivasi. Tujuan dilakukan hal tersebut untuk meneruskan nilai yang menampilkan fitur dominan dari citra yang masuk ke lapisan berikutnya. Terdapatbeberapa fungsi aktivasi yang umum untuk digunakan, namun dalam penelitian hanya menggunakan aktivasi *ReLu* dan *Softmax.*

4. Pooling Layer

Selanjutnya masukan dari lapisan aktivasi akan menuju *pooling layer*, kemudian lapisan ini akan mengurangi parameternya. *Pooling* juga bisa disebut sebagai *subsampling* atau *downsampling* yang berfungsi untuk mengurangi dimensi dari *feature map* tanpa menghilangkan informasi penting di dalamnya. Proses padalapisan ini cukup sederhana, dimana dengan menentukan ukuran *down sampling* yang akan digunakan pada *feature map*, sebagai contoh 2x2 yang selanjutnya akandilakukan proses *pooling* pada *feature map*. Ada beberapa macam proses dari *pooling* antara lain *max pooling*, *mean pooling*, dan *sum pooling*. Contoh proses *pooling* ditunjukan pada gambar 2.5.



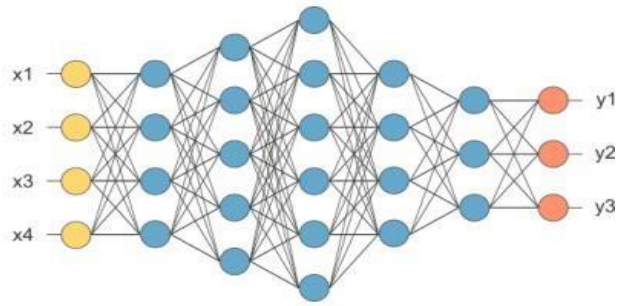
Gambar 2.5 Proses *Pooling* [6]

Setelah dilakukan *pooling layer*, maka dapat diketahui tujuan dari *pooling layer* tersebut yaitu untuk mengurangi dimensi dari *feature map*. Sehingga proses ini gambar 2.5 matriks *feature map* 4x4 dengan proses *pooling* 2x2 akan mempercepat komputasi karena parameter yang harus diperbaharui semakin sedikitdan mengatasi

overfitting.

5. Fully Connected Layer

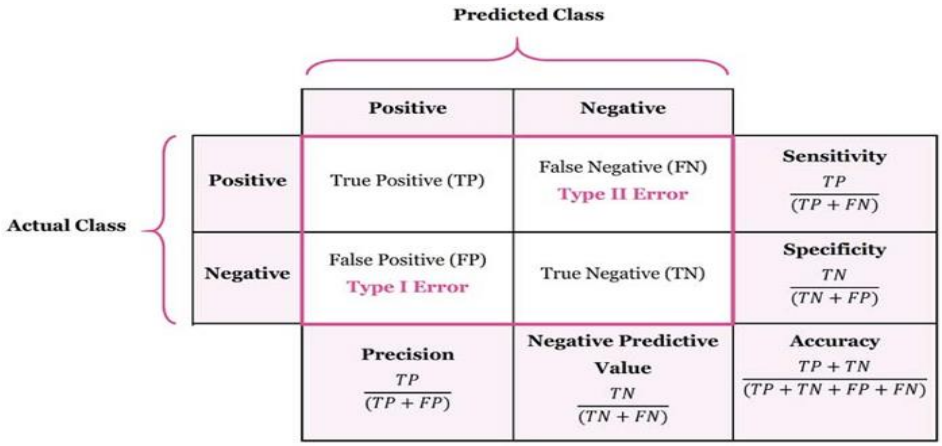
Dilapisan ini bekerja setelah melakukan beberapa lapisan diatas dan menghasilkan *pooling layer* yang digunakan untuk *input*-an untuk *fully connected layer*. Lapisan ini mempunyai kesamaan struktur dengan *Artificial Neural Network* (ANN) yang memiliki lapisan *input*-an, lapisan tersembunyi, dan lapisan *output* yang masing-masing memiliki *neuron* yang saling berhubungan dengan *neuron* yang berada dalam lapisantetangganya. Dalam konsep sebelum *pooling* digunakanuntuk input, hasil *pooling* terlebih dahulu diubah menjadi vektor (x1, x2, x3, x4, dan seterusnya) yang akan diproses ke dalam *fully connected layer* yang nantinya pada lapisan terakhir *fully connected layer* akan digunakan fungsi ReLu atau *softmax*untuk menentukan klasifikasi dari citra masukan yang dari lapisan masukan CNN.Untuk lebih jelas contoh *fully connected layer* pada Gambar 2.6.



Gambar 2.6 *Fully Connected Layer* [6]

2.2.4 Confusion Matrix

Confusion Matrix adalah salah satu metode yang dapat digunakan untuk mengukur kinerja suatu metode klasifikasi. *Confusion Matrix* mengandung informasi yang membandingkan hasil klasifikasi yang dilakukan oleh sistem dengan hasil klasifikasi yang sebenarnya. Pengukuran kinerja dengan *confusion matrix*, terdapat 4 (empat) istilah sebagai representasi hasil proses klasifikasi yang ditampilkan dalam gambar 2.7.



Gambar 2.7 *Confusion Matrix* [20]

Penjelasan gambar 2.7 tersebut adalah *True Positive* (TP) adalah kasus di mana model klasifikasi yang dibuat dengan benar diprediksi *positif*. *False Negative* (FN) adalah kasus di mana model klasifikasi diprediksi salah, tetapi sebenarnya *positif*. Ini juga dianggap sebagai kesalahan Tipe II. *False Positif* (FP) adalah *positif* palsu kasus dimana model klasifikasi diprediksi positif, tetapi sebenarnya negatif. Ini juga dianggap sebagai kesalahan Tipe I. *True Negative* (TN) adalah kasus di mana model klasifikasi memprediksi negatif dengan benar. Setelah hasil dari masalah klasifikasi telah diterima, *matrix* klasifikasi dapat menghitung nilai *Sensitivitas, Spesifisitas*, Akurasi, Nilai Prediktif *Negatif*, dan *Presisi*. Dimana sensitivitas yang biasa disebut *True Positif Rate* atau *Recall* adalah rasio prediksi benar *positif* dibandingkan dengan keseluruhan data yang benar *positif*. *Spesifisitas* juga dikenal sebagai *True Negative Rate* merupakan kebenaran memprediksi *negatif* dibandingkan dengan keseluruhan data *negatif*. Akurasi merupakan rasio prediksi benar (*positif* dan *negatif*) dengan keseluruhan data. Nilai *Prediktif Negatif* merupakan hasil yang diberi label dengan benar sebagai salah. *Presisi* merupakan rasio prediksi benar *positif* dibandingkan dengan keseluruhan hasil yang diprediksi *positf*. *F1-Score* merupakan perbandingan rata-rata *presisi* dan *recall* yang dibobotkan *F1 Score* = 2 * *(Recall*Precission*) / (*Recall + Precission*) [20].

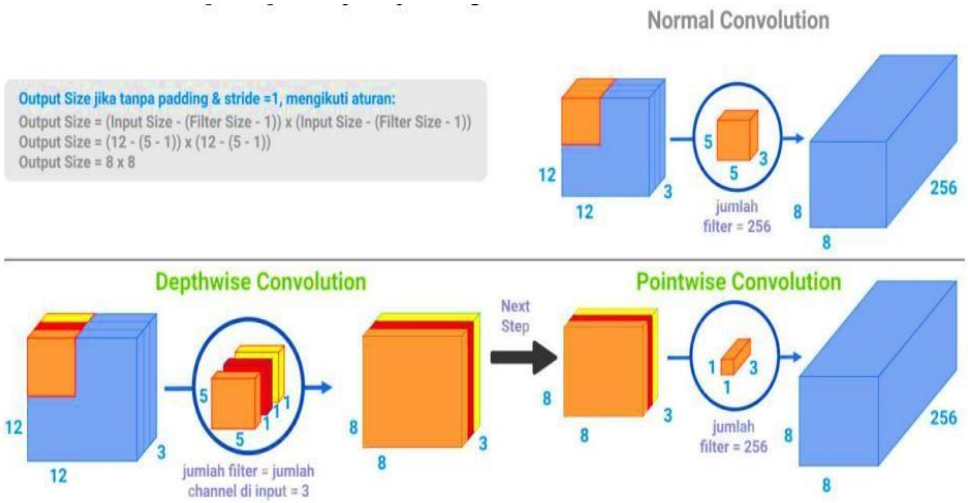
2.2.5 Arsitektur CNN *MobilenetV2*

Berbagai macam arsitektur CNN mulai banyak bermunculan, dan masing-masing arsitektur berlomba-lomba agar memperoleh skor akurasi tertinggi yang di

training terhadap *ImageNet*. Pengertian dari *ImageNet* sendiri merupakan sekumpulan *dataset* gambar yang berjumlah sangat besar dan dirancang oleh akademisi yang ditujukan untuk penelitian *computer vision*.

Dalam penelitian ini, pemilihan arsitektur CNN *MobileNetV2* dikarenakan skor akurasinya cukup tinggi dan fungsi utamanya ialah perbandingan jumlah *training parameters* yang kecil dibandingkan dengan arsitektur CNN yang lain sehingga kebutuhan akan komputasinya jauh lebih ringan dan juga *model size MobileNetV2* cukup kecil dengan ukuran 14MB dan performansi yang cukup baik sehingga kedepannya pada saat *model* akan di *deploy* kedalam sebuah *real app*, seperti dibuat menjadi sebuah aplikasi *android* maupun aplikasi berbasis *website* ringan dan berukuran kecil.

Sebuah perancangan yang dibuat oleh *Google* dan memiliki *layer* khusus yang disebut dengan *depthwise separable convolution* ialah *MobileNet*. Layer ini memiliki fungsi untuk mereduksi komputasi sehingga menghasilkan ukuran *model* yang lebih kecil [10].



Gambar 2.8 *Normal Convolution vs Depthwise Separable Convolution* [10]

Penjelasan gambar 2.8 *filter* melakukan sebuah konvolusi terhadap *input image* dan juga membentuk sebuah *feature map* atau *output image* pada *layer normal convolutional*, dan dalam *depthwise convolution*, jumlah *filter* sama dengan jumlah *channel* yang ada pada *input image*. Setiap *filter* nantinya melakukan

konvolusi terhadap masing-masing *channel* pada *input image* (atau dengan kata lain, secara tidak langsung melakukan konvolusi terhadap seluruh *channel* pada *input image* seperti yang dilakukan pada *normal convolution*). Kemudian agar *output image* dengan *depthwise convolution* sama dengan *output image* yang dihasilkan dengan *normal convolution*,akan dilakukan *pointwise convolution* dan hasil dari tahap *depthwise convolution* akan dikonvolusi lagi dengan filter 1x1, filter ini memiliki kedalaman yang sama dengan jumlah *channel* di *output image* sebelumnya.

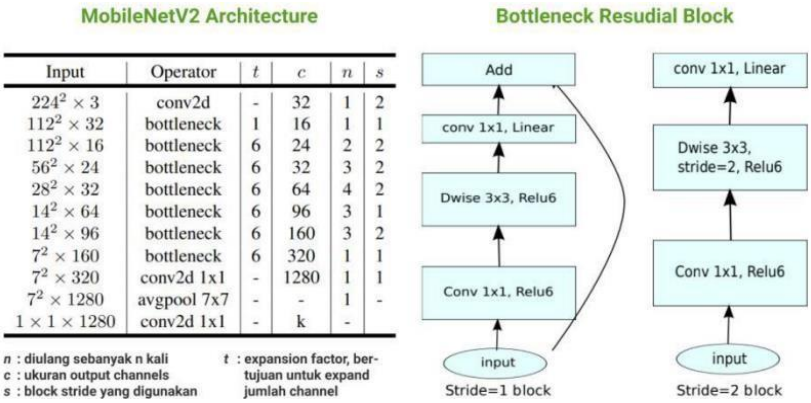
Perbandingan jumlah komputasi antara *normal convolution* dibandingkan dengan *depthwise separable convolution* berdasarkan contoh gambar 2.7 *Normal Convolution vs Depthwise Separable Convolution* [10].

1. *Normal Convolution*

Ada 256 *filter* berukuran 5x5x3 yang bergerak sebanyak delapan kali delapan jumlah pergeseran *filter* 5x5 dari kiri atas ke kanan bawah terhadap *input image* 12x12). Yang mempunyai arti ketika proses konvolusi 256x5x5x3x8x8=1.228.800 total perkalian yang dilakukan.

2. *Depthwise Separable Convolution*

Dalam tahap *depthwise convolution* terdapat 3(tiga) *filter* 5x5x1 yang bergeser sebanyak delapan kali delapan dengan arti 3x5x5x1x8x8=4.800 yang kemudian, dalam *pointwise convolution* terdapat 256 filter 1x1x3 yang bergeser sebanyak delapan kali delapan juga memiliki arti 256x1x1x3x8x8=49.152 yang jika ditotal perkalian yang dilakukan saat proses konvolusi pada *depthwise separable convolution* ini yaitu 4.800+49.152=52.952. Dapat disimpulkan bahwa jumlah 52.952 hanya kurang lebih sekitar 22,7% dari 1.228.800 pada *normal convolution*.

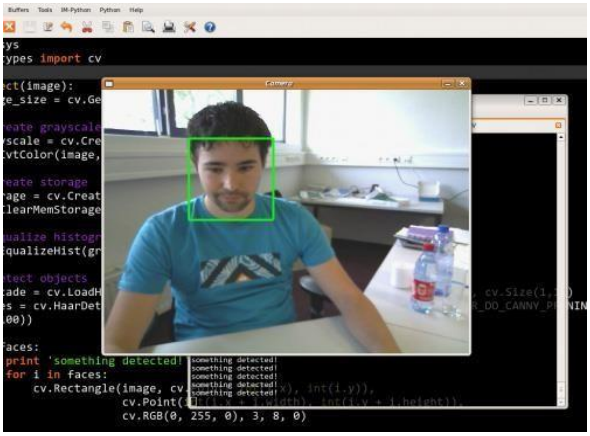


Gambar 2.9 Arsitektur *MobileNetV2* [10]

MobileNetV2 merupakan perkembangan lebih lanjut dari *MobileNet* dimana ditambahkan fitur baru yaitu sebuah blok dalam arsitektur yang didalamnya menggunakan *depthwise separable convolution* [10].

2.2.6 Modul DNN *OpenCV*

Model caffe yang didasarkan pada *Single Shot-Multibox Detector* (SSD) dan menggunakan arsitektur *ResNet*-10 sebagai tulang punggungnya merupakan penjelasan Modul *Deep Neural Network* (DNN) *OpenCV*. *OpenCV* membuat pendeteksian pengenalan wajah dapat dilakukan dengan menggunakan *model* pendeteksi *wajah deep learning* yang sudah terlatih. Penggunaan dari modul DNN *OpenCV* dengan *model caffe* membutuhkan *prototxt* yang mengartikan arsitektur *model ResNet10* dan *caffe model* merupakan sebuah *file* yang berisi bobot untuk lapisan sebenarnya. Pendeteksian wajah dengan menggunakan *OpenCV* pada gambar 2.10 Deteksi dengan *OpenCV* [11].



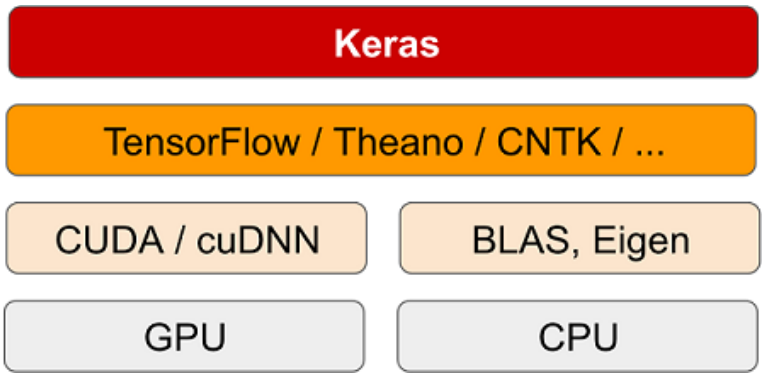
Gambar 2.10 Penerapan deteksi dengan *OpenCV* [14]

2.2.7 Google Colaboratory

Google Colab atau biasa disebut *Google Colaboratory* adalah sebuah produk *Google* berbasis *cloud* yang bisa diakses atau digunakan secara gratis. Perbedaan dengan produk *Google* lainnya yaitu pada *coding environment Google Colab* bisa diaplikasikan dengan bahasa pemrograman *Python* dengan format "*notebook*" yang memiliki kesamaan dengan *Jupyter Notebook,* yang seakan-akan *Google* memberikan fasilitas untuk para *programmer* atau *researcher* untuk menggunakan sebuah komputer secara gratis tetapi dengan spesifikasi yang tinggi [13].

2.2.8 Framework Tensorfow dan *Keras*

Cukup banyak *framework* yang dapat digunakan untuk pengguna *deep learning*, antara lain ada *framework TensorFlow* dan *Keras*. Kedua *framework* tersebut merupakan yang paling banyak digunakan dalam dunia *deep learning*. *TensorFlow* merupakan sebuah platform sumber terbuka untuk *deep learning*. Ini merupakan sebuah *library* dengan API level tinggi, dengan begitu TensorFlow dapat membantu dalam pembuatan *neural network* dalam skala yang besar. *framework deep learning* untuk *python* yang menyediakan cara mudah untukmendefinisikan dan melatih hampir semua jenis *model deep learning* pengertian dari Keras. Awal dikembangkan *Keras* mempunyai tujuan untuk mempercepat kemungkinan bereksperimen.



Gambar 2.11 Perangkat Lunak dan Perangkat Keras *Deep Learning*[15]

Penjelasan gambar 2.11 menjelaskan bahwa *TensorFlow, Keras* dapat berjalandengan mulus di CPU dan GPU, ketika berjalan pada CPU, *TensorFlow*

sendiri akanmelibatkan *library* tingkat rendah untuk operasi tensor yang disebut dengan *Eigen* edangkan pada GPU, *TensorFlow* akan melibatkan *library* operasi *deep learning* yang dioptimalkan dengan baik menggunakan cuDNN atau biasa disebut dengan *library* NVIDIA CUDA *Deep Neural Network* [12].

2.2.9 Spyder

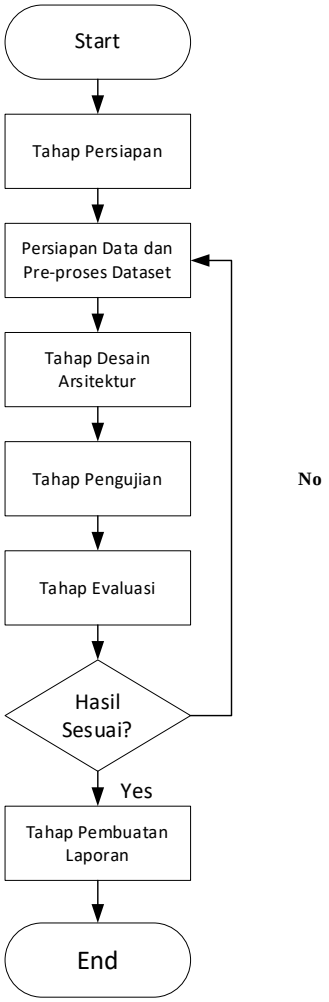
Spyder adalah *development environment* terintegrasi ilmiah yang ditulis dengan *python*. *Software* ini dirancang untuk dan oleh para *data scientist* yang terintegrasi dengan beberapa *library python* seperti Mat*plot*lib, SciPy, NumPy, Pandas, Cython, I*Python*, SymPy, dan *software open source* lainnya. *Spyder* tersedia melalui distribusi *Anaconda* di Windows, MacOS, dan Linux. *Software* ini dapat diakses secara gratis dengan berbagai fitur, seperti dapat menjalankan kode *python* berdasarkan sel, baris, atau *file*, menyediakan *histogram* dan *plot time series*, penyelesaian kode otomatis dengan pemisah *horizontal* atau *vertikal*, dan lain sebagainya [14].

BAB 3

METODE PENELITIAN

3.1 ALUR PENELITIAN

Dalam melakukan sebuah penelitian memerlukan alur penelitian yang diharapkan proses yang dilakukan dapat berjalan dengan baik, lebih terstruktur dan lebih efisien. *Flowchart* merupakan salah satu bentuk alur penelitian dan fungsi sebuah *flowchart* adalah untuk menjelaskan secara singkat alur atau proses dari penelitian. *Flowchart* dalam penelitian ini ditampilkan pada gambar 3.1.



Gambar 3.1 *Flowchart* Alur Penelitian

Bagan *flowchart* pada Gambar 3.1 menjelaskan secara singkat bahwa

penelitian dimulai dengan melakukan studi litelatur pada tahap persiapan diantaranya melakukan pencarian dan membandingkan beberapa jurnal yang digunakan sebagai referensi melalui internet yang dapat digunakan untuk bahan risetdalampenelitian ini. Blok tahap persiapan *data* dan *pre*-proses *dataset* merupakan tahap menyiapkan *data* yang dibutuhkan dan melakukan *pre*-proses *dataset* dengan cara mengumpulkan gambar seseorang yang memakai masker dan tidak memakai masker kemudian *data* tersebut dilakukan proses *data cleaning* yang berfungsi untuk membersihkan *data* yang dianggap tidak akurat atau yang tidak dibutuhkan agar bisa diproses dengan baik pada tahap *training data* dan *training data* digunakanuntuk melatih algoritma dalam mencari *model* yang sesuai. *Dataset* pelatihan menggunakan *dataset* dari proyek *chandrikadeb7* dengan judul "*Face Mask Detection*" pada *Github* yang selanjutnya *data*s`et tersebut di*download* dan di*upload* kembali kedalam penyimpanan baru dengan nama */tianpresti6/Face-Mask-Detection-Using-CNN*. Selanjutnya *dataset* tersebut dimasukkan ke *Google Colaboratory.*

Blok tahap desain arsitektur dimana tahapan yang dimulai dari membuat kode sumber untuk program CNN, meng-*import Framework Keras* dan *TensorFlow* dengan menggunakan infrastruktur *Google Colaboratory* dan bahasa pemrograman *Python* disimpan kedalam bentuk *file* Jupyter Notebooks ".ipynb" yang selanjutnya akan disimpan ke *Github*. Selanjutnya *dataset* pelatihan diambil yang kemudian disimpan ke dalam tempat penyimpanan sementara pada *Google Colaboratory* dari *Github*. Sistem pendeteksian masker wajah dirancang menggunakan arsitektur jaringan CNN yang bernama *MobileNetV2* yang dalam sistem pembuatan *model* sudah dipelajari sebelumnya pada *base model* dan *file* tersebut dilakukan *feature extraction* dengan kata lain melakukan pengambilan ciri dari yang digunakan untuk menganalisa proses selanjutnya dan kemudian ditambahkan *model* arsitektur tambahan pada *head model atau* kerangka *model*.

Blok pada tahap pengujian merupakan tahap dimana seluruh *dataset* pelatihan dilatih terhadap arsitektur jaringan CNN yang dibuat dengan panjang *epochs* 80. Setelah mencapai hasil prediksi yang memuaskan, kemudian dilakukan pengujian pada *Google Colabolatory* dan sistem pendeteksi masker wajah. Pengujian yang dilakukan pada *Google Colabolatory* dengan menggunakan bantuanpendeteksi wajah

yaitu SSD *ResNet10* sedangkan pada *video* dengan cara mengkonversi hasil *model* pelatihan ke dalam format ".h5" yang kemudian *model* tersebut dicoding pada *Spyder* dan di implementasikan pada sistem pendeteksian masker wajah.

Blok tahap evaluasi dilakukan untuk memperbaiki sistem jika terjadi kesalahan pada sistem tersebut, sehingga dapat memperolehhasil sistem yang baik dan layak digunakan untuk banyak orang. Jika tidak terjadi kesalahan dan sistem mudah digunakan oleh banyak orang, maka dapat disimpulkan bahwa sistem layak dan siap untuk digunakan.

Dalam blok hasil, jika hasilnya sesuai maka dilanjutkan dengan pembuatan laporan sedangkan jika hasil tidak sesuai maka kembali pada tahap desain arsitektur. Pada blok pembuatan laporan merupakan tahap pembuatan laporan penelitian yang sudah dilakukan.

3.2 ALAT DAN BAHAN

3.2.1 Perangkat Keras (*Hardware*)

Penelitian kali ini menggunakan 1 (satu) perangkat untuk menjalankan *software* dan kamera pada laptop digunakan untuk mendapatkan hasil pengujian pada penelitian Deteksi Masker Wajah dengan Metode CNN, dengan keterangan spesifikasi pada tabel 3.1

Tabel 3.1 Spesifikasi Laptop

Spesifikasi	
Processor	*Intel Core* I5-2430M (2.40 GHz (4 CPUs) – 2.4 Ghz)
RAM	6 GB DDR3
SSD	240 GB
VGA	Intel(R) HD *Graphics* 3000

3.2.2 Perangkat Lunak (*Software*)

Penelitian ini menggunakan perangkat lunak sistem operasi *Windows*dengan *tools* dan *software* untuk melakukan penelitian Deteksi Masker Wajah Dengan Metode CNN pada tabel 3.2

Tabel 3.2 Perangkat Lunak (*Software*)

No.	Nama *Software*	Fungsi
1	*Windows* 10	Sistem Operasi
2	*Google Chrome*	Menjelajah Situs *Web*
3	*Google Colaboratory*	*Jupyter Noteboook* Versi *Cloud*
4	*Spyder*	untuk *running project*

3.2.3 Dataset

Dataset pelatihan dan pengujian berupa gambar orang menggunakan masker dan tidak menggunakan masker didapatkan dari *https://github.com/chandrikadeb7/Face-Mask-Detection* yang di download dan di upload kembali kedalam *reporsitory* baru dengan nama */tianpresti6/Face-Mask-Detection-Using-CNN* ke dalam *Github.*

Penelitian ini menggunakan *dataset* yang terdiri dari 2(dua) jenis didalamnya antara lain gambar wajah yang mengenakan masker dan wajah yang tidak mengenakan masker. *Dataset* tersebut nantinya berfungsi untuk melakukan pembuatan *model* yang akan dilakukan di *Google Colaboratory*. *Model* yang telah dibuat akan dilakukan pengujian pada *Google Colaboratory* dan sistempendeteksian masker menggunakan bantuan SSD *Resnet10* yang selanjutnya akan dikonversi menggunakan format "h5" yang nantinya akan di implementasikan pada sistem pendeteksi masker pada *Software Spyder*. Dalam tahap pre-proses *data* dilakukan sebuah proses yang bernama *data cleaning* untuk membersihkan *data* yang dianggap tidak akurat atau yang dianggap tidak dibutuhkan agar bisa diproses dengan baik pada tahap *training data* agar dapat digunakan untuk melatihalgoritma untuk memperoleh *model* yang sesuai.

Dataset yang digunakan pada pendeteksian masker *covid-19* ini ada 3 macam, antara lain *dataset* pelatihan , *dataset* validasi dan *dataset* pengujian. *Dataset* pelatihan merupakan *dataset* yang memiliki fungsi untuk dijadikan sebuah data pelatihan untuk mendapatkan sebuah *model* yang sesuai dengan apa yang diinginkan, sedangkan *dataset* pengujian adalah *dataset* yang berfungsi untuk

menguji tingkat keakurasian yang diinginkan dari *model* tersebut. Keseluruhan *dataset* yang digunakan pada penelitian ini menggunakan *dataset* berwarna RGB (*Red,Green,Blue*) dan diubah ukurannya menjadi 224x224x3pixel.

1. *Dataset* yang digunakan untuk Pelatihan

Penelitian ini menggunakan *dataset* proyek *chandrikadeb7* dengan judul '*Face Mask Detection*" pada *Github* yang telah di *download* dan di *upload* kembali kedalam penyimpanan baru dengan nama */tianpresti6/Face-Mask-Detection- Using-CNN* dengan jumlah *dataset* 3833 *data* gambar yang terdiri dari wajah yangmenggunakan masker maupun wajah yang tidak menggunakan masker. *Dataset* pelatihan ini nantinya akan diambil sebanyak 75% dari kedua jenis gambar bermasker maupun tidak bermasker dengan total 2874 data dari total keseluruhan 3833 *data*.

2. *Dataset* yang digunakan untuk Validasi

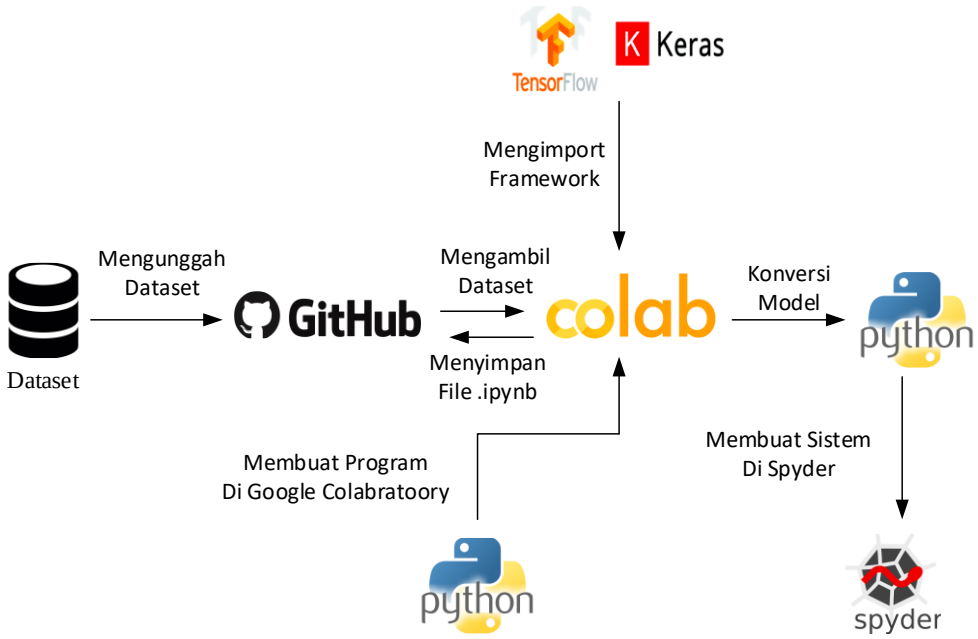
Dataset validasi didalam penelitian ini digunakan sebanyak 25% dari *dataset* pelatihan dari total *dataset* sebanyak 959 *data* dari total keseluruhan 3833 *data*. *Dataset* ini berfungsi untuk menguji dan membandingkan hasil dari proses pelatihan dengan *dataset* pelatihandi setiap *epoch*-nya.

3. *Dataset* yang digunakan untuk Pengujian

Dataset yang digunakan untuk pengujian berjumlah sama seperti pada proses validasi. *Dataset* ini bertujuan untuk menguji hasil dari proses pelatihan pada jaringan CNN yang digunakan.

3.3 PERANCANGAN SISTEM

Penelitian dilaksanakan melalui beberapa tahapan yaitu tahap persiapan dengan studi litelatur, tahap persiapan *data* dan *pre*-proses *dataset*, tahap *desain* arsitektur, tahap pengujian dan tahap evaluasi. Arsitektur sistem dibuat seperti yang ditampilkan pada Gambar 3.2

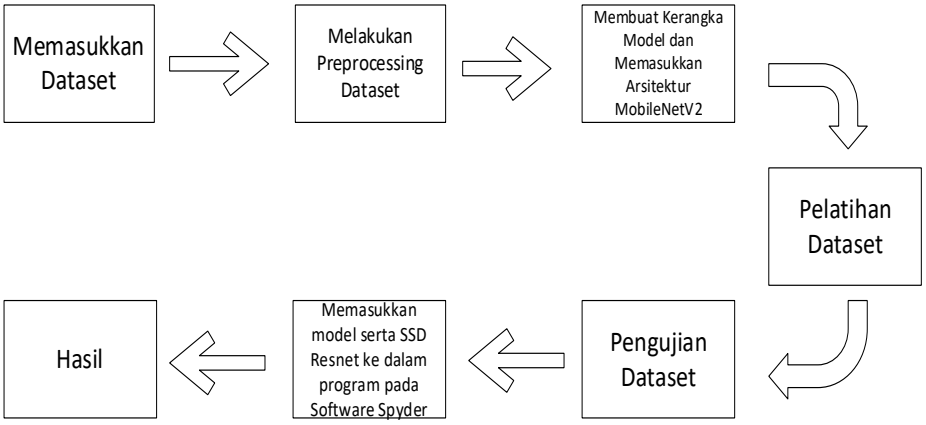


Gambar 3.2 *Desain* Arsitektur Sistem

Penjelasan arsitektur sistem pada gambar 3.2, pada langkah pertama pembuatan sistem yaitu dengan menggunggah *dataset* proyek *chandrikadeb7*dengan judul '*Face Mask Detection*" pada *Github* yang telah di *download* dan di *upload* kembali kedalam *reporsitory* baru dengan nama */tianpresti6/Face-Mask- Detection-Using-CNN* ke dalam *Github* selanjutnya mengambil *dataset* dari *reporsitory* baru yang selanjutnya membuat program pada *google colaboratory* menggunakan bahasa pemrograman *python* dan meng-*import framework tensorflow* dan *keras* yang digunakan untuk membuat model dan disimpan kembali ke dalam *github* dengan format file ".ipynb", kemudian model dikonversikedalam format .h5 yang kemudian hasil konversi model digunakan pada program*python* pada *spyder* untuk pendeteksian wajah secara real-time.

3.4 ALUR SISTEM

Untuk mendapatkan hasil yang di inginkan dimulai dengan melewati beberapa alur sistem pada penelitian pendeteksian masker seperti yang digambarkan pada blok diagram gambar 3.3.



Gambar 3.3 Alur Sistem

Diagram alur sistem pada gambar 3.3 menjelaskan bahwa pada langkah pertama dilakukan dengan memasukan *dataset* pelatihan ke dalam *Google Colaboratory* selanjutnya melakukan *preprocessing* data untuk menormalisasikan *dataset* dimana dalam tahapan ini dilakukan mengubah dari data gambar menjadi data numerik dan menyesuaikan dengan format model agar proses komputasi berjalan dengan baik. Kemudian memasukkan arsitektur *MobileNetV2* ke dalam kerangka model yang kemudian mempersiapkan layer-layer yang akan digunakan untuk membuat model yang di inginkan dengan melakukan pengurangan ukuran *matrix* dan mengurangi *volume ouput* pada *feature map* dan mengambil nilai rata-ratanya, selanjutnya mengubah gambar yang mempunyai banyak dimensi menjadi hanya 1(satu dimensi) dan membuat jaringan bekerja secara efisien dan mempercepat waktu komputasi serta membuat prediksi model dengan nilai probabilitas 1(satu) dan 0(nol) yang selanjutnya mempersiapkan model. Kemudian setelah melakukan proses perancangan kerangka model tahapan selanjutnya melakukan proses pelatihan pada model dengan memasukkan data *training*, menentukan step per *epoch*, validasi data, dan melakukan langkah setiap *epoch* validasi dan juga menentukan *epoch* sebanyak 80 *epoch* dan batch size sebanyak

64 yang bertujuan untuk mempercepat proses komputasi yang akan dijalankan. Pengujian hasil *training* dilakukan setelah mencapai hasil prediksi yang di inginkan dan pengujian dilakukan didalam Google Colabolatory. Setelah selesai dilakukan pengujian pada *Google Colaboratory* selanjutnya model yang telah dirancang dimasukkan bersama SSD *Resnet10* yang berfungsi untuk mendeteksi wajah kedalam program pada *Software Spyder* untuk melihat hasil dari penelitian ini.

3.5 SKENARIO PENGUJIAN

Skenario pengujian yang dilakukan pada penelitian ini dengan cara seluruh *dataset* pelatihan dilatih terhadap masing-masing arsitektur jaringan CNN yang dibuat dengan panjang 80 *epochs* atau proses *training* sekali putaran dari seluruh *dataset training*. Setelah mencapai hasil prediksi yang memuaskan, dilakukan pengujian pada *Google Colabolatory* dan sistem pendeteksian masker. Pengujian dilakukan pada *Google Colabolatory* menggunakan bantuan pendeteksi wajah menggunakan SSD *ResNet10* sedangkan pada sistem dengan mengkonversi hasil *model* pelatihan ke dalam format ".h5" yang selanjutnya *model* pelatihan tersebut diimplementasikan pada sistem pendeteksi masker menggunakan *spyder*.

3.6 PARAMETER PENGUJIAN

Pengujian parameter pada penelitian ini dilakukan untuk mengetahui akurasi, presisi, *recall* dan nilai F1-*Score* pada pendeteksian masker wajah menggunakan arsitektur *MobileNet-V2* dan *SSD Resnet10*. Selain itu terdapat variasi pengujian yang akan dilakukan seperti perbedaan jarak dan jenis masker muka yang dikenakan.

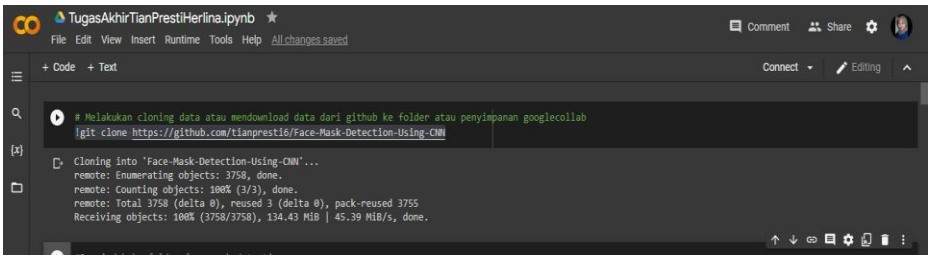
BAB 4

HASIL DAN PEMBAHASAN

4.1 Tahap Desain Arsitektur

Tahap ini melakukan perancangan program untuk mendapatkan *model* yang di inginkan pada *Google Colaboratory* dengan membuat beberapa sel/baris *text* untuk memberikan informasi terkait keterangan dan penjelasan program disetiap sel/baris *text*untuk program CNN dan meng-*import framework* keras dan *tensorflow* yang dibuat menggunakan infrastuktur *Google Colaboratory* dengan menggunakan bahasa pemograman *python* dan disimpan ke dalam bentuk *file jupyter noteboks ".ipynb"* yangselanjutnya akan disimpan ke *github.* Kemudian *dataset* pelatihan akan diambil dan disimpan ke dalam tempat penyimpanan sementara pada *Google Colaboratory* dari *github.* Sistem pendeteksian masker wajah dirancang dengan menggunakan arsitektur jaringan CNN yaitu *MobileNetV2* dengan cara membuat *model* dari jaringan CNN yangsudah dipelajari sebelumnya pada tahap pengujian atau dengan kata lain pada *base model* dan *file* tersebut akan dilakukan *feature extraction* atau mencari ciri khas dari suatu bentuk yang nantinya dapat diproses pada tahap selanjutnya dan kemudian akan ditambahkan *model* arsitektur tambahan berupa *file* yang berfungsi untuk mendeteksi wajah yaitu SSD *Resnet10* pada *head model* atau kerangka *model*.

Langkah awal pembuatan program yaitu men*download dataset* dari *github* kedalam penyimpanan *Google Colaboratory* dengan menggunakan perintah seperti pada gambar 4.1.

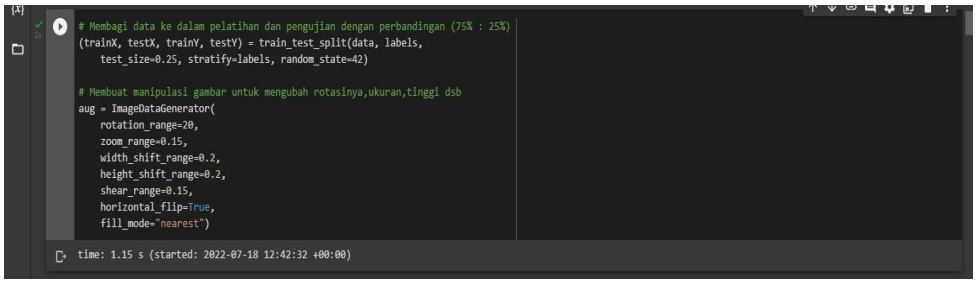


Gambar 4.1 Men*download dataset* dari *github* ke penyimpanan *Google Colaboratory*

Pada gambar 4.1 menjelaskan proses mendownload *dataset* dari github https://github.com/tianpresti6/Face-Mask-Detection-Using-CNN ke dalam penyimpanan yang ada pada Google Colaboratory yang nantinya digunakan untuk proses *training* data.

Proses selanjutnya yaitu melakukan inisialisasi untuk nilai *Learning Rate* atau dengan kata lain parameter *training* untuk menghitung nilai koreksi pada waktu pelatihan dalam penelitian ini 1e-4 sebagai initial *learning rate* dengan pengertian ada 4(empat) angka 0(nol) dibelakang koma, kemudian ada 80(delapan puluh) *Epoch* pelatihan atau ketika seluruh *dataset* sudah melalui proses *training* sampai kembali ke awal untuk sekali putaran dan *Batch Size* sebesar 64(enam puluh empat) atau dengan kata lain jumlah *training sample* untuk sekali *forward-pass* (proses kalkulasi dari *input neural network* ke *output*) dan *backward-pass*(proses kalkulasi dari *output neural network* ke *input*). Kemudian melakukan proses pengambilan gambar dari *folder dataset* yang kemudian melakukan inisialisasi *data* dan *label* (bermasker dan tidak bermasker) selanjutnya melakukan perulangan pada *image paths* dilanjutkan mengesktrak *class label* (bermasker atau yang tidak bermasker) dari *file name* dan memuat *input* gambar berukuran (224x224) dan melakukan proses mengubah gambarjadi *array* dan juga meng*update data* dan *label list* secara berurutan selanjutnya mengkonversi *data* dan *label* ke dalam NumPy *Array*s dan melakukan *one-hotencoding* pada *label* (mengubah *data* kategorik ke numerik).

Selanjutnya membagi *data* ke dalam pelatihan dan pengujian dengan perbandingan 75% : 25% dan membuat manipulasi gambar untuk mengubah rotasi,ukuran maupun tinggi pada *data* seperti gambar 4.2.



Gambar 4.2 Pembagian perbandingan data untuk pelatihan da pengujian dan juga mengubah rotasi,ukuran,maupun tinggi data.

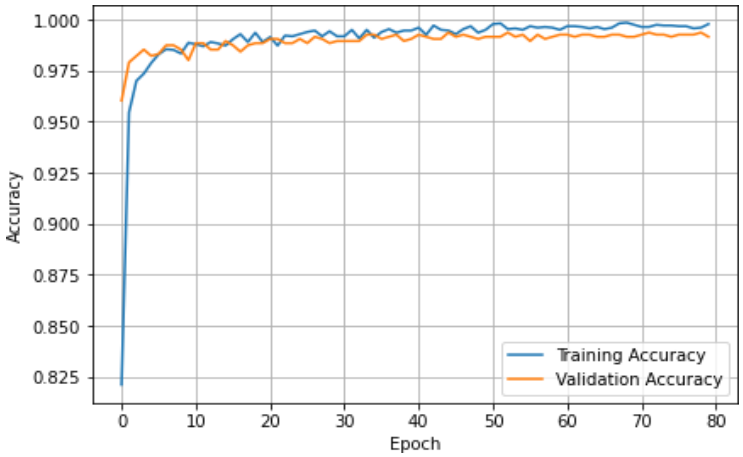
Kemudian melakukan pembentukan head dari model yang nantinya akan ditempatkan pada base model (inputan), melakukan penempatan head model pada base model dilanjutkan melakukan perulangan pada seluruh base model dan mempersiapkan penyusunan model seperti pada gambar 4.3.

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

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

# Perulangan pada seluruh base model
for layer in baseModel.layers:
    layer.trainable = False

# Persiapan menyusun model
print("Mengkompilasi model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
    metrics=["accuracy"])
```

Gambar 4.3 Melakukan pembentukan pada *head model* pada *base model* kemudian menempatkan *head model* ke *base model*, perulangan pada seluruh *base model* dan persiapan *model*.

Pada tahap seperti yang ada pada gambar 4.3 proses penyusunan *model* atau dengan kata lain membuat suatu kerangka *model* yang nantinya akan diterapkan pada proses *training data* yang kemudian dilakukan proses pelatihan model yang sudah dibuat.

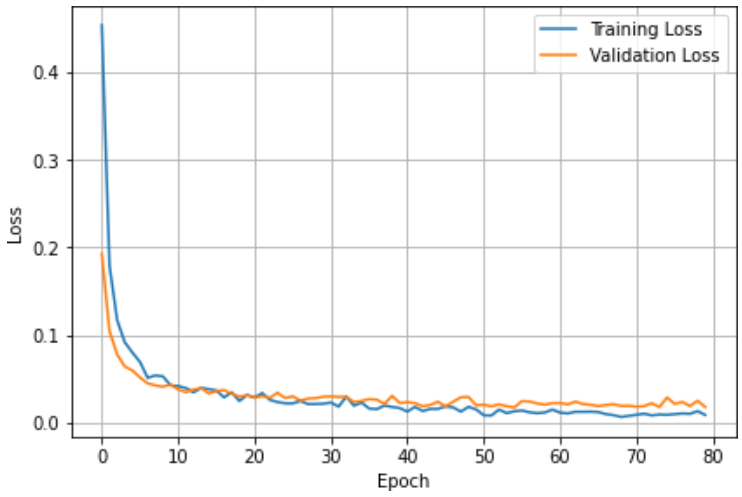
Kemudian langkah selanjutnya menampilkan plot *training* akurasi dan validasi akurasi seperti pada gambar 4.4.



Gambar 4.4 Grafik dari hasil *training* akurasi dan validasi akurasi pada *MobileNetV2*

Dari grafik pada gambar 4.4 dapat dilihat bahwa grafik dari hasil pelatihan dan validasi akurasi menunjukkan bahwa hampir mendekati angka 1.0 dengan bertambahnya *epochs* (lembar kerja). Kemudian pada *epoch* terakhir nilai akurasi pelatihan mencapai 0.9989 dan nilai validasi akurasi 0,9937. Grafik atau kurva dari akurasi pelatihan ditandai dengan warna biru dan grafik atau kurva validasi akurasi ditandai dengan warna oranye merupakan tanda nilai akurasi dari setiap *epoch*, dan jika nilainya semakin mendekati angka 1.0 selama bertambahnya *epoch* maka dapat disimpulkan bahwa proses pelatihan dan validasinya berhasil melakukan klasifikasi, dan jika nilai akurasi dari pelatihan maupun validasi tidakmeningkat atau bahkan menurun maka dapat disimpulkan bahwa jaringan CNN yang dilatih tidak berhasil melakukan klasifikasi dengan baik. Dari grafik atau kurva pada gambar 4.4 dapat dilihat bahwa dalam grafik atau kurva akurasi pelatihan hampir sama dengan peningkatan pada kurva atau grafik validasi akurasi. Hal tersebut menandakan bahwa hasil pelatihan berhasil.

Tahap selanjutnya menampilkan hasil dari pelatihan yang gagal dan validasi yang gagal seperti pada gambar 4.5.

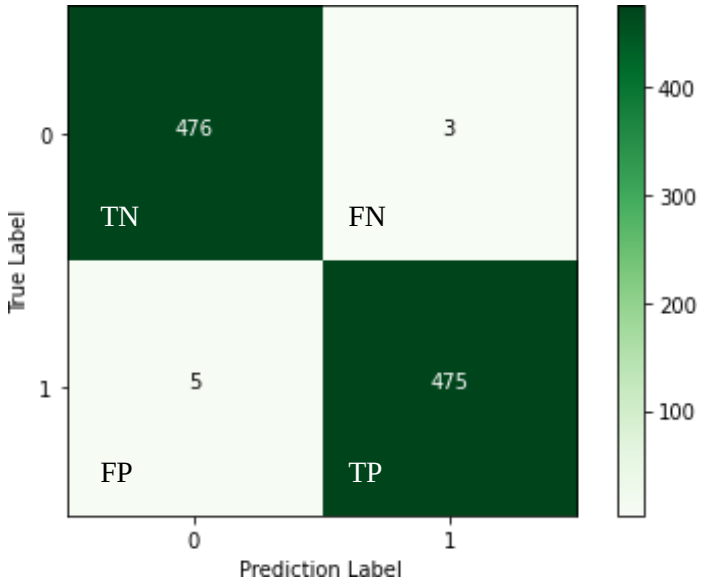


Gambar 4.5 Grafik atau kurva hasil pelatihan yang gagal dan validasi yang gagal pada *MobileNetV2*

Grafik atau kurva pada gambar 4.5 memperlihatkan hasil pelatihan yang gagal dan validasi yang gagal semakin menurun hampir mendekati 0.0 dengan bertambahnya *epoch* . Selanjutnya, dapat dilihat bahwa *epoch* terakhir nilai

pelatihan yang gagal mencapai 0.0054 dan nilai validasi akurasi yang gagal mencapai 0.0149. Sama seperti warna pada saat pelatihan dan validasi berhasil, pada grafik atau kurva pelatihan yang gagal ditandai dengan garis berwarna biru dan grafik atau kurva validasi yang gagal ditandai dengan garis berwarna oranye yang setiap garisnya dihasilkan dari setiap *epoch*. Dapat disimpulkan bahwa jika nilai hampir mendekati 0 menandakan bahwa jaringan CNN yang dilatih berhasil melakukan klasifikasi dengan baik, dan dalam grafik atau kurva pada pelatihan dan validasi yang gagal memiliki hasil penurunan yang hampir sama sehingga dapat disimpulkan bahwa hasil pelatihan berhasil.

Kemudian melakukan perintah untuk memeriksa matrik *model* dan juga evaluasi *data test* dilanjutkan dengan menampilkan matrik hasil prediksi dan juga *label* prediksi yang nantinya untuk menampilkan *confusion matrix* yang kemudian melakukan *plotting confusion matrix* dan diperolehlah gambar dari *confusion matrix* prediksi *label* yang telah dibuat seperti pada gambar 4.6.



Gambar 4.6 *Confusion matrix* prediksi label dan label benar

Confusion matrix pada gambar 4.6 menampilkan label benar dan juga label prediksi menggunakan angka 0 dan angka 1 dengan penjelasan bahwa angka 0 menandakan memakai masker dan angka 1 menandakan tidak memakai masker. Berdasarkan pada *confusion matrix* gambar 4.6 untuk nilai *True Positif*

mendapatkan hasil 475 menandakan bahwa terdapat 475 data yang memakai masker dan dari model yang dibuat memprediksi data tersebut memang memakai masker, *True Negative* mendapatkan hasil 476 menandakan ada 476 data yang tidak memakai masker dan dari model yang dibuat memprediksi data tersebut memang tidak memakai masker, *False Positif* mendapatkan hasil 5 menandakan ada 5 data yang tidak memakai masker tetapi dari model yang telah dibuat memprediksi bahwa data tersebut memakai masker dan *False Negative* mendapatkan hasil 3 menandakan terdapat 3 data yang memakai masker tetapi dari model yang dibuat memprediksi data tersebut tidak memakai masker. Untuk menghitunghasil dari pelatihan dalam pengujian dengan menggunakan cara sebagai berikut :

1. Akurasi = (TP+TN) / (TP+TN+FN+FP)
   = (475+476) / (475+476+3+5)
   = 951 / 959
   = 0,992
2. *Recall* = TP / (TP+FN)
   = 475 / (475+3)
   = 0,994
3. *Presisi* = TP / (TP+FP)
   = 475 / (475+5)
   = 0,990
4. *F1-Score*= 2 / ((1/presisi) + (1/*recall*))
   = 2 / (1,010+1,006)
   = 0,992

Tahap berikutnya yaitu membuat prediksi dari hasil pengujian dan untuk setiap gambar dalam satu set pengujian diperlukan penemuan indeks *label* dengan probabilitas prediksi terbesar dan juga menampilkan laporan klarifikasi yang dibuat menggunakan format yang baik dan untuk hasil dari pembuatan tabel dapatdilihat seperti dalam tabel 4.1.

Tabel 4.1 Tabel prediksi hasil pengujian

	Precision	*Recall*	*F1-Score*	*Support*
With Mask	0.99	0.99	0.99	479

	Precision	*Recall*	*F1-Score*	*Support*
Without Mask	0.99	0.99	0.99	480
Accuracy			0.99	959
Macro Avg	0.99	0.99	0.99	959
Weighted Avg	0.99	0.99	0.99	959

Tahap berikutnya membuat *directory* untuk menyimpan *model* dan mengubahformat *model* menjadi h5 dan menyimpannya kedalam *directory* yang selanjutnya *file* format tersebut akan digunakan pada *Software Spyder*.Proses selanjutnya membuat program pada *Spyder* yang nantinya digunakan untuk mendeteksi wajah.Langkah awal pembuatan program dengan meng*import* semua *library* yang akan digunakan yang kemudian melakukan pengambilan dimensi bingkai dan membuat*blob*. *Blob* sendiri merupakan *file* besar yang memerlukan proses lanjutan untuk deteksi kemudian jika *blob* sudah selesai selanjutnya membuat deteksi wajah danmenginisialisasi wajah , lokasi yang sesuai dan daftar prediksi dari masker wajah seperti pada gambar 4.7.

```
def detect_and_predict_mask(frame, faceNet, maskNet):
    # mengambil dimensi bingkai dan membuat blob
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),
        (104.0, 177.0, 123.0))

    # setelah blob jadi dan membuat deteksi wajah
    faceNet.setInput(blob)
    detections = faceNet.forward()
    print(detections.shape)

    #menginisialisasi daftar wajah, lokasi yang sesuai dan daftar prediksi
    #dari masker wajah
    faces = []
    locs = []
    preds = []
```

Gambar 4.7 Membuat deteksi wajah

Tahap selanjutnya melakukan pengulangan pendeteksian dan mengekstrak probabilitas yang terkait dengan deteksi, menyaring deteksi yang lemah dengan memastikan probabilitasnya lebih besar dibandingkan dengan probabilitas yang lebih rendah kemudian menghitung titik x dan y (garis *horizontal* dan garis *vertical*) yang digunakan sebagai koordinat kotak pembatas untuk objek dan memastikan kotak pembatas berada dalam dimensi bingkai, tahap selanjutnya mengekstrak

Region of Interest (ROI) wajah, *Region of Interest* (ROI) sendiri merupakan bagian dari suatu citra yang dipilih yang kemudian diproses, selanjutnya mengubah *dataset* yangteridentifikasi oleh sistem sebagai BGR menjadi RGB dan mengubah ukuran menjadi 224x224 dan memprosesnya terlebih dahulu dan menambahkan wajah dan kotak pembatas ke masing-masing daftar.

Langkah berikutnya yaitu membuat prediksi pada saat salah satu wajah berhasil terdeteksi dan untuk membuat kesimpulan yang lebih cepat, membuat prediksi *batch* pada semua wajah pada waktu yang sama kemudian kembali ke 2-tuple dari lokasi wajah. Kemudian memuat *model* deteksi wajah dari disk dan menginisialisasi *video stream* (*real-time*) dan mengambil bingkai pada saat *videostream* berjalan dan mengubah ukurannya dengan lebar maksimum 400pixel dan melakukan pendeteksian wajah dalam bingkai dan menggunakan masker atau tidak seperti pada gambar 4.8.

```
    # membuat prediksi jika setidaknya satu wajah terdeteksi
    if len(faces) > 0:
        # untuk kesimpulan yang lebih cepat, membuat prediksi batch di semua
        #wajah pada saat yang sama
        faces = np.array(faces, dtype="float32")
        preds = maskNet.predict(faces, batch_size=32)

    # kembali ke 2-tuple dari lokasi wajah dan lokasi yang sesuai
    return (locs, preds)

# memuat model detektor masker wajah dari disk
prototxtPath = r"D:\fix banget sempro\tugasakhir\deploy.prototxt"
weightsPath = r"D:\fix banget sempro\tugasakhir\res10_300x300_ssd_iter_140000.caffemodel"
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# memuat model detektor masker wajah dari disk
maskNet = load_model("mask_detector.model")

# inisialisasi real time (video stream)
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()

# melakukan perulangan pada video stream
while True:
    # mengambil bingkai pada saat video stream berjalan dan mengubah ukurannya
    # dengan lebar maksimum 400pixel
    frame = vs.read()
    frame = imutils.resize(frame, width=1000)

    # mendeteksi wajah dalam bingkai dan menentukan apakah menggunakan masker
    # atau tidak
    (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)
```

Gambar 4.8 Memuat prediksi sampai melakukan deteksi wajah dalam bingkai dan menentukan memakai masker atau tidak.

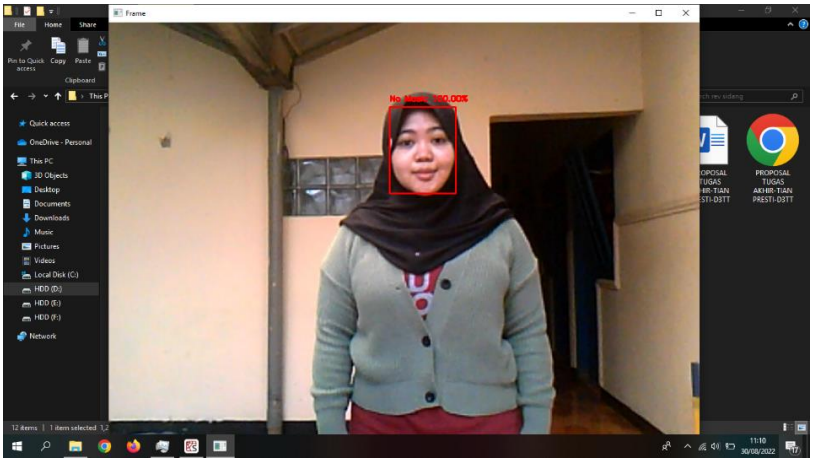
Tahap selanjutnya melakukan perulangan pada lokasi wajah yang terdeteksi dan lokasi yang sesuai, membongkar atau meng*unzip* kotak pembatas dan prediksi, kemudian melakukan penentuan *label* dan warna yang digunakan untuk menggambar kotak pembatas dan teks dan juga mensertakan probabilitas dalam *label*. Kemudian menampilkan *label* dan kotak pembatas pada *output* bingkai selanjutnya menampilkan bingkai dan mengatur tombol “C” untuk keluar dari pendeteksian seperti pada gambar 4.9.

```
    # melakukan perulangan pada lokasi wajah yang terdeteksi dan lokasi
    # yang sesuai
    for (box, pred) in zip(locs, preds):
        #membongkar(mengunzip) kotak pembatas dan prediksi
        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred

        #menentukan label dan warna yang akan digunakan untuk menggambar
        #kotak pembatas dan teks
        label = "Mask" if mask > withoutMask else "No Mask"
        color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

        # mensertakan probabilitas dalam label
        label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)

        # menampilkan label dan kotak pembatas pada output bingkai
        cv2.putText(frame, label, (startX, startY - 10),
            cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
        cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)

    # menampilkan bingkai
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    # jika menekan tombol 'C' keluar dari perulangan
    if key == ord("c"):
        break

# melakukan sedikit pembersihan
cv2.destroyAllWindows()
vs.stop()
```

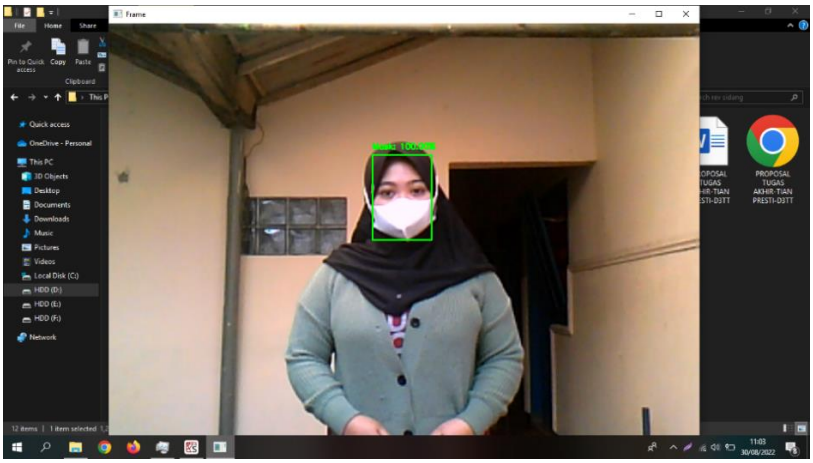
Gambar 4.9 Melakukan perulangan pada lokasi wajah yang terdeteksi hingga pengaturan tombol “C” untuk keluar dari pendeteksian

4.1.1 Hasil Pengujian Pendeteksian Masker

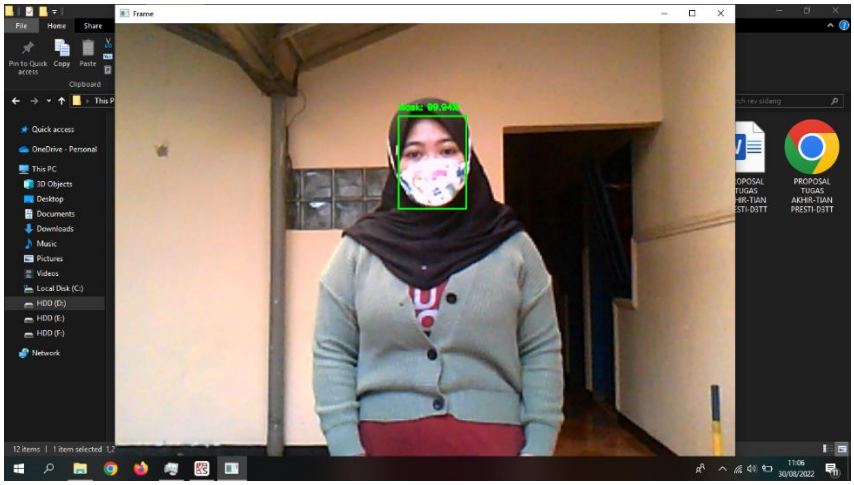
A. Hasil Pengujian Berbagai Motif Masker Jarak 1(satu) Meter



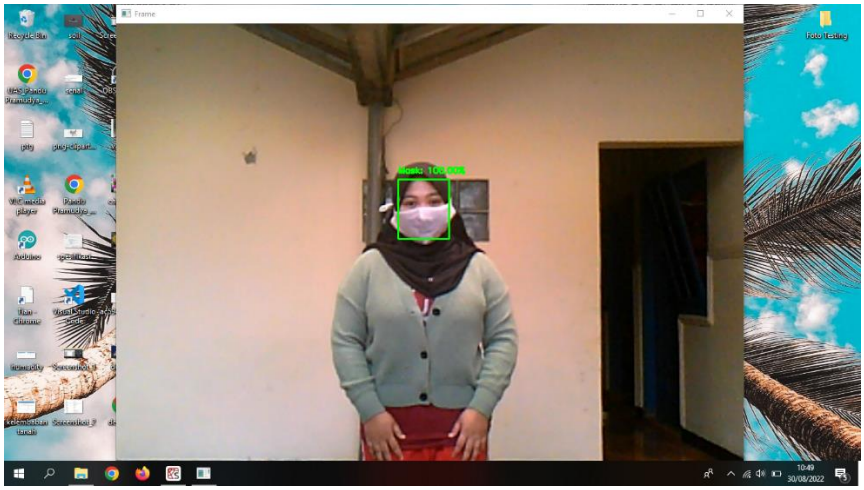
Gambar 4.10 Hasil Pengujian Tidak Memakai Masker Jarak 1(satu) Meter



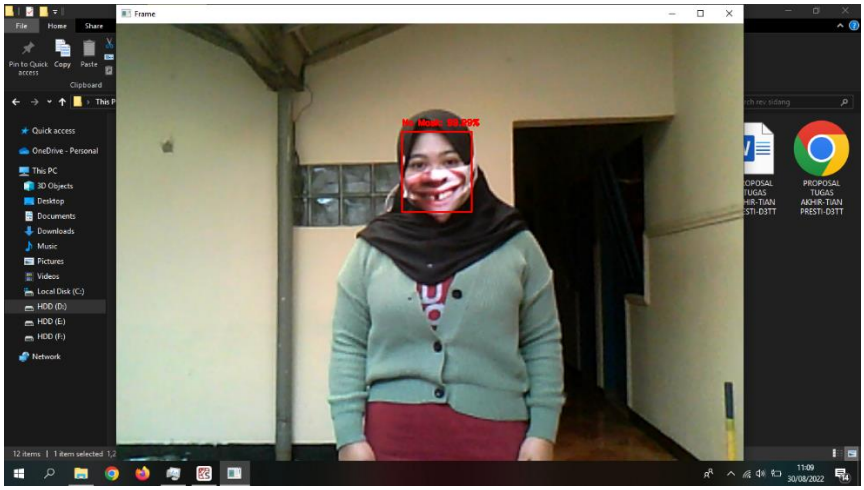
Gambar 4.11 Hasil Pengujian Memakai Masker Putih Polos 1(satu) Masker



Gambar 4.12 Hasil Pengujian Memakai Masker Rainbow 1(satu) Meter



Gambar 4.13 Hasil Pengujian Memakai Masker Polos Karakter 1(satu) Meter

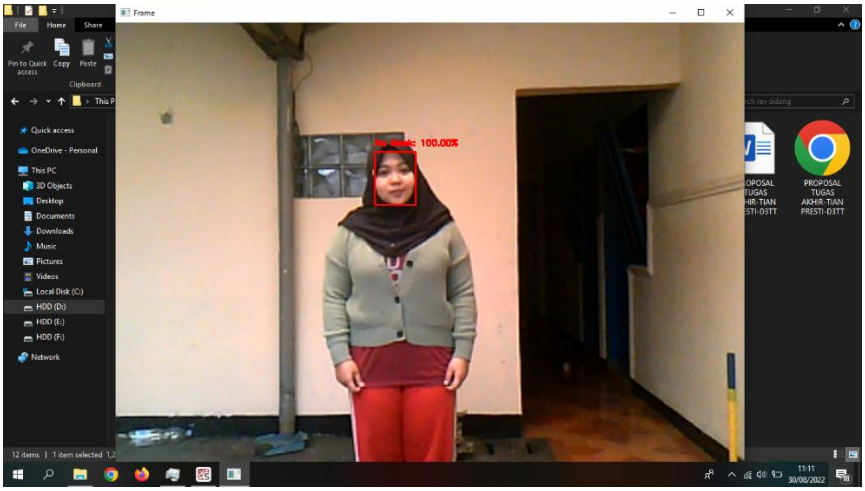


Gambar 4.14 Hasil Pengujian Memakai Masker Gigi Ompong 1(satu) Meter

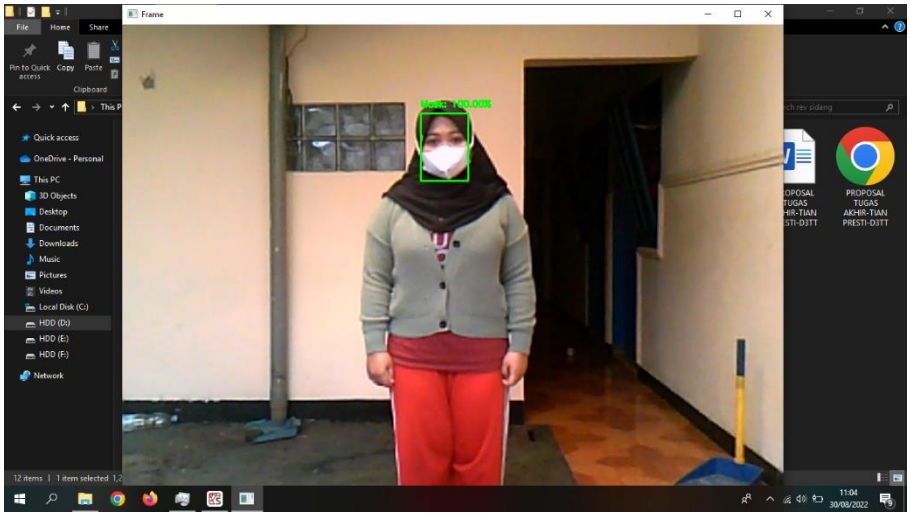
Dari 5(lima) kali pengujian dengan berbagai motif masker maupun

tanpa masker yang sudah dilakukan dapat disimpulkan bahwa dengan jarak pengujian dari 1(satu) meter dengan kondisi cahaya yang sama dengan menggunakan bantuan cahaya matahari, model yang telah dibuat dapat melakukan pendeteksian.Tetapi, pada saat pendeteksian memakai masker *rainbow*, polos karakter dan motif gigi ompong sistem yang telah dibuat mendapatkan hasil yang berubah-ubah dikarenakan sistem belum mempelajari model dengan penggunaan banyak motif ditambah kamera yang digunakan hanya kamera bawaan laptop pengujian yang memiliki spesifikasi yang kurang jernih.

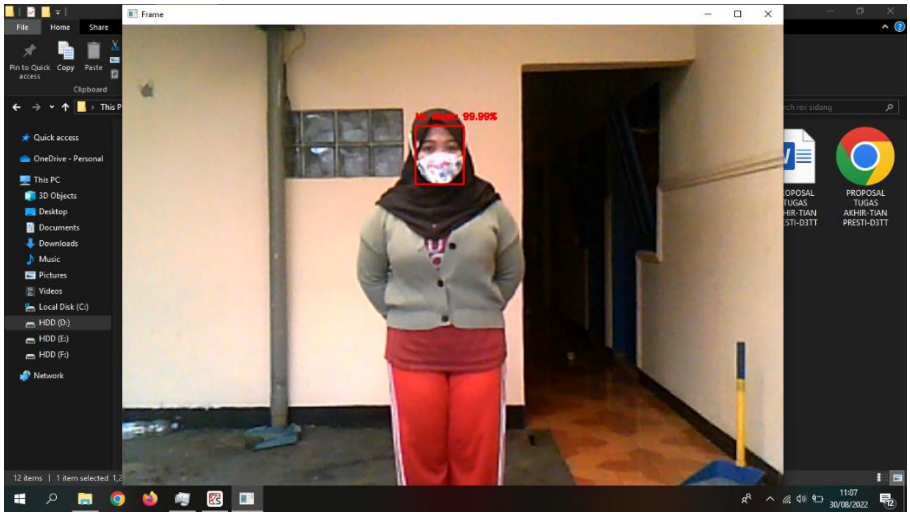
B. Hasil Pengujian Berbagai Motif Masker Jarak 1,5 Meter



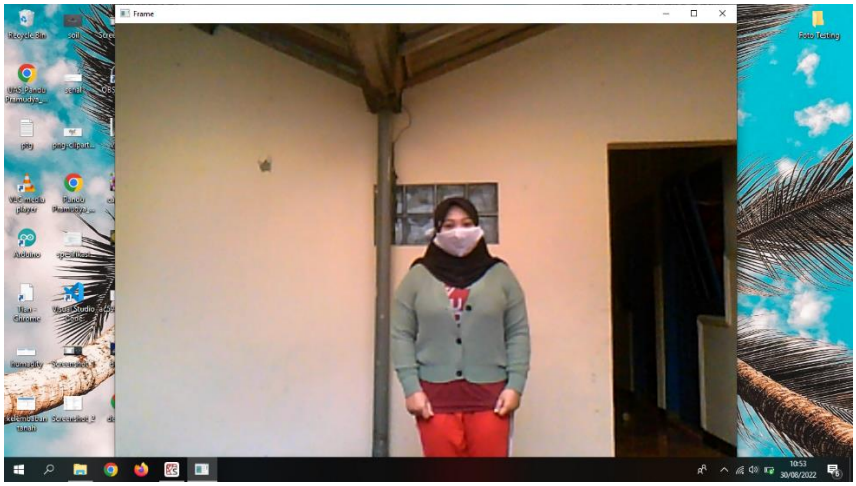
Gambar 4.15 Hasil Pengujian Tidak Memakai Masker Jarak 1,5 Meter



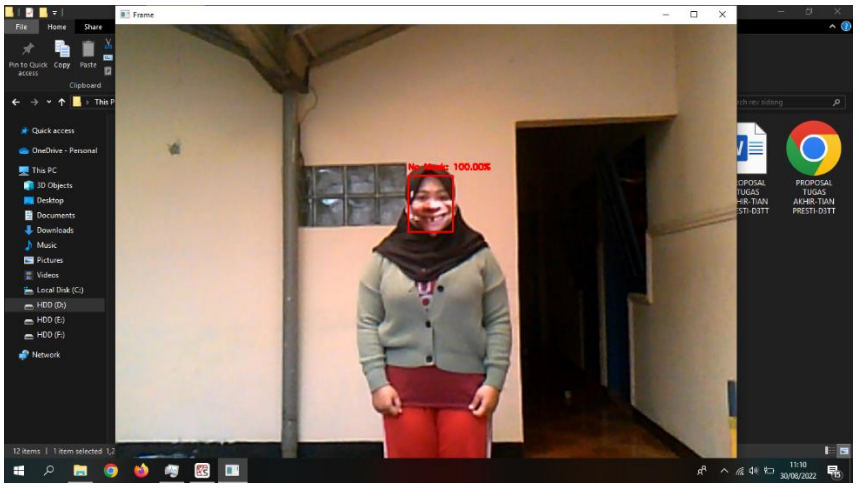
Gambar 4.16 Hasil Pengujian Memakai Masker Putih Polos Jarak 1,5 Meter



Gambar 4.17 Hasil Pengujian Memakai Masker Rainbow Jarak 1,5 Meter



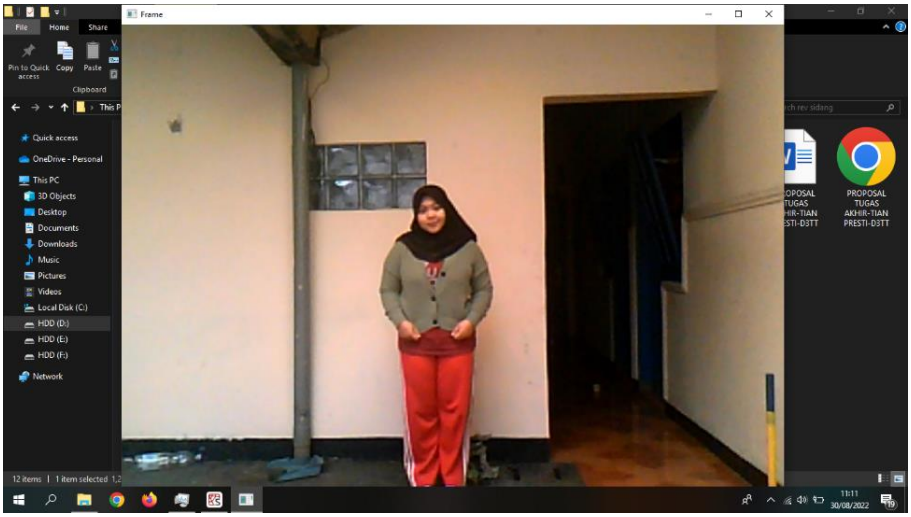
Gambar 4.18 Hasil Pengujian Memakai Masker Polos Karakter Jarak 1,5 Meter



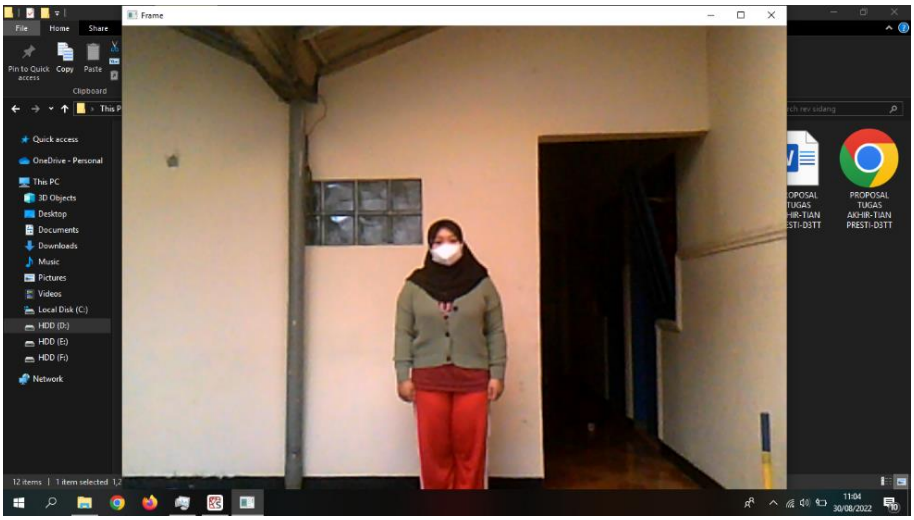
Gambar 4.19 Hasil Pengujian Memakai Masker Gigi Ompong Jarak 1,5 Meter

Dari beberapa pengujian yang telah dilakukan dengan jarak pengujian 1,5 Meter tidak memakai maupun memakai berbagai motif masker dapat disimpulkan bahwa sistem dapat mendeteksi wajah dalam kondisi tidak memakai masker maupun memakai masker dengan berbagai motif hanya saja pada saat pengujian masker polos karakter sistem tidak dapat mendeteksi wajah yang sedang dideteksi.

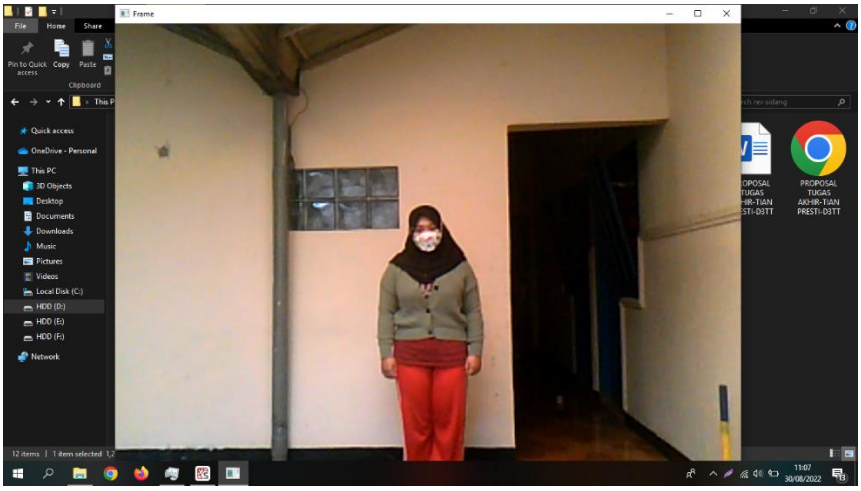
C. Hasil Pengujian Berbagai Motif Masker Jarak 2,5 Meter



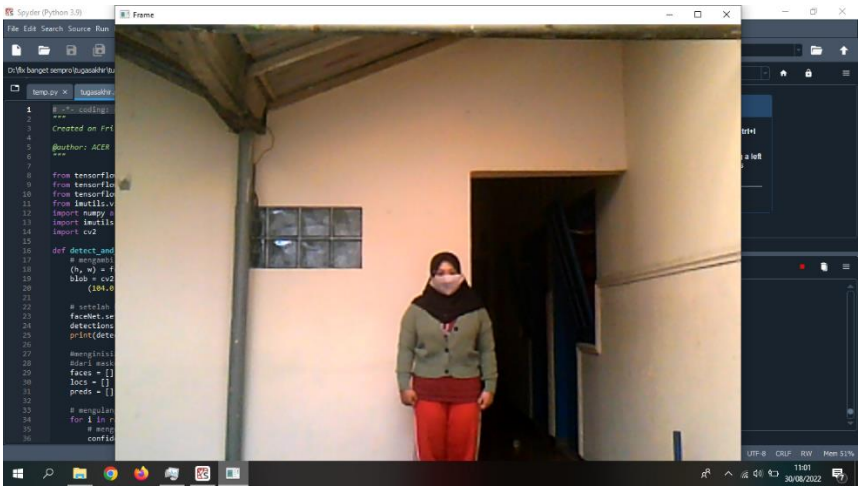
Gambar 4.20 Hasil Pengujian Tidak Memakai Masker Jarak 2,5 Meter



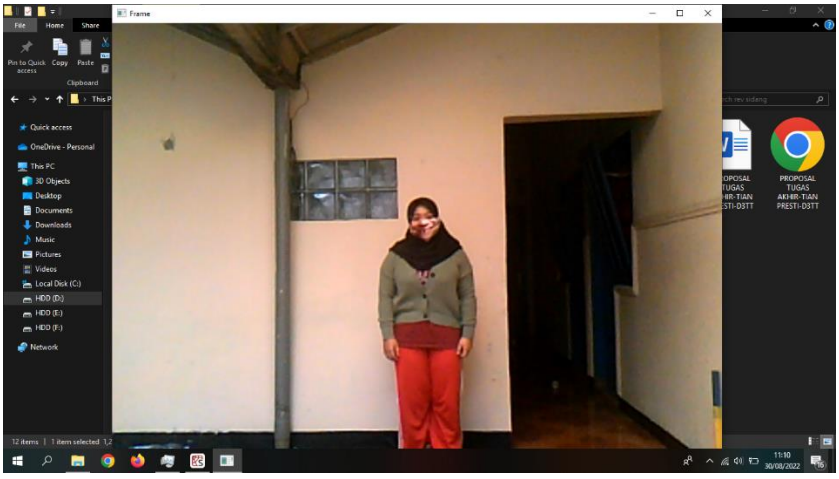
Gambar 4.21 Hasil Pengujian Memakai Masker Putih Polos Jarak 2,5 Meter



Gambar 4.22 Hasil Pengujian Masker Rainbow Jarak 2,5 Meter



Gambar 4.23 Hasil Pengujian Masker Polos Karakter Jarak 2,5 Meter



Gambar 4.24 Hasil Pengujian Memakai Masker Gigi Ompong Jarak 2,5 Meter

Hasil pengujian pendeteksian berbagai motif masker dengan jarak 2,5

meter dapat disimpulkan bahwa sistem belum bekerja dengan baik untuk melakukan pendeteksian dikarenakan jarak yang terlalu jauh dan juga spesifikasi kamera yang digunakan tidak bagus dan tidak jernih sehingga wajah yang sedang dideteksi tidak terlalu jelas dan sistem tidak bisa mengenali.

BAB 5

PENUTUP

5.1 KESIMPULAN

Dari beberapa hasil pengujian dan pembahasan yang telah dilakukan pada laporan penelitian Tugas Akhir pendeteksian masker wajah ini, dapat ditarik sebuah kesimpulan antara lain:

1. Pengimplementasian model arsitektur *mobilenetv2* dan *ssd resnet10* yang kemudian di *deploy* ke dalam *Software Spyder* untuk mendeteksi masker muka berhasil melakukan pendeteksian pada kondisi tidak memakai masker maupun memakai jenis masker berwarna polos dan masker motif wajah manusia dengan jarak yang direkomendasikan maksimal 1(satu) meter.
2. Kinerja model arsitekur *mobilenetv2* dan *ssd resnet10* untuk pendeteksian masker mendapatkan nilai akurasi dengan perolehan nilai 0.992, *presisi* bernilai 0.990, *recall* dengan perolehan nilai 0.994 dan *f1-score* dengan nilai 0.992.
3. Hasil pelatihan dan validasi akurasi setelah dilakukan sebanyak 80 *epoch* menunjukkan, bahwa pada *epoch* terakhir mendapatkan sebuah nilai akurasi pelatihan mencapai 0.9993 dan nilai validasi akurasi 0,9927. Dapat disimpulkan bahwa proses pelatihan dan validasinya berhasil melakukan klasifikasi, karena nilai akurasi dari pelatihan maupun validasi pada saat bertambahnya *epoch* semakin meningkat.

5.2 SARAN

Berdasarkan dari hasil penelitian yang sudah berhasil dilakukan, penelitian tentang pendeteksian masker ini masih dapat dikembangkan dan disempurnakan agar menjadi lebih baik antara lain :

1. Menambah *dataset* pelatihan yang digunakan pada penelitian ini agar hasil prediksi bisa semakin meningkat.
2. Proses pendeteksian masker wajah pada *Software Spyder* dapat dikembangkan lagi kedalam sebuah aplikasi,*website* maupun kedalam sebuah alat dengan bantuan *Rasberry Pi.*

3. Menambah jumlah *epoch* pada saat melakukan *training* model agar diharapkan mendapatkan hasil akurasi yang lebih tinggi.
4. Hasil prediksi pada saat menggunakan masker karakter mendapatkan hasil pendeteksian yang masih berubah-ubah, diharapkan dapat diperbaiki agar kedepannya sistem dapat memprediksi lebih baik.

DAFTAR PUSTAKA

[1] Seri Asnawati Munthe,Jasmen Manurung, Lia Rosa Veronika Sinaga, "Penyuluhan Dan Sosialisasi MaskerDi Desa Sifahandro Kecamatan Sawo Sebagai Bentuk Kepedulian TerhadapMasyarakat Ditengah Mewabahnya *Virus* Covid 19," *Jurnal Pengabdian Masyarakat Universitas Sari Mutiara* , vol. I, no. 2, pp. 115-123, 2020.

[2] Nany Hairunisa,Husnun Amalia,"Review: Penyakit *Virus* Corona Baru 2019 (Covid-19)," *Jurnal Biomedika Dan Kesehatan* , Vol. III, No. 2, Pp. 90-100, 2020.

[3] Arnaz Anggoro Saputro,Yudi Dwi Saputra,Guntum Budi Prasetyo, "Analisis Dampak *Covid-19* Terhadap Kesadaran Masyarakat Dalam Penerapan Protokol Kesehatan," *Journal Pendidikan Jasmani Kesehatan & Rekreasi (Porkes),* Vol. III, No.2, Pp. 81-92, 2020.

[4] Iqbal H. Sarker, "Machine *Learning*: Algorithms, *Real*-World Applications And Research Directions," *Sn Computer Science,* Vol. I, No. 7, Pp. 1-21, 2021.

[5] Endang Retnoningsih, Rully Pramudita, "Mengenal Machine *Learning* Dengan Teknik Supervised Dan Unsupervised *Learning* Menggunakan *Python*," Bina Insani Ict Journal, Vol. VII, No. 2, Pp. 156-165, 2020.

[6] Ivan Hartono, Agustinus Noertjahyana, Leo Willyanto Santoso, "Deteksi Masker Wajah Dengan Metode *Convolutional Neural Network*,"*Jurnal Infra* , Vol. X, No. 3, Pp. 203-209, 2022.

[7] Parmonangan R. Togatorop, Ahmad Fauzi, "Klasifikasi Penggunaan Masker WajahMenggunakan *Squeezenet*," *Jurnal Teknik Informatika Dan Sistem Informasi,* Vol. IX, No. 1, Pp. 379-406, 2022.

[8] Tri Septiana Nadia Puspita Putri,Mohamad Al Fikih, Novendra Setyawa, "*Face* Mask Detection *Covid-19* Using *Convolutional Neural Network* (Cnn)," *Seminar NasionalTeknologi Dan Rekayasa (Sentra)* , Vol. I, No. 5, Pp. 27-32, 2021.

[9] Yusuf Aleshinloye Abass, "Deep *Learning* Methodologies For Genomic *Data* Prediction: Review," *Journal Of Artificial Intelligence ForMedical Sciences,* Vol. X, No. 4, Pp. 1-11, 2021.

[10] M. Raihan Rafiiful Allaam, Agung Toto Wibowo, "Klasifikasi Genus Tanaman Anggrek MenggunakanMetode *Convolutional Neural Network* (Cnn)," *E-ProceedingOf Engineering,* Vol. VIII, No.2, Pp. 1153-1189, 2021.

[11] Preeti Nagrath, Rachna Jain, Agam Madan, Rohan Arora, Piyush Kataria, Jude Hemanth, "SSDMNV2: A *Real Time* DNN- *Based Face Mask Detection System Using Single Shot Multibox Detector* And *Mobilenetv2*," *Elsevier Public Health Emergency Collection* , Vol. I, No. 5,Pp. 1-11, 2020.

[12] Houssam Benbrahim, Hana Hachimi, Aouatif Amine, "Deep *Convolutional Neural Network* With Tensorflow And Keras To *Classify* Skin Cancer *Image*s," *Scalable Computing: Practice And Experience,* Vol. XXI, No. 3, Pp. 379-389, 2020.

[13] Daniel Febrian Sengkey, Feisy Diane Kambey, Salvius Lengkong, Salaki Reynaldo Joshua, Heny Valentino Florensius Kainde, "Pemanfaatan Platform Pemrograman Daring Dalam Pembelajaran Probabilitas Dan Statistika Di Masa Pandemi *Covid-19*," *Jurnal Informatika* , Vol. XV, No. 4,Pp. 257-264, 2020.

[14] Yudha Taufik, "*OPENCV* :: Penerapan Deteksi Wajah Menggunakan *Opencv* [Online] [Access : 22 April 2022]

[15] Muhammad Afif Amanullah Fawwaz, Kurniawan Nur Ramadhani, S.T.,M.T. Febryanti Sthevanie, S.T.,M.T. "Klasifikasi Raspada Kucing menggunakan Algoritma *Convolutional Neural Network* (CNN)," Jurnal Tugas Akhir Fakultas Informatika, Vol. 8, no. 1, pp.715-730, 2021.

[16] M. Aminullah, "Alat Deteksi Masker Dengan Metode *Convolutional Neural Network* Untuk Tunanetra Pada Era New Normal," Seminar Nasional Fortei Regional 7, vol. III, no. 5, pp. 321-324, 2021.

[17] Nyoman Purnama, Putu Kusuma Negara, "Deteksi Masker Pencegahan *Covid19* Menggunakan *Convolutional Neural Network* Berbasis Android," JURNALRESTI , vol. I, no. 1, 2017 sd vol.V, no. 3, pp. 576-583, 2021.

[18] Widi Hastomo, Sugiyanto, Sudjiran, "CONVOLUTION NEURAL NETWORKARSITEKTUR MOBILENET-V2 UNTUK MENDETEKSI TUMOR OTAK," Seminar Nasional Teknologi Informasi dan Komunikasi STI&K (SeNTIK), vol. 5, no. 1, pp. 17-21, 2021.

[19] Muhammad Aminullah, "Alat Deteksi Masker Dengan Metode *Convolutional Neural Network* Untuk Tunanetra Pada Era New Normal," Seminar Nasional Fortei Regional 7 (SinarFe 7), vol. IV, no.4,pp. 321-324, 2021.

[20] Nur Hadianto, Hafifah Bella Novitasari, Ami Rahmawati, "Klasifikasi Peminjakan Nasabah Bank Menggunakan Metode Neural Network", Jurnal PILAR Nusa Mandiri, Vol. VX, No.2, pp.163-170, 2019

LAMPIRAN

1. Lampiran Program Pembuatan *Model* pada *Google Colaboratory*

```
# Melakukan cloning data atau mendownload data dari github ke folder
      atau penyimpanan googlecollab
  !git clone https://github.com/tianpresti6/Face-Mask-
  Detection-Using-CNN
  #Berpindah ke folder face-mask-detection
  %cd Face-Mask-Detection-Using-CNN/
  #Kemudian memeriksa isi folder face-mask-detection apa saja
  !ls
  #Melakukan pengimport-an semua library yang akan digunakan
  import
  tensorflow as tf
  from tensorflow.keras.preprocessing.image import
  ImageDataGeneratorfrom tensorflow.keras.applications
  import MobileNetV2
  from tensorflow.keras.layers import
  AveragePooling2Dfrom
  tensorflow.keras.layers import Dropout
  from tensorflow.keras.layers
  import Flatten from
  tensorflow.keras.layers import
  Dense from
  tensorflow.keras.layers import
  Input from
  tensorflow.keras.models import
```

```
Model from tensorflow.keras.optimizers
    import Adam
    from tensorflow.keras.applications.mobilenet_v2
    import
    preprocess_input
    from tensorflow.keras.preprocessing.image import
    img_to_arrayfrom
    tensorflow.keras.preprocessing.image import
    load_img from tensorflow.keras.utils import
    to_categorical
from sklearn.preprocessing import LabelBinarizer
    from sklearn.model_selection import
    train_test_splitfrom sklearn.metrics import
    classification_report from sklearn.metrics import
    confusion_matrix
from imutils import paths
    import matplotlib.pyplot as
    pltimport pathlib
import numpy as np
    import argparse
    import os
```

```
import itertools

# Proses penginstallan ipython-autotime untuk menghitung waktu lamanya eksekusi tiap sel atau proses di Google Colaboratory

    !pip install ipython-autotime

    %load_ext autotime

# Memeriksa Versi TensorFlow yang digunakanprint(tf.

    version )

# Inisialisasi nilai Initial Learning Rate, berapa banyak Epoch pelatihan,dan Batch Size

    INIT_LR = 1e-4

    EPOCHS = 30

    BS = 32

# Mengambil gambar dari folder dataset , kemudian melakukan inisialisasi
    data dan label(bermasker dan tidak bermasker)

    print("Menginput gambar...")

    imagePaths = list(paths.list_images('dataset'))data =

    []

    labels = []
```

```
# Melakukan perulangan pada image paths
    for imagePath in imagePaths:
# Mengekstrak class label (bermasker dan tidak bermasker) dari filename
            label = imagePath.split(os.path.sep)[-2]
# Memuat input gambar (224x224) dan melakukan proses mengubah gambar menjadi array
            image = load_img(imagePath, target_size=(224, 224))
            image = img_to_array(image)
            image = preprocess_input(image)
# Mengupdate data dan labels lists secara berurutan
            data.append(image)
            labels.append(label)
# Mengkonversi data dan label ke dalam NumPy Arrays data
        = np.array(data, dtype="float32")
        labels = np.array(labels)
# Melakukan one-hot encoding pada label // mengubah kategori ke numerik
        lb = LabelBinarizer()
        labels = lb.fit_transform(labels)
        labels = to_categorical(labels)
        print("Input gambar berhasil")
# Membagi data ke dalam pelatihan dan pengujian dengan perbandingan (75% : 25%)
        (trainX, testX, trainY, testY) = train_test_split(data, labels,
            test_size=0.25, stratify=labels, random_state=42)
```

```
#Membuat manipulasi gambar untuk mengubah rotasinya,ukuran,tinggidsb
        aug = ImageDataGenerator(
            rotation_range=20,
            zoom_range=0.15,
            width_shift_range=0.2,
            height_shift_range=0.2,
            shear_range=0.15,
            horizontal_flip=True,
            fill_mode="nearest")
# mengimport Arsitektur jaringan MobileNetV2
        baseModel=MobileNetV2(weights="imagenet",
    include_top=False,
            input_tensor=Input(shape=(224, 224, 3)))
#untuk melihat informasi apa saja yang ada pada model yang akan dibuat
        baseModel.trainable = False
        baseModel.summary()
# Membentuk bagian head dari model yang nantinya akan ditempatkan
pada base model || head model (var) output dan base model input
        headModel = baseModel.output
        headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
        headModel = Flatten(name="flatten")(headModel)
        headModel = Dense(128, activation="relu")(headModel)
        headModel = Dropout(0.5)(headModel)
        headModel = Dense(2, activation="softmax")(headModel)
```

```
# Menempatkan head model pada base model
        model = Model(inputs=baseModel.input, outputs=headModel)#
Perulangan pada seluruh base model
        for layer in baseModel.layers:
            layer.trainable = False
# Persiapan menyusun model
        print("Mengkompilasi model...")
        opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
        model.compile(loss="binary_crossentropy", optimizer=opt,
            metrics=["accuracy"])
        model.summary()
#melakukan Pelatihan model
        print("Training head model...")H
        = model.fit(
            aug.flow(trainX, trainY, batch_size=BS),
            steps_per_epoch=len(trainX) // BS,
            validation_data=(testX, testY),
            validation_steps=len(testX) // BS,
            epochs=EPOCHS)
```

```
# menampilkan plot training loss dan validasi loss
        N = EPOCHS
        fig = plt.figure(figsize=(7, 4))
        fig.set_figheight(10)
        fig.set_figwidth(15)
        plt.subplot(2, 2, 2)
        plt.plot(np.arange(0, N), H.history["loss"],label = "Training Loss")
        plt.plot(np.arange(0,  N),  H.history["val_loss"],label  =  "Validation
    Loss")
        plt.legend()
        plt.xlabel("Epoch")
        plt.ylabel("Loss")
        plt.grid(zorder = 0)
        plt.show()
        # Memeriksa matriks model
        print(model.metrics_names)
        # Evaluasi data test
        print(model.evaluate(x= testX, y = testY))
```

```
# Menampilkan matriks yang benar dan matriks hasil prediksi
# Label yang benar
        yTrue = np.argmax(testY, axis=1)
# Label prediksi
        YPred = model.predict(testX, batch_size=BS)
        yPred = np.argmax(YPred, axis=1)
        print(yTrue)
        print(yPred)
#fungsi menampikan confusion matrix
        def get_confusion_matrix(yTrue, yPred):
            n_classes = len(np.unique(yTrue))
            conf = np.zeros((n_classes, n_classes))
            for actual, pred in zip(yTrue, yPred):
                conf[int(actual)][int(pred)] += 1
            return conf.astype('int')
        #inisialisasi nilai conf sebagai confusion matrix dan menampilkan
    jumlah array didalamnya
        conf = get_confusion_matrix(yTrue, yPred)
        conf
```

```
# Plotting confusion matrix
        plt.imshow(conf, interpolation='nearest', cmap=plt.cm.Greens)
# plt.title("Confusion Matrix")
        plt.colorbar()
        tick_marks = np.arange(len(classes))
        plt.xticks(tick_marks, classes)
        plt.yticks(tick_marks, classes)
        fmt = 'd'
        thresh = conf.max() / 2.
        for i, j in itertools.product(range(conf.shape[0]),
    range(conf.shape[1])):
            plt.text(j, i, format(conf[i, j], fmt),
                horizontalalignment="center",
                color="white" if conf[i, j] > thresh else "black")
        plt.tight_layout()
        plt.ylabel('True Label')
        plt.xlabel('Prediction Label')
# Berdasarkan confusion matrix
        TP = true_pos = 467
        TN = true_neg = 478
        FP = false_pos = 13
        FN = false_neg = 1
        results = {}
```

```
# Akurasi
        metric = "Akurasi"
        results[metric] = (TP + TN) / (TP + TN + FP + FN)
        print(f"{metric} = {results[metric]: .3f}")
# Recall
        metric  = "Recall"
        results[metric] = TP / (TP + FN)
        print(f"{metric} = {results[metric]: .3f}")
# Presisi
        metric = "Presisi"
        results[metric] = TP / (TP + FP)
        print(f"{metric} = {results[metric]: .3f}")
# Nilai F1
        metric = "F1"
        results[metric] = 2 / (1 / results["Presisi"] + 1 / results["Recall"])
        print(f"{metric} = {results[metric]: .3f}")
# Membuat prediksi dari pengujian
        predIdxs = model.predict(testX, batch_size=BS)
# Untuk setiap gambar dalam set pengujian, kita perlu menemukan indeks
label dengan probabilitas prediksi terbesar
        predIdxs = np.argmax(predIdxs, axis=1)
# Menampilkan laporan klasifikasi yang diformat dengan baik
        print(classification_report(testY.argmax(axis=1), predIdxs,
            target_names=lb.classes_))
```

```
#membuat directory untuk menyimpan model
        export_dir='saved_models/1'
        tf.saved_model.save(model, export_dir)
#menyimpan model dengan format h5
        model.save("/content/simpan_model/model_work.h5")
# menyimpan model h5 kedalam directory
        print("[INFO] saving mask detector model...")
        model.save("/content/simpan_model/mask_detector.model",
    save_format="h5")
```

2. Lampiran Program *Model* di *Software Spyder*

```
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import imutils
import cv2
def detect_and_predict_mask(frame, faceNet, maskNet):#
mengambil dimensi bingkai dan membuat blob
(h, w) = frame.shape[:2]
blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),(104.0, 177.0, 123.0))
# setelah blob jadi dan membuat deteksi wajah
                faceNet.setInput(blob) detections
                = faceNet.forward()
                print(detections.shape)
#menginisialisasi daftar wajah, lokasi yang sesuai dan daftar prediksi dari masker wajah
                faces = []
                locs = []
                preds = []
# mengulang pendeteksian
                for i in range(0, detections.shape[2]):
# mengekstrak probabilitas yang terkait dengan deteksi
                confidence = detections[0, 0, i, 2]
```

```
# mengulang pendeteksian
                for i in range(0, detections.shape[2]):
# mengekstrak probabilitas yang terkait dengan deteksi
                        confidence = detections[0, 0, i, 2]
#menyaring deteksi yang lemah dengan memastikan probabilitasnya lebih besar dibandingkan dengan probabilitas yang rendah
                        if confidence > 0.5:
#menghitung (x,y) untuk koordinat kotak pembatas untuk objek
                        box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
                        (startX, startY, endX, endY) = box.astype("int")
#memastikan kotak pembatas berada dalam dimensi bingkai
                        (startX, startY) = (max(0, startX), max(0, startY))
                        (endX, endY) = (min(w - 1, endX), min(h - 1, endY))
#mengekstrak ROI wajah, mengubah dari BGR menjadi RGB mengubah ukuran menjadi 224x224 dan memprosesnya terlebih dahulu c
                        face = frame[startY:endY, startX:endX]
                        face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
                        face = cv2.resize(face, (224, 224))
                        face = img_to_array(face)
                        face = preprocess_input(face)
# tambahkan wajah dan kotak pembatas ke masing-masing daftar
                        faces.append(face)
                        locs.append((startX, startY, endX, endY))
# membuat prediksi jika setidaknya satu wajah terdeteksi
                        if len(faces) > 0:
```

#untuk kesimpulan yang lebih cepat, membuat prediksi *batch* di semua wajah pada saat yang sama

*face*s = np.*array*(*face*s, dtype="float32")

preds = maskNet.predict(*face*s, *batch_size*=32)

#kembali ke 2-tuple dari lokasi wajah dan lokasi yang sesuai

return (locs, preds)

#memuat *model* detektor masker wajah dari disk

prototxtPath=r"D:\fixbangetsempro\tugasakhir\deploy.prototxt"

weightsPath=r"D:\fixbangetsempro\tugasakhir\res10_300x300_ssd_iter_140000.caffe*model*"

*face*Net = cv2.dnn.readNet(prototxtPath, weightsPath)

#memuat *model* detektor masker wajah dari disk

maskNet = load_*model*("mask_detector.*model*")

#inisialisasi *real time* (*video stream*)

print("[INFO] starting *video stream*...")

vs = *VideoStream*(src=0).start()

#melakukan perulangan pada *video stream*

while True:

#mengambil bingkai pada saat *video stream* berjalan dan mengubah ukurannya dengan lebar maksimum 400pixel

frame = vs.read()

frame = imutils.res*ize*(frame, width=1000)

#mendeteksi wajah dalam bingkai dan menentukan apakah menggunakan masker atau tidak

(locs, preds) = detect_and_predict_mask(frame, *face*Net, maskNet)

```
#melakukan perulangan pada lokasi wajah yang terdeteksi dan lokasi yang sesuai
        for (box, pred) in zip(locs, preds):
#membongkar(mengunzip) kotak pembatas dan prediksi
        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred
#menentukan label dan warna yang akan digunakan untuk menggambar kotak pembatas dan teks
        label = "Mask" if mask > withoutMask else "No Mask"
        color = (0, 255, 0) if label == "Mask" else (0, 0, 255)
#mensertakan probabilitas dalam label
        label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)
#menampilkan label dan kotak pembatas pada output bingkai
        cv2.putText(frame, label, (startX, startY - 10),
        cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
        cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
#menampilkan bingkai
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF
#jika menekan tombol 'C' keluar dari perulangan
        if key == ord("c"):
                break
# melakukan sedikit pembersihan
        cv2.destroyAllWindows()
        vs.stop()
```