PENDETEKSI PENGGUNA MASKER PADA MEDIA VIDEO CCTV MENGGUNAKAN METODE DEEP LEARNING

(STUDI KASUS: PUSKESMAS SUDIANG RAYA)

SKRIPSI

Diajukan sebagai salah satu syarat untuk menyelesaikan Pendidikan Diploma Empat (D-4) Program Studi Teknik Komputer dan Jaringan Jurusan Teknik Elektro Politeknik Negeri Ujung Pandang

AINUN TRISNANINGRUN
425 18 028

PROGRAM STUDI D-4 TEKNIK KOMPUTER & JARINGAN

JURUSAN TEKNIK ELEKTRO

POLITEKNIK NEGERI UJUNG PANDANG

2022

HALAMAN PENGESAHAN

Skripsi dengan Judul **Pendeteksi Penggunaan Masker Pada Media Video CCTV Menggunakan Metode Deep Learning (Studi Kasus: Puskesmas Sudiang Raya)** oleh **Ainun Trisnaningrun (425 18 028)** dinyatakan layak untuk diujikan.

Makassar, September 2022

Mengesahkan,

Pembimbing I,	Pembimbing II,
Eddy Tungadi, S.T., M.T.	**Irfan Syamsuddin, S.T. M.Com.ISM., Ph.d**
NIP. 197908232010121001	**NIP. 197312202000031008**

Mengetahui,

Ketua Program Studi
Teknik Komputer dan Jaringan
Politeknik Negeri Ujung Pandang

Eddy Tungadi, S.T., M.T.
NIP. 197908232010121001

HALAMAN PENERIMAAN

Pada hari ini September 2022, Tim Penguji Sidang Tugas Akhir, telah menerima dengan baik hasil skripsi oleh mahasiswa **Ainun Trisnaningrun (42518028) dengan judul PENDETEKSI PENGGUNA MASKER PADA MEDIA VIDEO CCTV MENGGUNAKAN METODE DEEP LEARNING (STUDI KASUS: PUSKESMAS SUDIANG RAYA).**

Makassar, September 2022

Tim Penguji Ujian Skripsi :

1. Rini Nur, S.T., M.T.	Ketua	(.....)
2. Zawiyah Saharuna, S.T., M.Eng.	Sekretaris	(.....)
3. Sahbuddin Abdul Kadir, ST., MT.	Anggota	(.....)
4. Muhammad Nur Yasir Utomo, S.ST., M.Eng.	Anggota	(.....)
5. Eddy Tungadi, S.T., M.T.	Pembimbing I	(.....)
6. Irfan Syamsuddin, S.T., M.Com.ISM., Ph.D.	Pembimbing II	(.....)

KATA PENGANTAR

Puji syukur senantiasa penulis panjatkan ke hadirat Allah SWT atas Berkat dan Rahmat-Nya yang telah memberikan kesehatan dan keselamatan kepada penulis sehingga penulis dapat menyelesaikan skripsi ini dengan baik. Shalawat dan salam kepada baginda Rasul Muhammad SAW sebagai sebaik-baik panutan bagi seluruh manusia.

Skripsi ini disusun guna memenuhi salah satu syarat untuk menyelesaikan studi serta dalam rangka memperoleh gelar Diploma IV (D-4/S1 Terapan) pada Program Studi Teknik Komputer dan Jaringan Jurusan Teknik Elektro Politeknik Negeri Ujung Pandang.

Penulis menyadari bahwa keberhasilan penyusunan skripsi ini tidak lepas dari bantuan berbagai pihak baik secara langsung maupun tidak langsung. Oleh karena itu, dengan rendah hati penulis mengucapkan terima kasih kepada:

1. Orang tua penulis yakni Almarhum Ayah saya Abdul Ganing dan Ibu saya Darnawati S.T., M.T. serta kakak-kakak dan adik saya Muh. Ichsan, Muh. Nurul Ishaq, Salfia Yunus, dan Nur Aisyah Irmadani yang sampai saat ini selalu memberikan semangat, motivasi, dukungan, bimbingan dan doa kepada penulis.
2. Bapak Prof. Ir. Muhammad Anshar, M. Si., Ph.D. selaku Direktur Politeknik Negeri Ujung Pandang.
3. Bapak Ahmad Rizal Sultan, S.T., M.T., Ph.D. selaku Ketua Jurusan Teknik Elektro Politeknik Negeri Ujung Pandang.
4. Bapak Eddy Tungadi, S.T., M.T. selaku Koordinator Program Studi Teknik

Komputer dan Jaringan.

5. Bapak Eddy Tungadi, S.T., M.T. selaku pembimbing I dan Bapak Irfan Syamsuddin, S.T. M.Com.ISM., Ph.D. selaku pembimbing II. Terima kasih atas segala ilmu, motivasi, nasehat, arahan, bantuan dan kesediaan waktu dan kesabarannya dalam membimbing penulis hingga terselesaikannya penelitian ini.
6. Seluruh dosen dan Staf Jurusan Teknik Elektro, Khususnya Program Studi D4 Teknik Komputer dan Jaringan.
7. Teman-teman seperjuangan di Program Studi TKJ angkatan 2018 dan khususnya teman-teman TKJ B angkatan 2018 yang telah berjuang bersama selama 4 tahun dan mengajarkan banyak hal kepada penulis baik dari segi akademik maupun non akademik.
8. Terkhusus dan terspesial kepada Muh. Alief Hanafie. Terima kasih karena selalu memberi semangat yang tulus dan menemani selama pengerjaan skripsi dari awal hingga akhir.
9. Semua pihak yang telah memberikan bantuan moril maupun materil yang tidak dapat disebutkan satu per satu.

Penulis menyadari bahwa skripsi ini masih banyak kekurangan, sehingga penulis mengharapkan kritik dan saran yang sifatnya membangun untuk perbaikan di masa mendatang. Semoga tulisan ini bermanfaat.

Makassar, September 2022

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

SURAT PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Ainun Trisnaningrun
NIM : 425 18 028

Menyatakan dengan sebenar-benarnya bahwa segala pernyataan dalam skripsi ini yang berjudul **"Pendeteksi Pengguna Masker Pada Media Video Cctv Menggunakan Metode Deep Learning (Studi Kasus: Puskesmas Sudiang Raya"** merupakan gagasan dan hasil karya sendiri dengan arahan komisi pembimbing, dan belum pernah diajukan dalam bentuk apapun pada perguruan tinggi dan instansi mana pun.

Semua data dan informasi yang digunakan telah dinyatakan secara jelas dan dapat diperiksa kebenarannya. Sumber informasi yang berasal atau dikutip dari karya yang diterbitkan dari penulis lain telah disebutkan dalam naskah dan dicantumkan dalam skripsi ini.

Jika pernyataan saya tersebut di atas tidak benar, saya siap menanggung resiko yang ditetapkan oleh Politeknik Negeri Ujung Pandang.

Makassar, 20 September 2022

Ainun Trisnaningrun
NIM. 42518028

PENDETEKSI PENGGUNA MASKER PADA MEDIA VIDEO CCTV MENGGUNAKAN METODE DEEP LEARNING (STUDI KASUS: PUSKESMAS SUDIANG RAYA)

RINGKASAN

Penggunaan masker adalah salah satu hal yang perlu diperhatikan ketika ingin keluar rumah untuk menerapkan protokol kesehatan guna menghindari penyakit yang saat ini meresahkan masyarakat di dunia atau biasa dikenal dengan Covid-19. Juru Bicara Covid-19 Achmad Yurianto mengatakan, salah satu faktor utama penyumbang kasus positif Covid-19 disebabkan ketidakdisiplinan masyarakat menggunakan masker ketika berada di tempat keramaian (Rokom, 2021). Saat ini orang – orang enggan datang ke rumah sakit dikarenakan takut jika dituduh terkena virus Covid-19 (Fahlevi, 2021), sehingga orang – orang yang memerlukan pengobatan lebih memilih mengunjungi puskesmas yang berada di sekitar rumah mereka. Namun, masih banyak orang yang tidak menggunakan masker dengan alasan lokasi yang dituju dekat dari rumah (Demak Kominfo, 2020). Untuk mengatasi hal tersebut bisa dilakukan dengan cara mendeteksi wajah pengunjung menggunakan kamera. Sehingga diusulkan sebuah sistem yaitu pendeteksi pengguna masker dengan metode *Convolutional Neural Network* (CNN). Salah satu metode CNN yang banyak diaplikasikan untuk mengolah data citra adalah YOLO. YOLO (*You Only Look Once*) adalah salah satu model berbasis *deep learning* yang dikembangkan untuk mendeteksi sebuah objek secara real-time (Rachmawati & Widhyaestoeti, 2020). Cara kerja YOLO dengan melihat citra secara keseluruhan, kemudian melewati *neural network* dan otomatis mendeteksi objek yang ada. Sehingga pada penelitian ini digunakan model YOLO yaitu YOLOv4 sebagai model pendeteksi objek pada sistem pendeteksi masker dengan media video CCTV yang datanya dikirim secara *real-time*.

Keywords: Pendeteksi Masker, CCTV, *Deep Learning, Convolutional Neural Network,* YOLO.

BAB I PENDAHULUAN

1.1 Latar Belakang

Penggunaan masker adalah salah satu hal yang perlu diperhatikan ketika ingin keluar rumah untuk menerapkan protokol kesehatan guna menghindari penyakit yang saat ini meresahkan masyarakat di dunia atau biasa dikenal dengan Covid-19. Varian baru Covid-19 yang telah masuk di Indonesia memiliki risiko penularan lebih cepat dibandingkan varian-varian sebelumnya, sehingga masyarakat harus lebih memperhatikan protokol kesehatan seperti mencuci tangan, menjaga jarak, menggunakan masker, menjauhi kerumunan, dan melakukan vaksinasi (Ninggar, 2021). Juru Bicara Covid-19 Achmad Yurianto mengatakan, salah satu faktor utama penyumbang kasus positif Covid-19 disebabkan ketidakdisiplinan masyarakat menggunakan masker ketika berada di tempat keramaian (Rokom, 2021). Salah satu tempat yang banyak mengundang keramaian di masa pandemi adalah puskesmas.

Puskesmas adalah tempat pelayanan kesehatan yang menyelenggarakan upaya kesehatan masyarakat tingkat pertama. Saat ini orang – orang enggan datang ke rumah sakit dikarenakan takut jika dituduh terkena virus Covid-19 (Fahlevi, 2021), sehingga orang – orang yang memerlukan pengobatan lebih memilih mengunjungi puskesmas yang berada di sekitar rumah mereka. Namun, masih banyak orang yang tidak menggunakan masker dengan alasan lokasi yang dituju dekat dari rumah (Demak Kominfo, 2020). Hal tersebut bisa menjadi penyebab penularan virus Corona pada lingkungan sekitar, oleh karena itu petugas yang berada di puskesmas

harus memperhatikan orang yang tidak menggunakan masker. Akan tetapi, petugas puskesmas tidak bisa menjangkau seluruh sudut ruangan yang ada di puskesmas. Untuk mengatasi hal tersebut bisa dilakukan dengan cara mendeteksi wajah pengunjung menggunakan kamera.

Mendeteksi wajah adalah cara mengenali wajah berdasarkan ciri–ciri yang berada di wajah tersebut baik dalam bentuk gambar, pola, warna, dan lainnya (Kenda, 2021). Manusia mempunyai kemampuan untuk mendeteksi suatu objek dengan benar, tetapi memiliki keterbatasan sendiri seperti kelelahan untuk bekerja dalam waktu yang lama tanpa istirahat (Darmanto, 2019). Sehingga membutuhkan sistem pendeteksi penggunaan masker di lingkungan puskesmas. Sistem pendeteksi penggunaan masker tersebut menggunakan metode *Convolutional Neural Network* berbasis *Machine Learning*.

Peneltian sebelumnya yang berjudul Face Mask Detection Covid-19 Using *Convolutional Neural Network (CNN)* merupakan penelitian yang mendeteksi masker wajah dengan menggunakan kamera webcam laptop dan pengambilan data *training* yaitu mengambil gambar secara langsung dari mahasiswa dan di internet (Septiana et al., 2020). Selanjutnya ada penelitian yang berjudul Incorrect Facemask-Wearing Detection Using Convolutional Neural Networks with Transfer Learning merupakan penelitian pendeteksi masker menggunakan algoritma *Convolutional Neural Networks* berupa aplikasi (Tomás et al., 2021).

Berdasarkan uraian tersebut, maka diusulkan solusi untuk mendeteksi penggunaan masker menggunakan metode CNN dengan media video CCTV

dimana data tersebut dikirim secara *real-time.*

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan sebelumnya, maka dapat dirumuskan permasalahan sebagai berikut :

1) Bagaimana menerapkan metode Convolutional Neural Network (CNN) dalam mendeteksi penggunaan masker.
2) Bagaimana merancang sistem pendeteksi penggunaan masker untuk mendeteksi pengunjung yang tidak menggunakan masker di Puskesmas Sudiang Raya.

1.3 Ruang Lingkup Penelitian

Penelitian ini memiliki batasan-batasan sebagai berikut :

1) Pendeteksi pengguna masker di Puskesmas Sudiang Raya dengan menggunakan metode Convolutional Neural Network (CNN).
2) Dataset yang digunakan adalah gambar wajah yang menggunakan masker, tidak menggunakan masker, dan penggunaan masker yang salah.
3) Data yang digunakan untuk menguji adalah data yang berbentuk video.
4) Pengambilan data dilakukan pada salah satu pintu di Puskesmas Sudiang Raya pada pukul 08.00 sampai 11.00 WITA.
5) Pengambilan data menggunakan kamera webcam eksternal yang dijadikan sebagai CCTV.

1.4 Tujuan Penelitian

Berdasarkan latar belakang dan rumusan masalah yang telah diuraikan sebelumnya, tujuan penelitian ini adalah sebagai berikut :

1) Menerapkan metode Convolutional Neural Network (CNN) dalam mendeteksi penggunaan masker.
2) Merancang sistem pendeteksi penggunaan masker untuk mendeteksi pengunjung yang tidak menggunakan masker di Puskesmas Sudiang Raya.

1.5 Manfaat Penelitian

Penelitian ini diharapkan memiliki manfaat sebagai berikut :

1) Memperkenalkan algoritma Convolutional Neural Network (CNN) yang dapat diterapkan untuk sistem atau aplikasi pendeteksi wajah yang menggunakan masker atau tidak.
2) Membantu Puskesmas Sudiang Raya dalam mengawasi orang-orang yang menggunakan masker atau tidak.

BAB II TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Ada beberapa penelitian sebelumnya yang mendukung penelitian ini diantaranya adalah penelitian yang dilakukan oleh Septiana dkk yang berjudul Face Mask Detection Covid-19 Using Convolutional Neural Network (CNN) dimana penelitian ini membangun sistem pendeteksi masker dengan menggunakan kamera webcam laptop. Dataset yang diambil bervariasi dengan gambar wajah menggunakan masker dengan atribut memakai topi, hijab, dan kacamata. Sistem ini memiliki jarak pendeteksian maksimal 2,3 meter (Septiana et al., 2020).

Penelitian lain yang dilakukan oleh Abdul dkk yang berjudul Prototipe Pendeteksi Masker Pada Ruangan Wajib Masker Untuk Kendali Pintu Otomatis Berbasis Deep Learning Sebagai Pencegahan Penularan Covid-19 dimana penelitian ini membangun sistem pendeteksi masker untuk mengendalikan pintu otomatis dengan menggunakan kamera webcam laptop (Abdul et al., 2020). Oleh karena itu, penelitian ini membahas mengenai pendeteksi masker melalui video CCTV dengan menggunakan metode CNN. Penelitian ini dibuat berdasarkan permasalahan yang ada pada saat ini dan merujuk pada penelitian sebelum–sebelumnya.

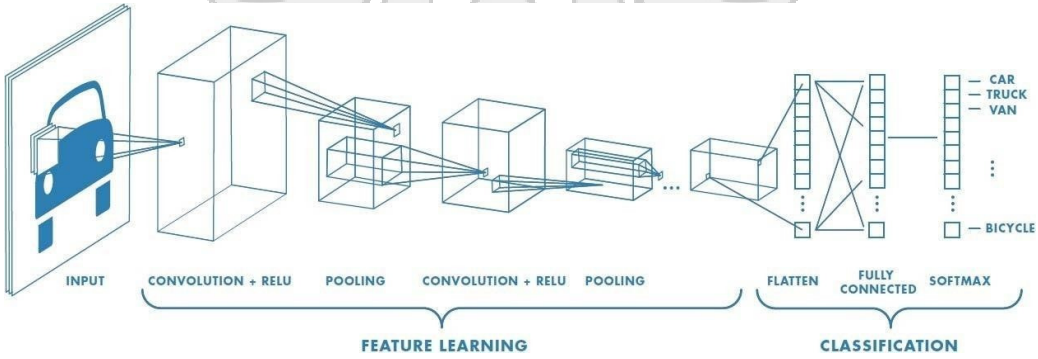
2.2 Deep Learning

Deep Learning adalah bagian dari *Machine Learning* yang saat ini ramai diperbincangkan dengan memiliki banyak lapisan (*hidden layer)* sehingga membentuk tumpukan lapisan dimana lapisan tersebut merupakan metode yang melakukan klasifikasi perintah yang diinput dan menghasilkan output (Santoso &

Ariyanto, 2018). Salah satu metode Deep Learning yangsaat ini sedang berkembang adalah metode Convolutional Neural Network (CNN).

2.2.1 Convolutional Neural Network

Convolutional Neural Network (CNN) adalah salah satu metode deep learning yang dapat menerima input berupa gambar dan menentukan objek apa saja yang terdapat pada gambar tersebut sehingga mampu membuat mesin belajar untuk membedakan gambar yang satu dengan gambar yang lainnya. CNN digunakan untuk mengklasifikasi data yang terlabel dengan menggunakan metode Supervised Learning. Cara kerja dari supervised learning adalah terdapat data yang dilatih dan terdapat variabel yang ditargetkan sehingga tujuan dari metode ini adalah mengelompokan suatu data ke data yang sudah ada (Ilahiyah & Nilogiri, 2018). CNN terdiri dari beberapa layer yang berfungsi untuk melakukan filter pada gambar di setiap proses training dimana proses training ini memiliki 3 tahapan yaitu Convolutional Layer, Pooling Layer, dan Fully Connected Layer.



Gambar 2. 1 Arsitektur CNN

Pada gambar di atas merupakan arsitektur dari CNN yang dimana memiliki 2 lapisan diantaranya adalah Feature Learning yang berupa Convolutional dan Pooling. Lapisan kedua yaitu Classification yang berupa Flatten, Fully Connected,

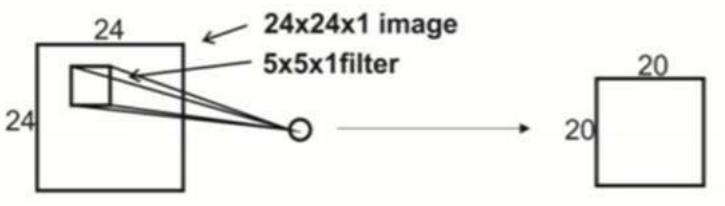
dan Softmax.

a. Feature Learning

Lapisan-lapisan yang terdapat dalam Feature Learning berguna untuk mentranslasikan suatu input menjadi menjadi features berdasarkan ciri dari input tersebut yang berbentuk angka–angka dalam vektor. Lapisan ekstraksi fitur ini terdiri dari Convolutional Layer dan Pooling Layer.

1) Convolutional Layer

Convolutional layer merupakan layer pertama yang menerima gambar pada arsitektur dengan melakukan konvolusi yaitu mengkombinasikan linier filter terhadap daerah lokal (Nurfita & Ariyanto, 2018). Convolutional layer terdiri dari neuron yang tersusun sedemikian rupa sehingga membentuk sebuah filter dengan panjang dan tinggi (pixels). Sebagai contoh, layer pertama pada feature extraction layer biasanya adalah conv. Layers dengan ukuran 5x5x3. Panjang 5 pixels, tinggi 5 pixels dan tebal atau jumlah 3 buah sesuai dengan channel dari image tersebut. Ketiga filter ini akan digeser ke seluruh bagian dari gambar. Setiap pergeseran akan dilakukan operasi "dot" antara input dan nilai dari filter tersebut sehingga menghasilkan sebuah output atau biasa disebut sebagai activation map atau feature map (P. A. Nugroho et al., 2020).

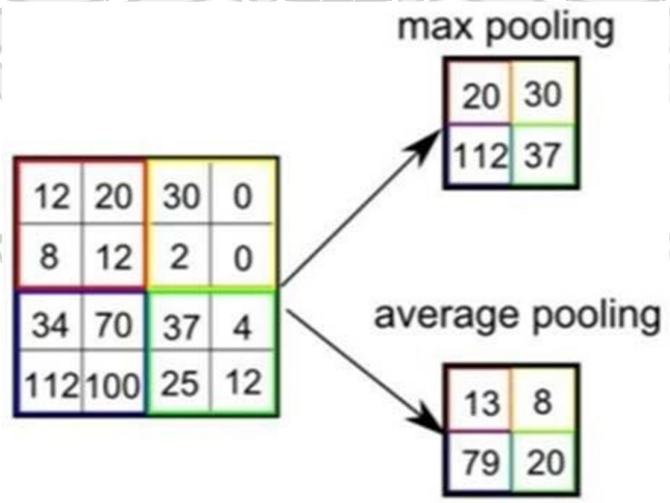


Gambar 2. 2 Proses Konvolusi

Sumber: (Nurfita & Ariyanto, 2018)

2) Pooling Layer

Pooling layer menerima output dari convolution layer. Tujuan dari penggunaan pooling layer adalah mengurangi dimensi dari feature map (downsampling), sehingga mempercepat komputasi karena parameter yang harus diupdate semakin sedikit dan mengatasi overfitting (Eural & Cnn, 2021). Pooling bekerja dengan cara membagi input menjadi potongan-potongan kecil, seperti pada penjelasan convolutional layer, dan mengambil nilai terbesar, atau rata-rata dari seluruh value dalam potongan tersebut, tergantung dari jenis pooling yang digunakan (Prayogo et al., 2020). Pooling yang biasa digunakan adalah Max Pooling dan Average Pooling. Max Pooling untuk menentukan nilai maksimum tiap pergeseran filter, sementara Average Pooling akan menentukan nilai rata-ratanya.



Gambar 2. 3 Proses Pooling Layer

Sumber: (Prayogo et al., 2020)

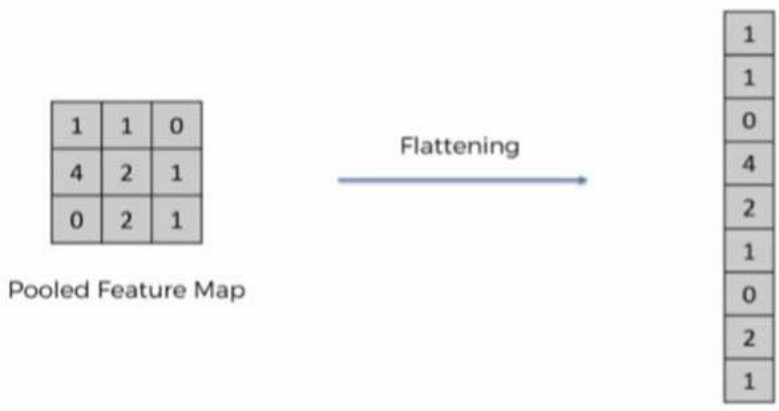
b. Classification

Lapisan ini berguna untuk mengklasifikasikan tiap neuron yang telah

diekstraksi fitur pada sebelumnya. Lapisan ini terdiri dari Flatten, Fully Connected, dan Softmax.

1) Flatten

Lapisan pertama pada tahap klasifikasi adalah flatten yang dimana Lapisan ini akan mengubah bentuk dari feature map yang semula berbentuk matriks menjadi sebuah vector satu dimensi. Hal ini dilakukan agar feature map dapat diproses pada lapisan fully connected (Handono et al., 2020).



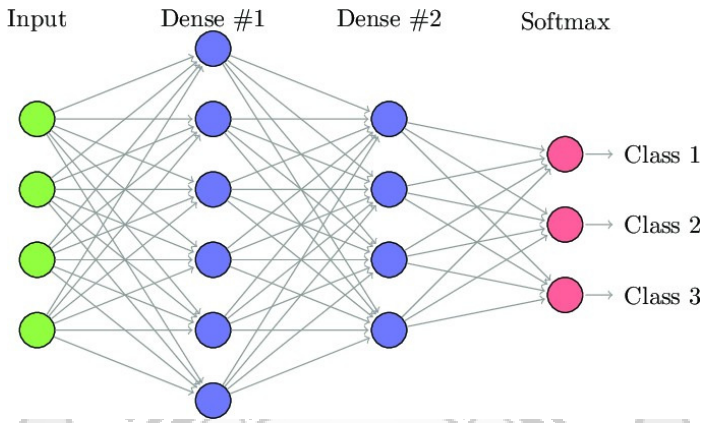
Gambar 2. 4 Proses Flatten

Sumber: (Handono et al., 2020)

2) Fully Connected

Fully-Connected Layer biasanya digunakan pada metode Multilayer Perceptron dimana semua neuron dihubungkan secara keseluruhan yang bertujuan untuk mengolah data sehingga dapat diklasifikasikan (Alamsyah & Pratama, 2020). Pada fully connected layer memiliki bagian yang sama yaitu : beberapa input layer, hidden layer, activation function, dan output layer (Qudsi et al., 2020). Perbedaan antara lapisan Fully- Connected dan lapisan konvolusi biasa adalah neuron di lapisan konvolusi terhubung hanya ke daerah tertentu pada input. Sementara lapisan Fully-Connected memiliki neuron yang secara keseluruhan terhubung. Namun,

kedua lapisan tersebut masih mengoperasikan produk dot, sehingga fungsinya tidak jauh berbeda.



Gambar 2. 5 Fully-Connected Layer

Sumber: (Pelletier et al., 2019)

2.3 OpenCV

OpenCV atau *Computer Vision* adalah *library* komputer yang dapat mengolah gambar atau video yang dikonversi dari analog menjadi digital kemudian diolah di komputer. OpenCV merupakan *library* yang *open source* dan memiliki banyak pilihan algoritma yang dibutuhkan (Zhu & Cheng, 2020) misalnya mendeteksi dan pengenalan wajah, mendeteksi objek, menemukan gambar yang mirip dengan gambar dari database, menghapus mata merah dari gambar yang diambil menggunakan flash, dan lain – lain (Sudhir Bussa et al., 2020).

2.4 YOLO (You Only Look Once)

YOLO (You Only Look Once) adalah salah satu model berbasis deep learning yang dikembangkan untuk mendeteksi sebuah objek secara real-time (Rachmawati & Widhyaestoeti, 2020). Dalam penelitian yang dilakukan oleh Redmon, YOLO dapat melakukan pengenalan objek secara *real-time* dengan kecepatan 45 fps

karena memiliki arsitektur yang sederhana, selain itu akurasi rata-rata yang didapat mencapai 88% dalam ImageNet 2012 Validation (Rahman et al., 2022). Cara kerja YOLO dengan melihat citra secara keseluruhan, kemudian melewati *neural network* dan otomatis mendeteksi objek yang ada.

YOLOv4 dikembangkan oleh 3 authors yaitu Alexey Bochkovskiy, Chien-Yao, dan Hong-Yuan Mark Liao. Pada tulisan (Hidayatullah, 2021) dicantumkan YOLOv5 memiliki kelemahan pada saat dijadikan pembanding pada karya ilmiah dan dokumentasi YOLOv5 kurang lengkap dibandingkan YOLOv4 sehingga YOLOv4 lebih direkomendasikan dalam mendeteksi sebuah objek dibandingkan YOLOv5. YOLOv4 sangat direkomendasikan untuk mendeteksi objek secara *real-time* dikarenakan mampu mendeteksi objek lebih cepat dan akurat dibandingkan dengan versi YOLO lainnya dan object detector lainnya (Noor et al., 2020).

2.4.1 Perbandingan YOLOv4 dengan Model Deep Learning Lain

Beberapa model *deep learning* yang popular yaitu *Single Shot Multibox Detector* (SSD) dan *Mask Region Based Convolotional Neural Network* (Mask-RCNN) (Hidayatullah, 2021). Beberapa hal perbandingan YOLOv4 dengan model *deep learning* tersebut yaitu sebagai berikut.

1) Model Pendeteksian

YOLOv4 dan SSD menerapkan pendeteksian objek dengan satu tahap, sedangkan Mask-RCNN menerapkan pendeteksian objek dengan dua tahap. Adapun kelemahan dari metode dua tahap yaitu proses pendeteksian prosesnya berjalan lambat karena harus melakukan dua kali proses menggunakan *neural*

network. Proses pendeteksian secara dua tahap yaitu tahap pertama digunakan untuk mencari *region of interest (RoI)* yang memiliki probabilitas tinggi akan adanya objek. Setelah itu, pada tahap kedua adalah tahap klasifikasi dari *RoI* hasil tahap pertama.

2) Ukuran *Weights*

Ukuran *Weights* YOLOv4 sebesar 256 MB, SSD sebesar 138 MB, dan Mask-RCNN sebesar 256 MB. Pada umumnya besarnya ukuran *weights* menandakan bahwa *layer* yang digunakan lebih banyak dan akan berpengaruh ke kecepatan saat proses deteksi. Namun berbeda dengan YOLOv4, meskipun ukuran *weights*nya besar proses deteksinya pun cepat.

3) Akurasi

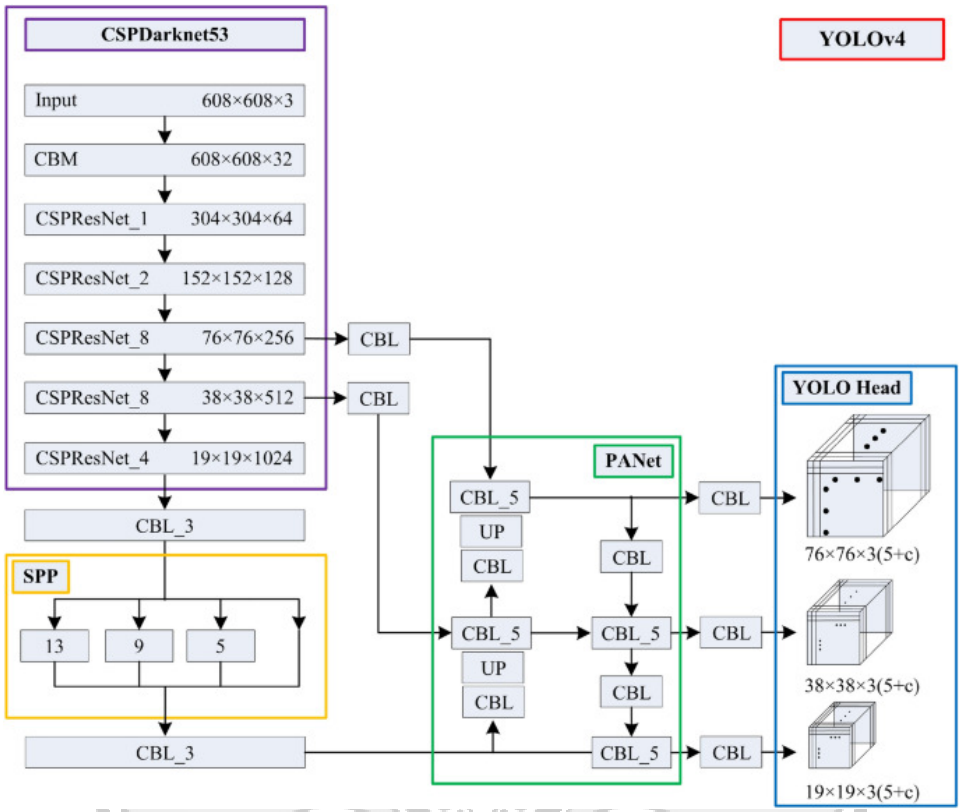
Akurasi dihitung menggunakan *mean Average Precision (mAP)* pada pendeteksian objek terhadap Microsoft *Common Objects in Context* (COCO) yang terdiri dari 80 kategori objek. YOLOv4 mendapatkan mAP tertinggi dibanding dua model lainnya yaitu 43, sedangkan Mask-RCNN yang mendapat mAP sebesar 40,3, dan SSD dengan mAP sebesar 35,1.

4) Kecepatan

Kecepatan dihitung dengan *Frame Per Second* (FPS). FPS adalah seberapa banyak *frame* yang dapat diproses dalam satu detik. Sama seperti perhitungan akurasi, dataset yang digunakan yaitu dataset COCO. YOLOv4 mendapatkan hasil FPS yang paling cepat dibandingkan SSD dan Mask-RCNN yaitu 83 FPS. Sedangkan SSD dan Mask-RCNN mendapatkan hasil FPS yaitu 39 FPS dan 3 FPS.

2.4.2 Arsitektur YOLOv4

Arsitektur YOLOv4 menggunakan backbone: CSPDarknet53, neck: SPP dan PAN, dan head: YOLOv3 seperti pada Gambar 2.1. Feature extraction yang bernama CSPDarknet53 ini memiliki 29 convolutional layers 3×3, 725×725 pada receptive field dan 27.6 M parameter. Berdasarkan nilai ini, CSPDarknet53 merupakan model yang optimal sebagai feature extraction atau backbone dari detektor. Beberapa pengaruh dari kemampuannya saat citra di-input yakni bisa melihat objek secara detail, dan meningkatkan jumlah kombinasi setiap titik di suatu citra. SPP pada neck membantu memisahkan fitur konteks secara signifikan dan tidak mengurangi kecepatan operasi jaringan. Sedangkan, PANet digunakan sebagai parameter penyatuan dari beberapa level backbone (Bochkovskiy et al., 2020).



Gambar 2. 6 Arsitektur YOLOv4

Sumber : (Wang et al., 2021)

2.4.3 Konfigurasi Hyperparameter YOLOv4

Model YOLOv4 yang merupakan *neural network* memiliki *hyperparameter* seperti *batch size, learning rate,* dan lain-lain. *Hyperparameter* tersebut harus disetel saat proses training agar mendapatkan nilai yang optimal. Untuk *hyperparameter tunning* tersebut dibutuhkan data untuk menguji saat proses training. Namun data yang digunakan bukan dari *training set*, oleh karena itu dibutuhkan satu bagian lagi yaitu *validation set* (Hidayatullah, 2021). Biasanya *validation* dialokasikan sekitar 10-20% dari *training set* (Rosebrock, 2017). Adapun *Hyperparameter* yaitu sebagai berikut.

1) Batch Size

Batch Size adalah jumlah sample data yang akan diproses dalam satu iterasi pelatihan atau variable yang menentukan seberapa banyak *training data* yang dimasukkan saat melakukan *training.* Semakin kecil nilai *batch size* yang digunakan, maka proses *training* semakin cepat. Sementara semakin besar nilai batch size, proses *training* pun akan memakan waktu yang cukup lama karena membutuhkan kapasitas *storage* yang lebih banyak.

Hal ini juga turut mempengaruhi akurasi pada sistem. Jika nilai batch size yang digunakan semakin besar, maka akurasinya pun akan semakin tinggi, karena akan semakin banyak fitur yang dipelajari oleh sistem (Radiuk, 2018).

2) *Subdivisions*

Subdivisions membagi nilai batch menjadi lebih kecil lagi, dan dapat disebut dengan mini-batch. Jika menggunakan nilai batch sebesar 64 dan dibagi dengan 8 sub divisions, maka menghasilkan nilai 8 yang berarti dilakukan proses training untuk 8 images tiap mini-bacth-nya (Kusuma et al., 2021). Proses ini berlangsung selama delapan kali hingga proses training pada satu batch tersebut selesai. Kemudian, sistem memproses batch selanjutnya yang juga bernilai 64. Proses sub divisions bertujuan untuk mempercepat proses training sekaligus meningkatkan akurasi dengan bantuan GPU.

3) *Learning Rate*

Learning rate merupakan penentu seberapa banyak weight yang diperbarui dalam proses backpropagation. Learning rate juga menentukan kecepatan pada iterasi sehingga dapat mencapai loss function minimum. Proses training berjalan

semakin cepat jika nilai learning rate semakin tinggi (Wu et al., 2019). Namun, nilai learning rate yang terlalu tinggi juga dapat menyebabkan nilai loss function turun-naik tidak menentu, sehingga dibutuhkan beberapa kali percobaan untuk mendapatkan nilai learning rate yang optimal (Konar et al., 2020).

4) *Max Batches*

Max batches merupakan banyaknya iterasi pada proses training data. Semakin tinggi nilai *max batch,* maka sistem akan semakin banyak mempelajari data training. Jumlah training data tidak boleh lebih dari jumlah *max batches*. Nilai max batches perlu disesuaikan dengan jumlah kelas dari objek yang akan dideteksi.

$$Max\ Batches = Jumlah\ classes \times 2000 \qquad (1)$$

Adapun *Hyperparameter* lainnya yang dikonfigurasi ditunjukkan berikut ini.

Tabel 2. 1 Hyperparameter YOLOv4

Hyperparameter	**Deskripsi**
Weight dan Height	Dimensi input gambar yang akan dilatih
Steps	Epoch pada saat dimana *learning rate* mengecil sesuai dengan *scale factor*
Scale	*Scale factor* pengecil *learning rate*
Momentum	*Hyperparameter* yang digunakan untuk membantu mengetahui langkah

	selanjutnya dengan pengerahuan langkah sebelumnya pada proses *training*
Filters	Jumlah filter yang disesuaikan dengan jumlah *class.* *Class = (jumlah class + 5) x 3* *(2)*
Class	Jumlah objek *class* yang akan di-*train*

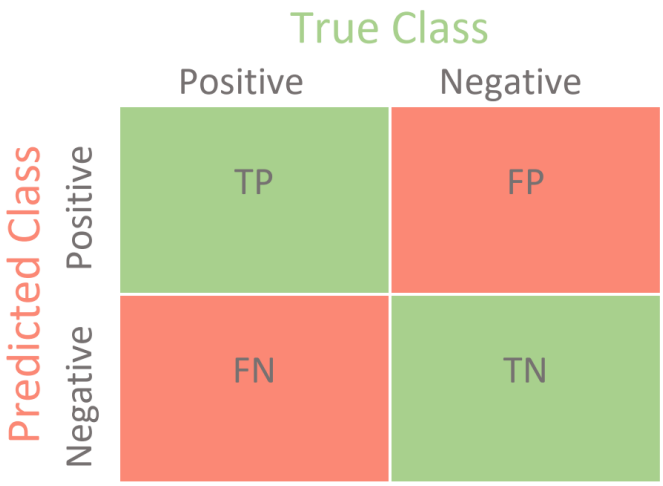
2.5 Parameter Kinerja

Parameter kinerja digunakan untuk menentukan tingkat keberhasilan dari kinerja sistem menggunakan model yang telah dilatih untuk mendeteksi objek. Berikut parameter–parameter kinerja tersebut.

2.5.1 Confusion Matrix

Confusion matrix dapat disebut sebagai, *error matrix* yang memberikan informasi perbandingan hasil klasifikasi yang dilakukan oleh sistem (model) dengan hasil klasifikasi sebenarnya. *Confusion matrix* adalah alat yang berguna untuk menganalisis seberapa baik atau seberapa akurat metode klasifikasi dapat mengenali objek pengamatan dari kelas yang berbeda (K. S. Nugroho, 2020).

Confusion matrix berbentuk tabel matriks yang menggambarkan kinerja model klasifikasi pada serangkaian data uji yang nilai sebenarnya diketahui.



Gambar 2. 7 Confusion Matrix

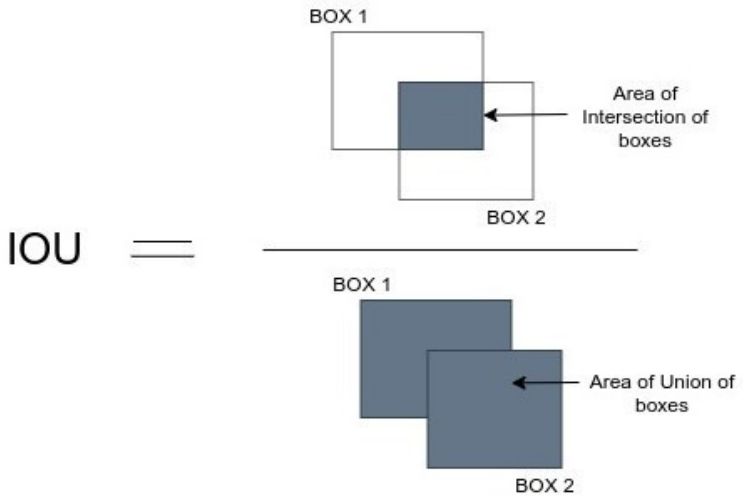
(Sumber : towardsdatascience.com)

Dimana TP (*True Positive*) merupakan jumlah sampel positif yang diprediksi secara akurat, FP (*False Positive*) merupakan jumlah sampel yang sebenarnya negative tetapi diprediksi positif, FN (*False Negative*) yang berarti jumlah sampel yang sebenarnya positif tetapi diprediksi negative, dan TN *(True Negative)* merupakan jumlah sampel negative yang diprediksi secara akurat.

2.5.2 Intersection over Union (IoU)

IoU merupakan metrik yang mengevaluasi keakuratan sistem dalam mendeteksi objek pada dataset yang telah dilatih (Zhang et al., 2019). IoU membandingkan ground-truth atau objek pada citra dengan predicted bounding box dari model. Nilai IoU pada deteksi objek berperan sebagai nilai threshold Terdapat dua nilai threshold IoU yang umum digunakan, yaitu 0.5 dan 0.75. Pada penelitian ini, nilai threshold yang digunakan adalah 0.5. Jika nilai threshold IoU ≥ 0.5, maka objek terdeteksi sebagai True Positive (TP). Dan ketika threshold IoU < 0.5, maka

objek terdeteksi sebagai False Positive (FP). Pada object detection untuk menghitung presisi pada sebuah citra, diperlukan perhitungan Intersection over Union. Intersection over Union dapat dilihat pada Gambar 2.8 yang menghitung seberapa banyak wilayah prediksi *bounding box* yang beririsan dengan bounding box sebenarnya. IoU pada umumnya memiliki batasan (*threshold*) yang menentukan apakah sebuah prediksi bernilai *true positive* atau *false positive* (Rezatofighi et al., 2019).



Gambar 2. 8 Intersection of Union (IoU)
(Sumber : medium.com)

2.5.3 Mean Average Precision (mAP)

Salah satu metrik yang paling umum digunakan dalam permasalahan pendeteksian objek adalah metrik *Average Precision* (AP). Secara ringkas, average precision menghitung nilai rataan precision untuk setiap nilai recall yang berada pada rentang 0 hingga 1. Precision adalah nilai yang menentukan seberapa akurat prediksi yang dilakukan.

$$Precision = \frac{TP}{TP+FP} \qquad (3)$$

Recall menandakan seberapa baik model yang dihasilkan mencari nilai positif. Recall dapat dinyatakan dengan persamaan (4)

$$Recall = \frac{TP}{TP+FN} \qquad (4)$$

Sebagai ringkasan, nilai precision adalah jumlah prediksi nilai positif yang benar dibagi dengan jumlah keseluruhan nilai yang diprediksi sebagai nilai positif. Nilai recall adalah jumlah prediksi positif yang benar dibagi dengan jumlah nilai positif keseluruhan sebenarnya. Untuk perhitungan *mean Average Precision* atau mAP, hasil perhitungan average precision untuk tiap kelas dirata-ratakan sesuai jumlahnya. Semakin tinggi mAP berarti pendeteksi objek semakin akurat.

2.5.4 Accuracy

Akurasi adalah rasio prediksi benar (positif dan benar) dengan menggunakan keseluruhan data. Untuk mendapatkan hasil akurasi dengan menggunakan persamaan berikut ini.

$$Akurasi = (TP + TN) / (TP+FP+FN+TN) \qquad (5)$$

BAB III METODE PENELITIAN

3.1 Tempat dan Waktu Penelitian

Penelitian ini dilaksanakan di Puskesmas Sudiang Raya Bumi Sudiang Permai pada bulan Maret 2022 sampai dengan September 2022.

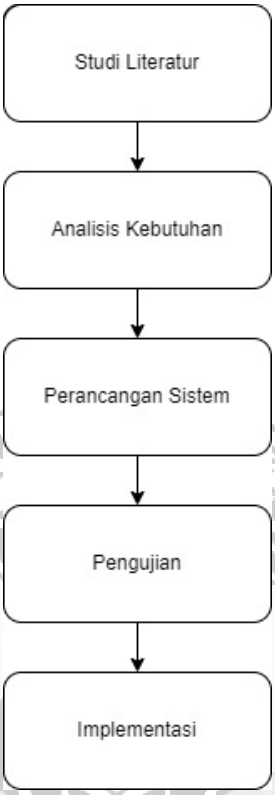
3.2 Jadwal Penelitian

Tabel 3. 1 Jadwal Penelitian

No.	Kegiatan	Maret	April	Mei	Juni	Juli	Agustus	September
1.	Studi Literatur							
2.	Analisis Kebutuhan							
3.	Perancangan Sistem							
4.	Implementasi							
5.	Pengujian							
6.	Evaluasi							
7.	Penyusunan Laporan							
8.	Sidang Skripsi							

3.3 Prosedur Penelitian

Prosedur penelitian dilakukan agar penelitian berjalan dengan terstruktur sehingga dapat mencapai tujuan penelitian. Berikut alur dari prosedur penelitian yang digambarkan pada Gambar 3.1.



Gambar 3. 1 Prosedur Penelitian

3.3.1 Studi Literatur

Studi literatur dilakukan dengan cara mengumpulkan data dari beberapa sumber yaitu dari beberapa jurnal ilmiah, artikel ilmiah, buku, skripsi, maupun literatur lainnya yang berkaitan dengan masalah penelitian yang dapat dijadikan acuan pembahasan dalam penelitian ini.

3.3.2 Analisis Kebutuhan

Dalam penelitian ini kebutuhan sistem yang digunakan terdapat beberapa perangkat keras dan perangkat lunak yaitu sebagai berikut.

a. Perangkat Keras

1) Laptop HP Pavilion 14-ce2013TX

2) GPU NVIDIA

3) Kamera Eksternal Webcam (CCTV)

b. Perangkat Lunak

1) Sistem Operasi Windows 10

2) OpenCV

3) Visual Studio 2019

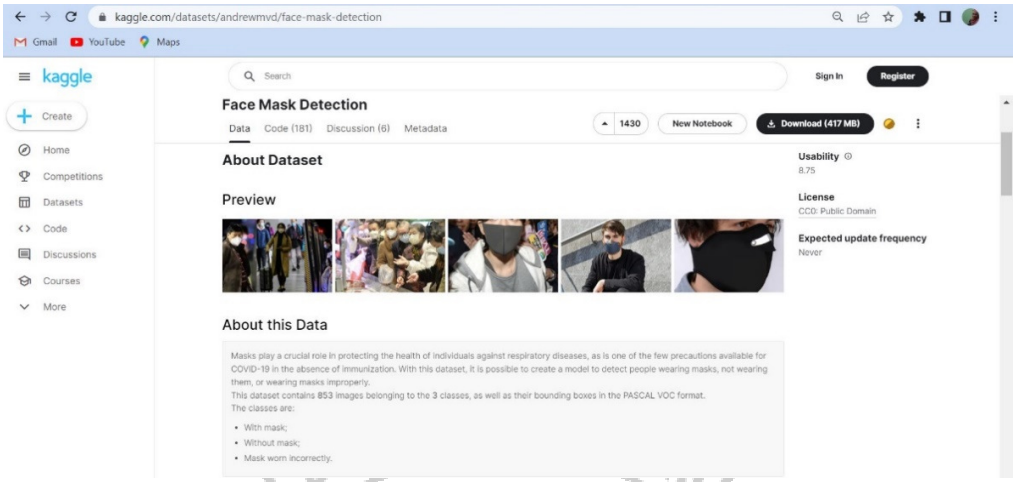
4) Bahasa Pemrograman Python

3.3.3 Perancangan Sistem

Dalam perancangan sistem terdapat beberapa tahapan proses yang dilakukan antara lain pengumpulan data, pelabelan data, pembagian dataset, dan klasifikasi deteksi pengguna masker.

1) Pengumpulan Data

Pada proses pengumpulan data ini menggunakan dataset dari website *www.kaggle.com*. Dataset ini terdiri dari kurang lebih 300 gambar yang berupa gambar orang menggunakan masker, tidak menggunakan masker, dan orang yang menggunakan masker dengan cara yang salah.



Gambar 3. 2 Website www.kaggle.com

2) Pelabelan / Anotasi Data Gambar

Tahapan kedua yang dilakukan yaitu pelabelan data gambar atau anotasi. Pelabelan berfungsi untuk memberikan identitas objek pada setiap gambar, agar objek tersebut memiliki identitas sesuai dengan jenis *class* masing-masing. Pelabelan atau anotasi data dilakukan dengan memberi *bounding box* setiap objek pada citra dengan nama *class* objeknya.

3) Pembagian Dataset

Tahapan selanjutnya adalah melakukan pembagian dataset yang berfungsi untuk mengevaluasi model dan mencegah adanya *overfitting.* Dalam model YOLOv4 memiliki *hyperparameter* seperti *learning rate, filters, classes,* dan lain – lain. Hal ini didapatkan dengan melakukan proses *hyperparameter tunning* pada saat proses *training* untuk mendapatkan nilai yang baik. Untuk melakukan hal tersebut dibutuhkan satu bagian lagi yaitu data *validation* sehingga pembagian

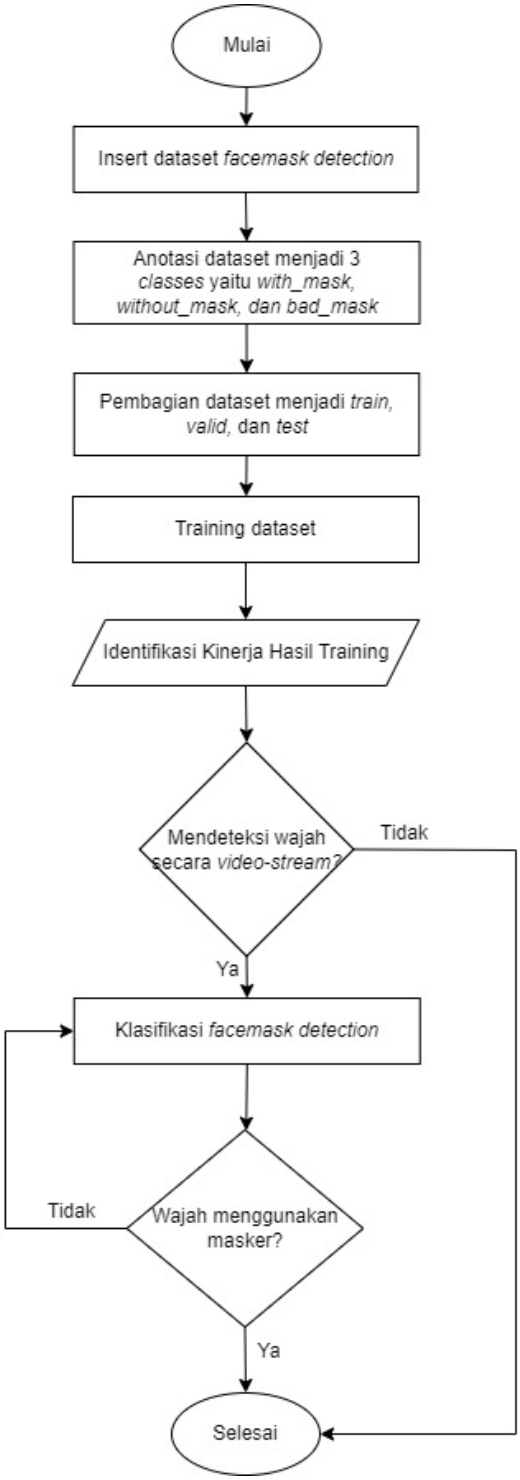
dataset yang telah dianotasi tersebut dibagi menjadi 3 bagian yaitu *training, validation,* dan *testing*.

4) *Training Model*

Tahapan selanjutnya adalah proses *training* yang bertujuan untuk mendapatkan hasil yang optimal dengan cara mengolah gambar yang telah dianotasi sehingga akan terbentuk karakteristik sesuai dengan masing – masing *class* yang akan menjadi bahan pertimbangan pada komputer untuk memprediksi sebuah objek.

5) Deteksi Wajah Pengguna Masker

Tahapan selanjutnya adalah tahapan untuk mendeteksi wajah yang menggunakan masker, tidak menggunakan masker, dan penggunaan masker dengan cara yang salah. Langkah pertama adalah menginput *video-stream* melalui kamera eksternal CCTV setelah itu mendeteksi apakah wajah tersebut menggunakan masker atau tidak, jika wajah terdeteksi tidak menggunakan masker maka akan muncul di layar tidak menggunakan masker, begitupun dengan wajah menggunakan masker dan wajah menggunakan masker namun penggunaan maskernya salah.



Gambar 3. 3 Proses Sistem Pendeteksi Masker

3.3.4 Implementasi

Pada tahap ini sistem diimplementasikan metode Deep Learning dalam pendeteksian objek dan melakukan klasifikasi *facemask* menggunakan algoritma *Convolutional Neural Network (CNN)* dengan model YOLOv4 dan menerapkan semua yang terdapat pada perancangan sistem yaitu dataset yang digunakan berupa dataset gambar berupa wajah bermasker, tidak bermasker, dan penggunaan masker yang keliru kemudian menggunakan kamera webcam eksternal yang dijadikan sebagai CCTV dan terdapat beberapa *tools* untuk menggunakan model YOLOv4 dengan GPU Nvidia dalam pendeteksian objek yaitu Visual Studio 2019 untuk komponen C++ yang diperlukan oleh Nvidia untuk menjalankan proses *deep learning*, bahasa pemrograman Pyhton yang digunakan pada saat pembagian dataset, library OpenCV yang digunakan memproses video saat melakukan pendeteksian objek dengan menggunakan YOLOv4, CMake digunakan untuk melakukan *compile* kode pada YOLOv4 serta CUDA dan cuDNN yang digunakan untuk menjalankan GPU Nvidia pada *deep learning.* Tahap ini menentukan keberhasilan dari sistem yang akan dibangun.

3.3.5 Pengujian

Pada tahapan pengujian merupakan tahapan terakhir dalam penelitian ini yaitu untuk menguji tingkat akurasi atau performansi model YOLOv4 berupa *precision, recall, Average Precision,* dan mAP (*mean Average Precision)* kemudian memastikan apakah sistem ini dapat mendeteksi wajah dan mampu klasifikasi objek wajah yang menggunakan masker dan tidak menggunakan masker melalui *video-stream* CCTV yang dilakukan di Puskesmas Sudiang Raya.

Selanjutnya adalah melakukan evaluasi model yang bertujuan untuk mengetahui prediksi sistem bekerja dengan baik atau tidak dengan menggunakan skenario pengujian menghitung jumlah data sampel pengunjung yang datang (data secara aktual) dan membandingkan dengan jumlah data sampel pengunjung yang terdeteksi oleh sistem (data sistem).

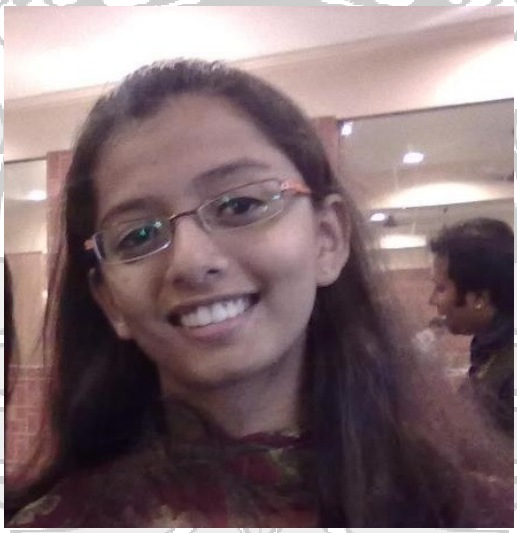
Gambar 3. 4 Peta Map Puskesmas Sudiang Raya

(Sumber : Google Maps)

BAB IV HASIL DAN PEMBAHASAN

4.1 Pengumpulan Dataset

Dataset yang digunakan adalah data yang berupa sekumpulan gambar. Dataset ini didapatkan dari website *www.kaggle.com* yang terdiri dari 3 *class* yaitu wajah yang menggunakan masker, tidak menggunakan masker, dan wajah yang menggunakan masker namun penggunaan maskernya salah. Semua file gambar tersebut disimpan dalam satu folder.



Gambar 4. 1 Wajah Tidak Menggunakan Masker

Gambar 4. 2 Gambar Wajah Menggunakan Masker

Gambar 4. 3 Gambar Wajah Menggunakan Masker dengan Cara yang Salah

4.2 Implementasi

Pada tahap implementasi ini menerapkan semua yang terdapat perancangan sistem dan dataset yang akan digunakan.

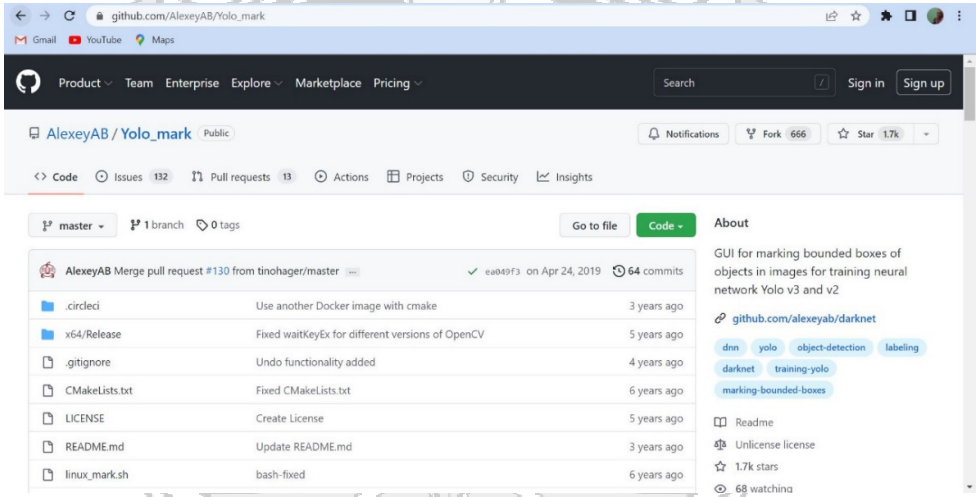
4.2.1 Anotasi Dataset

Proses anotasi dataset atau pelabelan data dilakukan dengan cara membuat label dengan cara memberikan *bounding box* beserta nama *class* pada objek di setiap gambar. *Class* terdiri dari 3 jenis yaitu *with_mask* merupakan *class* yang berupa gambar wajah menggunakan masker, *without_mask* merupakan *class* yang berupa

gambar wajah tidak menggunakan masker, dan *bad_mask* merupakan *class* yang berupa gambar wajah menggunakan masker namun penggunaan masker salah.

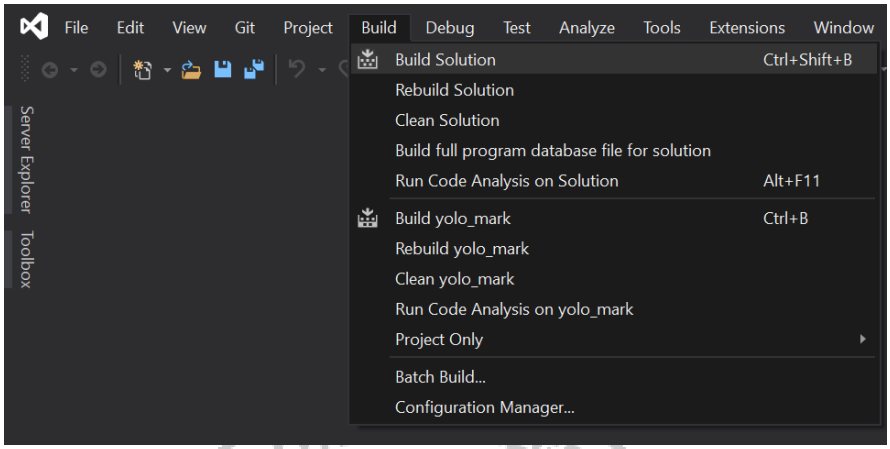
Setelah mengetahui jumlah dan nama *class* selanjutnya adalah melakukan anotasi atau pelabelan data. Proses anotasi ini dilakukan dengan menggunakan *tools Yolo mark.*

1. Referensi *Tools Yolo_mark* diambil dari *repository* Github AlexeyAB pada link https://github.com/AlexeyAB/Yolo_mark.



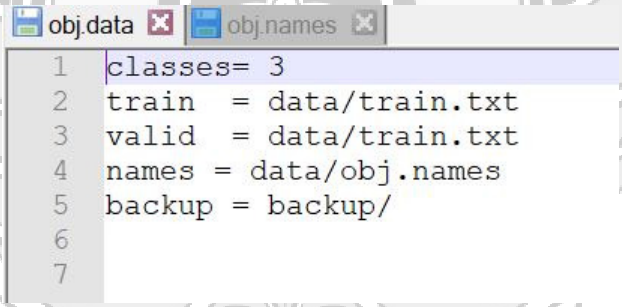
Gambar 4. 4 Website Github AlexeyAB

2. Setelah mengunduh *tools Yolo_mark* dan mengesktraknya buka *file yolo_mark.sln* menggunakan Visual Studio ubah mode *Debug* menjadi *Release* dan ubah *solution platform* menjadi x64. Selanjutnya adalah *Build Yolo_mark* seperti pada Gambar 4.5.

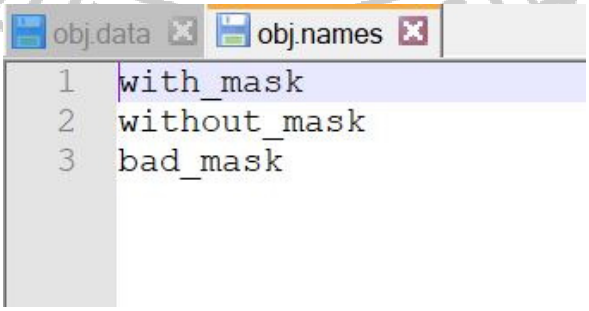


Gambar 4. 5 Build Yolo Mark

3. Selanjutnya adalah menentukan jumlah *class* dan nama *class* dengan membuat 2 *file* lalu simpan dengan *file* yang berekstensi *.data* dan *.names* seperti pada Gambar 4.6 dan Gambar 4.7 di bawah ini.

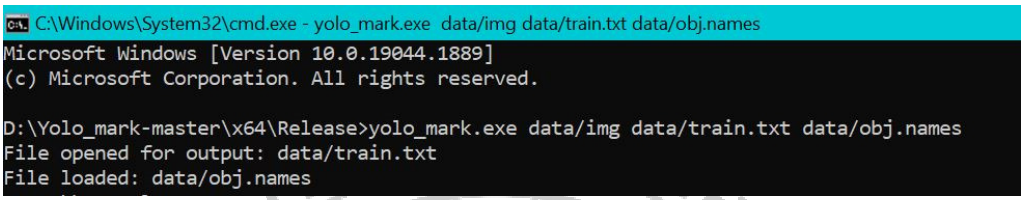


Gambar 4. 6 File obj.data



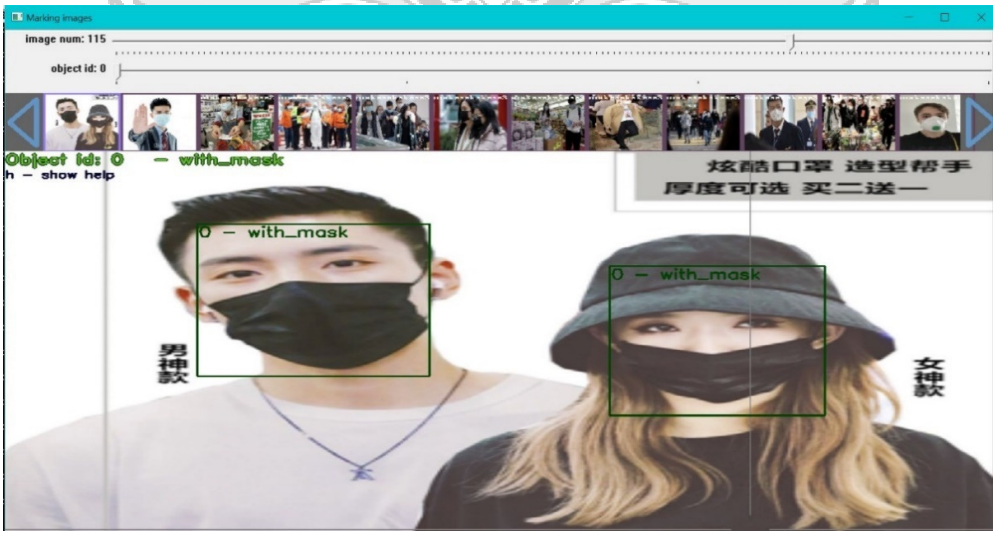
Gambar 4. 7 File obj.names

Pada gambar di atas menunjukkan bahwa file tersebut menentukan jumlah *class* dan nama dari setiap masing-masing *class* yang nantinya akan digunakan sebagai label atau identitas dari data setiap gambar pada proses anotasi model. Perintah untuk menjalankan proses anotasi ditunjukkan pada Gambar 4.8.



Gambar 4. 8 Perintah untuk Menjalankan Anotasi Data

Pada saat proses anotasi data masing – masing *class* diberi *id* yang berupa *id* 0 untuk *class with_mask, id* 1 untuk *class without_mask,* dan *id* 2 untuk *class bad_mask.* Berikut adalah proses anotasi dari setiap masing – masing *class.*



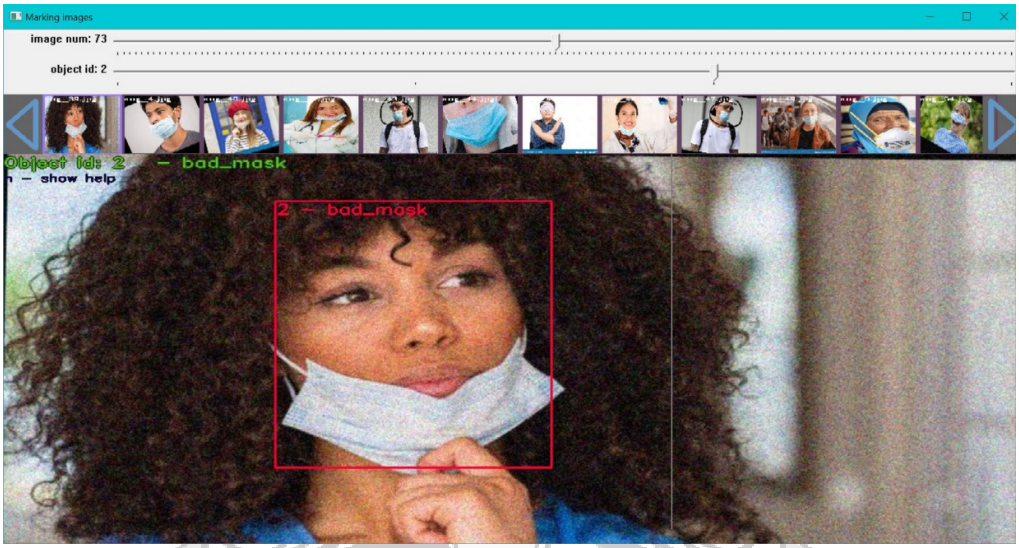
Gambar 4. 9 Anotasi Data Class with_mask

Gambar 4. 9 menunjukkan proses anotasi data dengan *class with_mask* dimana objek yang ditandai adalah objek wajah manusia yang menggunakan masker. Wajah yang menggunakan masker tersebut ditandai dengan *bounding box*

yang memiliki *id* 0 yang merupakan *id* untuk *class with_mask.* Proses anotasi data untuk *class with_mask* dilakukan secara berulang sesuai dengan jumlah data gambar yang telah dimasukkan yaitu sebanyak 100 gambar.

Gambar 4. 10 Anotasi Data Class without_mask

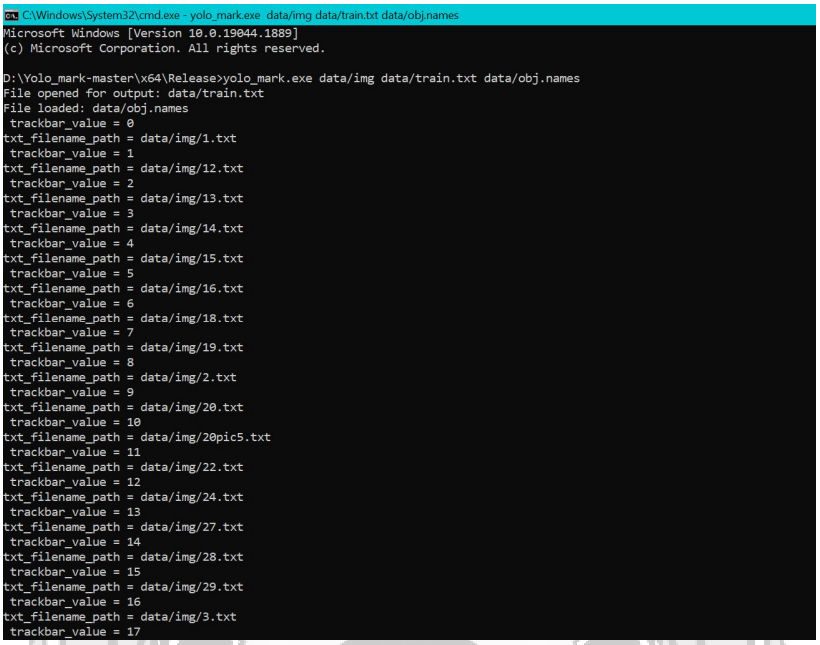
Gambar 4.10 menunjukkan proses anotasi data untuk *class without_mask*. Gambar yang akan dianotasi pada *class without_mask* adalah gambar manusia yang tidak menggunakan masker. Proses anotasi data ini ditandai dengan memberi *bounding box* pada wajah manusia yang tidak menggunakan masker. Gambar untuk objek wajah manusia tidak menggunakan masker yang dianotasi adalah sebanyak kurang lebih 100 gambar.



Gambar 4. 11 Anotasi data Class bad_mask

Sama halnya dengan *class with_mask* dan *without_mask* proses anotasi data untuk *class bad mask* dilakukan dengan cara memberi *bounding box* pada objek wajah manusia yang menggunakan masker namun cara penggunaannya salah. *Class bad_mask* menggunakan *id 2* yang sesuai dengan urutan baris yang telah ditentukan di file *obj.names.* Proses anotasi untuk *class bad_mask* yang dilakukan adalah sebanyak kurang lebih 100 gambar.

Hasil dari proses anotasi/pelabelan data akan tersimpan dalam file yang berekstensi .txt dan disimpan dalam folder yang sama dengan data-data gambar tersebut yang ditunjukkan pada Gambar 4.12 di bawah ini.



Gambar 4. 12 Proses Anotasi data

4.2.2 Pembagian Dataset

Pembagian dataset dibagi menjadi 3 bagian yaitu *training, validation,* dan *testing.* Data *training* berfungsi untuk training model. Data *validation* digunakan untuk proses validasi model dan mencegah adanya *overfitting*. Data *testing* digunakan untuk testing model.

```
6   # Add Parser
7   parser = argparse.ArgumentParser()
8
9   parser.add_argument("--train", type=int, default=80, help="Percentage of train set")
10  parser.add_argument("--validation", type=int, default=10, help="Percentage of validation set")
11  parser.add_argument("--test", type=int, default=10, help="Percentage of test set")
12  parser.add_argument("--folder", type=str, default="img", help="Folder that contain image")
13
14  args = parser.parse_args()
15
```

Gambar 4. 13 Penentuan persentase training, validation, dan testing set

Dalam pembagian dataset data *training* harus jauh lebih besar jumlahnya dibandingkan data *validation* dan *testing* dan yang digunakan adalah 80% *training*,

10% *validation,* dan 10% *testing.* Pembagian dataset merupakan folder yang berisi data berupa gambar atau berekstensi *img.*

```
def get_split_data(list_id):
    # # Train Set
    # Menghitung jumlah data train
    n_train = (count * args.train) / 100
    # Random
    train = sample(list_id, int(n_train))

    list_id = get_difference_from_2_list(list_id, train)

    # # Validation Set
    # Menghitung jumlah data validation
    n_valid = (count * args.validation) / 100

    # Random
    valid = sample(list_id, int(n_valid))

    # # Test Set
    test = get_difference_from_2_list(list_id, valid)

    return train, valid, test
```

Gambar 4. 14 Perhitungan jumlah data tiap set

Setelah menentukan persentase dari setiap masing-masing *set* selanjutnya adalah menghitung jumlah data *training, validation,* dan *testing.*

```
# Check train set
if((args.train < args.validation) or (args.train < args.test) ):
    print("Train set must has a biggest Percentage")
    exit()

# Check total percentage
total = args.train + args.validation + args.test
if(total > 100):
    print("Total Percentage must 100%")
    exit()

# Menghitung Jumlah Data
count = 0
list_id = []
for file in os.listdir(args.folder):
    if ((file.endswith(".jpg")) or (file.endswith(".png"))):
        list_id.append(count)
        count+=1

train, valid, test = get_split_data(list_id)
```

Gambar 4. 15 Mengecek training set dan total persentase

Selanjutnya adalah mengecek jumlah data yang ada pada *set training.* Jika jumlah persentase di *training set* lebih kecil dibandingkan *validation set* maka akan

muncul peringatan bahwa jumlah training harus lebih besar. Hal itu juga dapat terjadi pada kondisi jika jumlah persentase yang ada pada *training set* lebih kecil dibandingkan jumlah persentase pada *testing set* maka akan muncul peringatan yang sama sehingga pembagian dataset tidak dapat dilakukan.

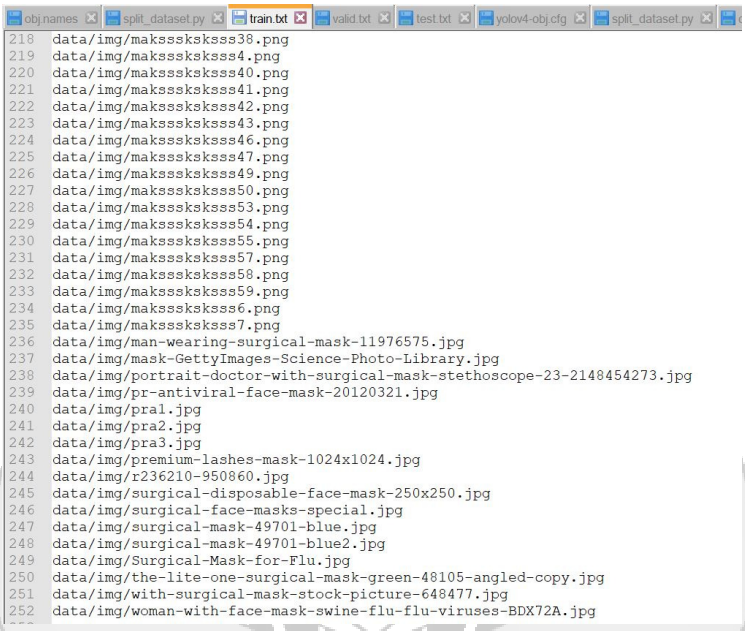
Setelah itu mengecek jumlah persentase dari setiap masing-masing *set* dengan menambahkan seluruh jumlah persentase *training, validation,* dan *testing set* maka hasilnya harus sebesar 100%, jika total persentase lebih dari 100 maka akan mengeluarkan output Total persentase harus 100 dan proses pembagian dataset tidak diproses. Setelah itu menghitung jumlah data untuk dimasukkan ke dalam setiap *set* berdasarkan perhitungan persentasenya.

```
67  # Overwrite text file
68  t = open("train.txt", "w")
69  t.close()
70  v = open("valid.txt", "w")
71  v.close()
72  te = open("test.txt", "w")
73  te.close()
74
75  count = 0
76  for file in os.listdir(args.folder):
77      if ((file.endswith(".jpg")) or (file.endswith(".png"))):
78          if(count in train):
79              f = open("train.txt", "a")
80              f.write("data/"+args.folder+"/"+file+"\n")
81              f.close()
82          elif(count in valid):
83              f = open("valid.txt", "a")
84              f.write("data/"+args.folder+"/"+file+"\n")
85              f.close()
86          else:
87              f = open("test.txt", "a")
88              f.write("data/"+args.folder+"/"+file+"\n")
89              f.close()
90
91          count+=1
92
```

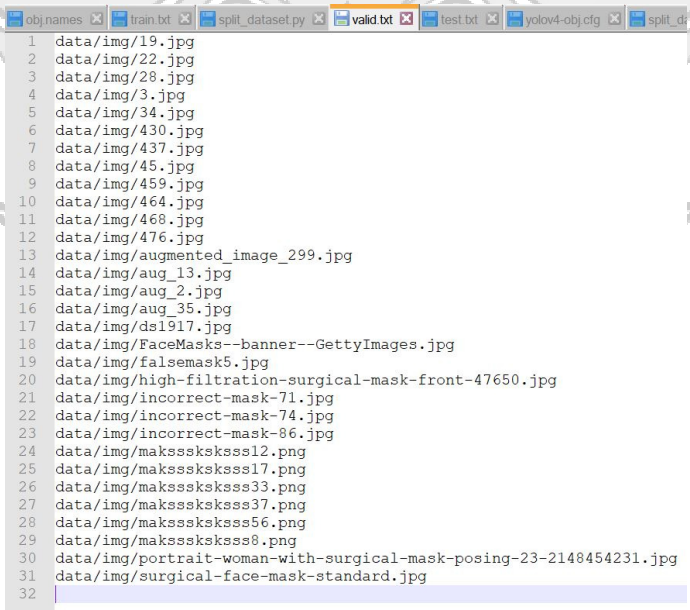
Gambar 4. 16 Overwrite text file

Setelah penghitungan jumlah data dan pembagian data berdasarkan persentase dari masing-masing *set* maka selanjutnya adalah pembuatan 3 file yaitu *train, valid,* dan *test* yang berekstensi .txt dan masing-masing file tersebut berisi

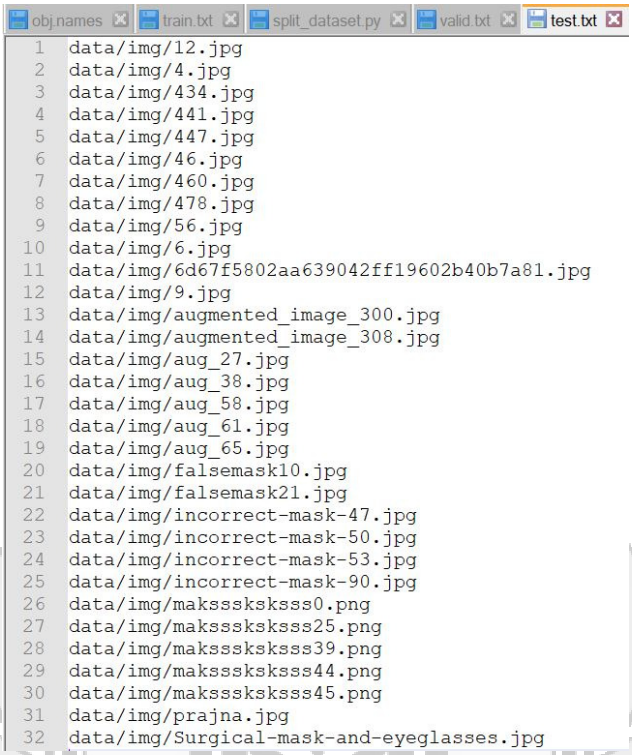
identitas data gambar yang masuk pada masing-masing *set* yaitu direktori mana data gambar tersebut tersimpan beserta nama filenya.



Gambar 4. 17 Isi file training



Gambar 4. 18 Isi file validation



Gambar 4. 19 Isi file testing

Pada Gambar 4.17 – Gambar 4.19 merupakan isi dari file masing-masing *set training, validation,* dan *testing* yaitu lokasi tersimpannya data-data gambar beserta nama file dan ekstensi data-data gambar tersebut. *Set training* berisi 252 data gambar, *set validation* berisi 31 gambar, dan *set testing* berisi 32 gambar yang dipilih secara random oleh komputer. Hal ini menunjukkan bahwa proses pembagian dataset berdasarkan persentase yang telah ditentukan telah berhasil dilakukan.

4.2.3 Training Model

Sebelum melakukan proses *training* terlebih dahulu melakukan konfigurasi untuk menyesuaikan nilai *hyperparameter* seperti pada Tabel 4.1 berikut ini.

Tabel 4. 1 Konfigurasi *Hyperparameter*

Jenis Konfigurasi	Keterangan
Batch	64
Subdivisions	64
Max Batches	6000
Steps	4800, 5400
Filters	24
Class	3

Penjelasan mengenai konfigurasi *Hyperparameter* dijelaskan sebagai berikut.

1) *Batch* dan *Subdivisions* dapat disesuaikan dengan spesifikasi GPU dengan VRAM yang digunakan.
2) *Max_batches* merupakan batas iterasi dalam melakukan *training*. Ketika iterasi sudah mencapai 6000, maka proses *training* akan otomatis berhenti. Ini didapatkan dengan cara persamaan (1) yaitu :

$$Max\ batches = Jumlah\ class\ x\ 2000$$
$$= 3\ x\ 2000$$
$$= 6000$$

3) *Steps* merupakan hasil 80% dan 90% dari *max_batches.*
4) *Filters* merupakan jumlah filter yang didapatkan dari persamaan (2) yaitu :

$$Filters = (Jumlah\ class + 5)\ x\ 3$$
$$= (3+5)\ x\ 3$$
$$= 24$$

Proses training dilakukan sebanyak 2 kali yaitu dataset pertama jumlah dataset untuk *class bad_mask* tidak sebanding jumlahnya dengan *class with_mask* dan *without_mas* dikarenakan dataset yang pertama jumlah data dari masing-masing *class* berbeda yaitu jumlah *bad_mask* tidak sebanding dengan jumlah data untuk *class with_mask dan without_mask* hal ini dikarenakan pada dataset tersebut banyak gambar yang dalam 1 *frame* terdiri dari beberapa *class* yaitu orang yang menggunakan masker, tidak menggunakan masker, dan menggunakan masker secara keliru sehingga dalam dataset tersebut hanya sedikit gambar yang dianotasi untuk *class bad_mask*. Sedangkan untuk dataset yang kedua data gambar yang akan di-*training* merupakan gambar yang dalam 1 *frame* hanya terdapat 1 *class* sehingga jumlah data dari masing-masing *class* hampir sama jumlahnya yaitu untuk *class with_mask* sebanyak 105 gambar, *without_mask* sebanyak 104 gambar, dan *bad_mask* sebanyak 107 gambar.

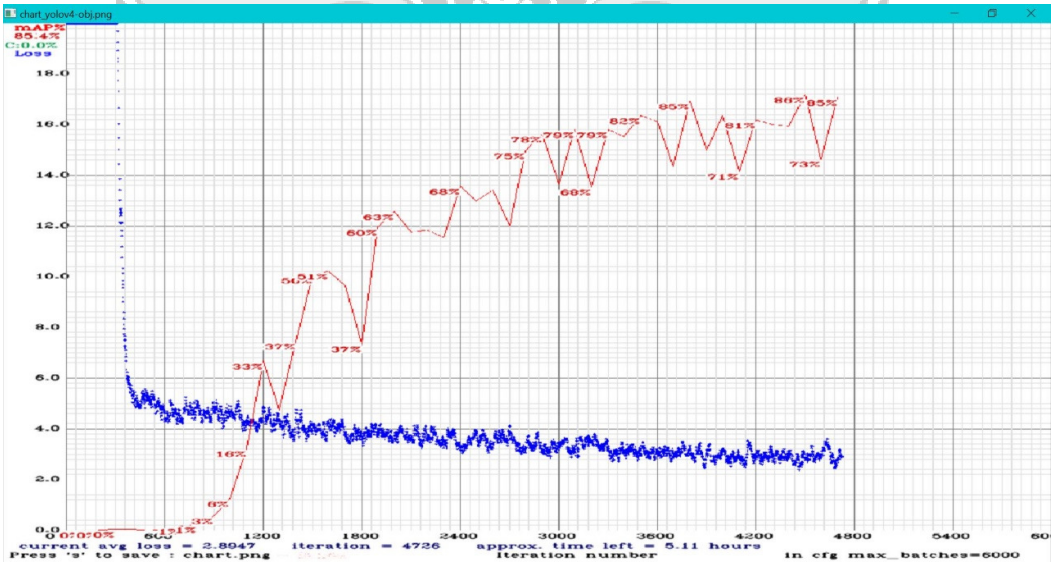
Proses *Training* dilakukan menggunakan *command prompt* dengan memasukkan folder hasil *build* YOLOv4, kemudian akan dibuat sebuah folder baru untuk menyimpan hasil training (*weights*).

```
D:\darknet-master-gpu>darknet.exe detector train data/obj.data cfg/yolov4-obj.cfg yolov4.conv.137 -map
 CUDA-version: 11030 (11060), cuDNN: 8.2.0, GPU count: 1
 OpenCV version: 4.5.2
 Prepare additional network for mAP calculation...
 0 : compute_capability = 610, cudnn_half = 0, GPU: NVIDIA GeForce MX250
net.optimized_memory = 0
mini_batch = 1, batch = 64, time_steps = 1, train = 0
   layer   filters  size/strd(dil)      input                output
   0 Create CUDA-stream - 0
 Create cudnn-handle 0
```

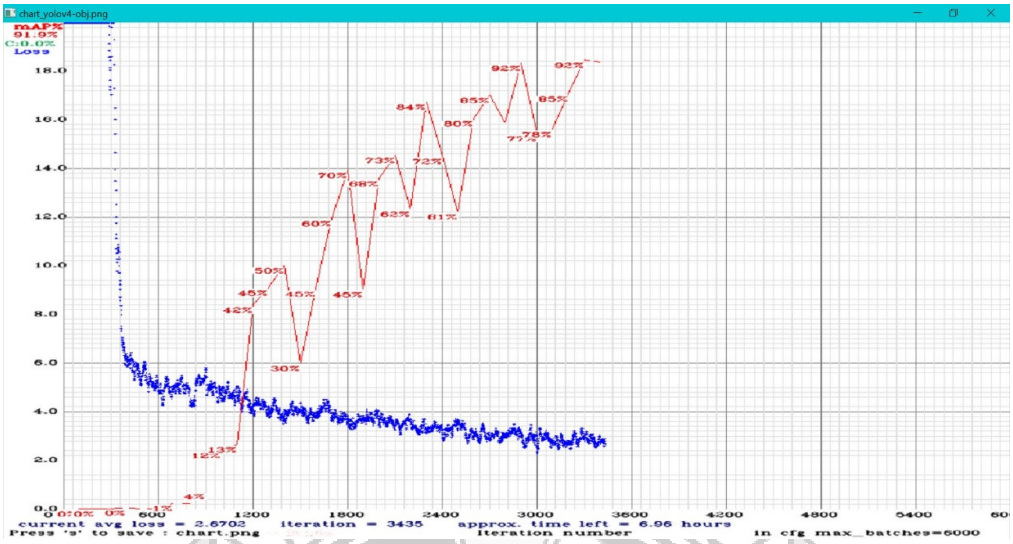
Gambar 4. 20 Menyimpan weights hasil training

Syntax pada Gambar 4.20 menunjukkan perintah untuk melakukan *training model* dari dataset yang telah dianotasi dan berikut penjelasan mengenai perintah tersebut.

1) *Darknet.exe* merupakan file *executable* YOLOv4.
2) *Detector train* merupakan perintah melakukan *training* model.
3) *Data/obj.data* merupakan file yang berisi tentang jumlah *class* dan *path* dataset. File ini berekstensi .data.
4) *Cfg/yolov4-obj.cfg merupakan file* konfigurasi yang berisi arsitektur dan *hyperparameter* YOLOv4 yang berekstensi file .cfg.
5) *Yolov4.conv.137* merupakan file *weights* yang digunakan sebagai *pretrained weights* untuk input *transfer learning.*



Gambar 4. 21 Grafik Training Dataset Pertama



Gambar 4. 22 Grafik Training Dataset Kedua

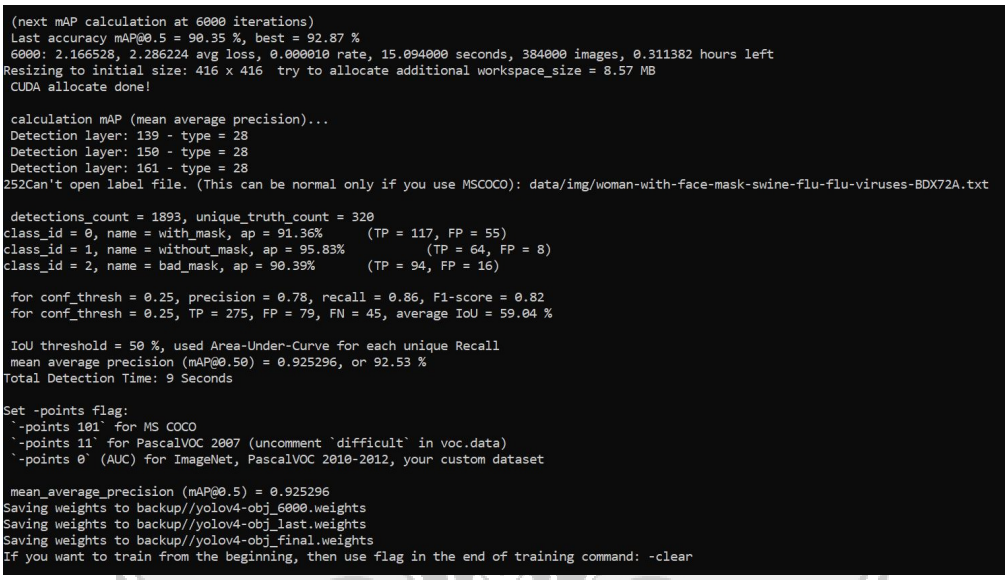
Setelah mengeksekusi perintah *training* maka akan muncul *chart* yang menunjukkan progress dari proses *training* beserta nilai *loss* dan grafik nilai mAP dalam setiap epoch dengan kelipatan 100. Pada Gambar 4.21 dan Gambar 4.22 hasil *training* untuk dataset kedua lebih besar nilai akurasinya dibandingkan dataset yang kedua sehingga dataset yang kedua yang akan digunakan untuk proses pengujian. Hasil *training* akan tersimpan pada *file weights*.

4.3 Pengujian

4.3.1 Pengujian Pada Performansi Model YOLOv4

Pengujian dilakukan bertujuan untuk mengetahui tingkat keakurasian dari model *weights* YOLOv4 yang sudah melalui proses *training.* Pengujian performa *training* model YOLOv4 digunakan sebanyak 316 data gambar dimana data *training* sebanyak 252 gambar dan data *validation* sebanyak 31 gambar yang menggunakan 3 kelas yaitu *with_mask, without_mask,* dan *bad_mask.* Data *validation* yang sudah dianotasi kemudian diolah sebagai ground-truth box

dibandingkan dan predicted box yang kemudian menghasilkan confusion matrix, kemudian dikalkulasi untuk mendapatkan nilai *F1-score, precision, recall*, *average precision* (AP), dan *mean average precision.*



Gambar 4. 23 Hasil Training Data

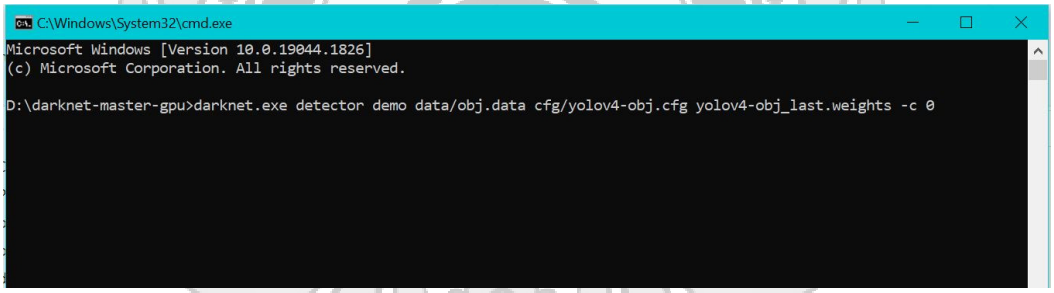
Dari proses *training* telah dilakukan yang merupakan *validation* dapat dilihat pada Gambar 4. 22 bahwa class with_mask menunjukkan jumlah TP *(True Positive)* adalah 117 dan FP *(False Positive)* sebesar 55 dengan nilai AP *(Average Precision)* adalah 91,36%. Pada tabel class without_mask menunjukkan jumlah *True Positive* sebesar 64 dan *False Positive* sebesar 8 dengan nilai *Average Precision* yaitu 95,83%. Terakhir, pada class bad_mask menunjukkan jumlah *True Positive* sebesar 94 dan *False Positive* sebesar 16 dengan *Average Precision* adalah 90,39%.

Dari hasil *training* yang telah dilakukan jumlah *True Positive* (TP) terlihat jauh lebih besar dibandingkan dengan jumlah *False Positive* (FP) *dan False Negative* (FN) hal ini menunjukkan bahwa sistem telah dapat mendeteksi objek dengan benar

dari dataset yang telah dilatih. Nilai presisi sistem mengklasifikasikan objek dengan tepat pada hasil *training* data adalah 78%, kemudian *recall* mengukur kemampuan model untuk menemukan seluruh objek positif mencapai nilai 86% dengan mAP sebesar 92,53%.

4.3.2 Pengujian Pada Hasil Deteksi Objek

Setelah itu deteksi objek dengan menggunakan input *video-stream* dimana menggunakan hasil data yang telah di-*training* yang berupa file *weights*. Untuk melakukan deteksi objek terdapat pada Gambar 4. 23 dengan menunjukkan perintah berikut ini.



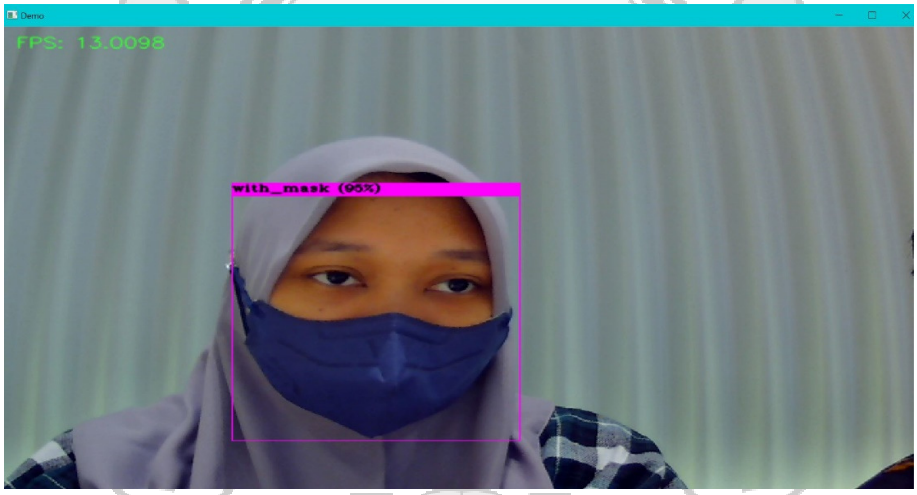
Gambar 4. 24 Perintah untuk mendeteksi objek dengan input *video-stream*

Penjelasan dari perintah untuk mendeteksi objek adalah sebagai berikut.

1) *Darknet.exe* merupakan file executable YOLOv4
2) *Detector demo* merupakan perintah untuk melakukan deteksi objek pada *video*
3) *data/obj.data* merupakan file yang berisi tentang jumlah *class* yang telah diatur sebelumnya
4) *Cfg/yolov4-obj.cfg* merupakan file konfigurasi *Hyperparameter* YOLOv4

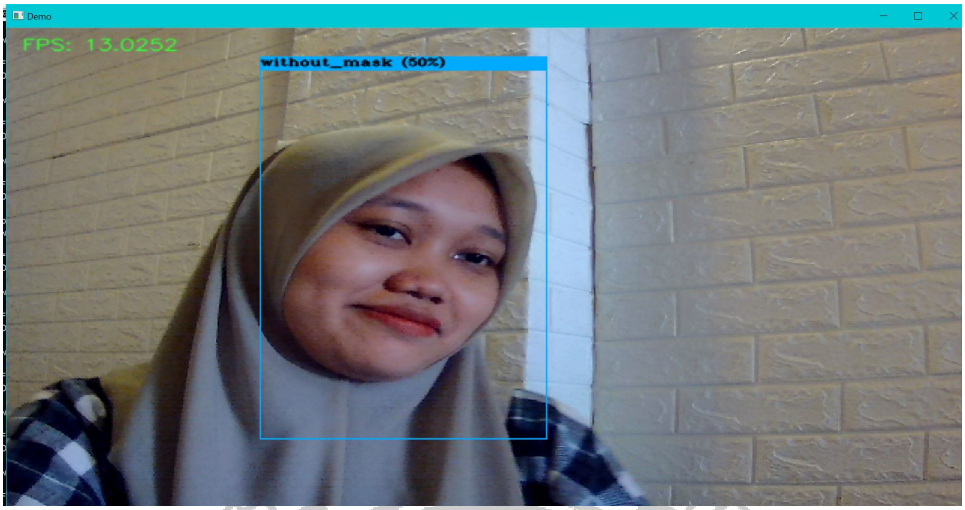
5) *Yolov4-obj_last.weights* merupakan file *weights* hasil training YOLOv4 terhadap dataset masker dengan 3 kategori *class* yang telah dilakukan.
6) *-c 0* merupakan perintah untuk menyatakan input video dari kamera yang digunakan. Jika menggunakan kamera internal maka menggunakan perintah *-c 0* dan jika menggunakan kamera eksternal maka menggunakan perintah *-c 1*.

Setelah melakukan perintah yang telah dijelaskan sebelumnya maka mendeteksi objek telah dilakukan seperti pada Gambar 4. 24 – Gambar 4. 27 di bawah ini.



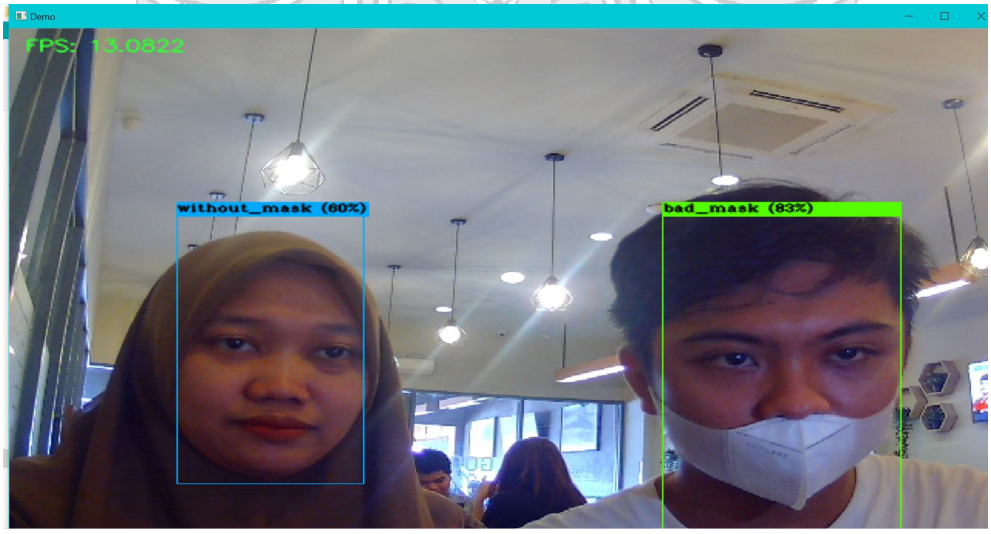
Gambar 4. 25 Hasil Mendeteksi Wajah dengan class with_mask

Pada Gambar 4.24 menunjukkan hasil mendeteksi objek wajah dengan menggunakan masker yang ditandai dengan *bounding box* dengan *class with_mask* yang mengelilingi objek yang terdeteksi sebagai wajah menggunakan masker.



Gambar 4. 26 Hasil Mendeteksi Wajah dengan class with_mask

Gambar 4.25 menunjukkan bahwa hasil mendeteksi objek wajah yang tidak menggunakan masker yang ditandai dengan *bounding box* dengan *class without_mask* mengelilingi objek yang terdeteksi sebagai wajah tidak menggunakan masker.



Gambar 4. 27 Hasil Mendeteksi Wajah dengan 2 class sekaligus

Begitupun dengan Gambar 4.26 yang merupakan hasil dari mendeteksi objek wajah dimana dalam Gambar 4.26 terdapat 2 wajah yaitu wajah yang terdeteksi

tidak menggunakan masker dan wajah menggunakan masker namun cara penggunaannya salah yang ditandai dengan *bounding box* dengan *class* without_mask dan bad_mask.

Gambar 4. 28 Hasil Mendeteksi Wajah dengan Class bad_mask

Pada Gambar 4. 27 merupakan hasil mendeteksi objek yang hanya menutupi wajahnya dengan menggunakan buku dimana kondisi tersebut masuk pada *class* bad_mask. Hasil mendeteksi wajah yang telah ditunjukkan pada Gambar 4.24 – 4.27 dengan *3 class* yaitu *class* with_mask, without_mask, dan bad_mask. Deteksi objek wajah bermasker ini berpengaruh terhadap kualitas kamera, cahaya, dan penempatan objek wajah pada kamera.

4.3.3 Pengujian Hasil Deteksi di Puskesmas Sudiang Raya

Proses pengujian dilakukan di Puskesmas Sudiang Raya dengan menggunakan kamera webcam eksternal sebagai kamera CCTV yang inputnya berupa *video-stream*. Pengujian di Puskesmas Sudiang Raya ini dilakukan pada salah satu pintu untuk masuk ke dalam Puskesmas Sudiang Raya. Pengunjung yang ingin masuk ke

dalam Puskesmas Sudiang Raya harus melewati sistem ini untuk mengecek apakah pengunjung tersebut menggunakan masker atau tidak ataupun menggunakan masker namun cara penggunaannya salah.

Pengujian sistem pada penelitian ini dilaksanakan selama dua hari dengan skenario seperti pada Tabel 4.2 berikut ini.

Tabel 4. 2 Hasil Evaluasi Pengujian

Hari ke-	**Class**	**Data Aktual**	**Data Sistem (Prediksi)**	**Jumlah Pengunjung**
1	With_mask	28	28	32 orang
	Without_mask	3	3	
	Bad_mask	1	1	
2	With_mask	21	22	24 orang
	Without_mask	3	2	
	Bad_mask	0	0	

Dari Tabel 4.2 yang ditunjukkan pada penelitian ini telah dilakukan selama 2 hari dimana hari pertama terdapat pengunjung sebanyak 32 orang dimana pengunjung tersebut menggunakan masker dengan benar sebanyak 28 orang dam yang terdeteksi oleh sistem sebanyak 28 orang, pengunjung yang tidak menggunakan masker sebanyak 3 orang dan yang terdeteksi oleh sistem sebanyak 3 orang, dan yang menggunakan masker secara keliru sebanyak 1 orang dan terdeteksi 1 orang oleh sistem.

Kemudian pengujian pada hari kedua terdapat pengunjung sebanyak 24 orang dimana terdapat orang yang menggunakan masker dengan benar sebanyak 21 orang namun yang terdeteksi oleh sistem sebanyak 22 orang, kemudian pengunjung yang tidak menggunakan masker sebanyak 3 orang namun yang terdeteksi tidak menggunakan masker sebanyak 2 orang, dan tidak ada pengunjung yang menggunakan masker dengan cara yang salah.

Untuk menentukan rasio prediksi pengujian yang telah dilakukan oleh system di Puskesmas Sudiang Raya dengan cara sebagai berikut.

Tabel 4. 3 Menentukan Confusion Matrix

		SECARA SISTEM		
		With_mask	**Without_mask**	**Bad_mask**
SECARA AKTUAL	**With_mask**	49	1	0
	Without_mask	0	5	0
	Bad_mask	0	0	1

TP (*True Positive*) = 49 FN (*False Negative*) = 1

FP (*False Positive*) = 0 TN (*True Negative*) = 50

Setelah itu menentukan nilai akurasi dengan menggunakan *Persamaan 5* yaitu sebagai berikut.

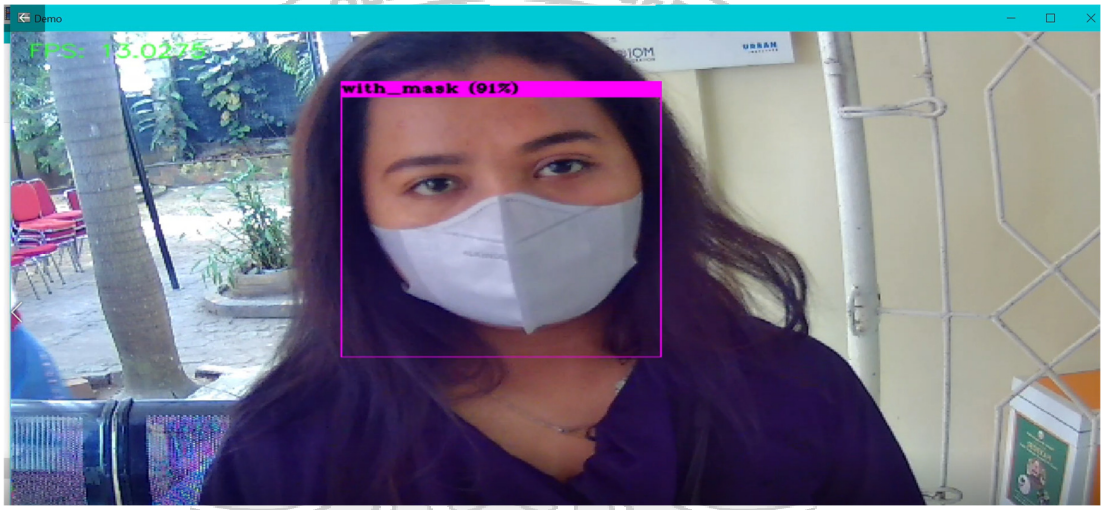
$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

$$= \frac{49+50}{49+50+0+1}$$

$$= 99\%$$

Akurasi yang didapatkan pada hasil pengujian system pendeteksi masker yang telah dilakukan sebesar 99%.

Berikut gambar pengunjung dari Puskesmas Sudiang Raya dalam beberapa kondisi.

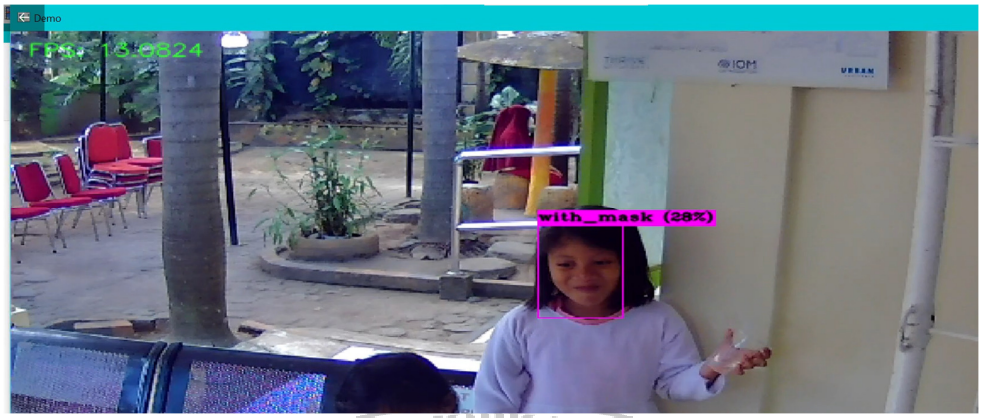


Gambar 4. 29 Pengunjung yang terdeteksi menggunakan masker

Gambar 4. 30 Pengunjung yang terdeteksi menggunakan masker dari jarak jauh

Gambar 4. 31 Pengunjung yang terdeteksi menggunakan masker secara keliru dan anak-anak yang tidak terdeteksi oleh sistem

Gambar 4. 32 Pengunjung yang terdeteksi tidak menggunakan masker



Gambar 4. 33 Pengunjung yang tidak menggunakan masker namun terdeteksi menggunakan masker

BAB V PENUTUP

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan mengenai pendeteksi objek wajah yang menggunakan masker, tidak menggunakan masker, dan penggunaan masker yang salah yang diimplementasikan di Puskesmas Sudiang Raya, maka dapat ditarik kesimpulan sebagai berikut.

1. Sistem pendeteksi objek wajah penggunaan masker dengan metode *Convolutional Neural Network (CNN)* menggunakan model YOLOv4 (*You Only Look Once Version 4)* mampu mendeteksi wajah yang menggunakan masker, wajah yang tidak menggunakan masker, dan wajah yang menggunakan masker namun penggunaan masker yang keliru. Berdasarkan 3 *class* data yang telah melewati proses *training* mampu mencapai nilai mAP *(mean Average Precision)* sebesar 92,53%.
2. Implementasi sistem pendeteksi objek wajah penggunaan masker di Puskesmas Sudiang Raya mampu mendeteksi pengunjung yang menggunakan masker, tidak menggunakan masker, dan menggunakan masker namun penggunaan masker yang keliru. Hasil akurasi yang didapatkan adalah 99%.

5.2 Saran

Berdasarkan penelitian yang telah dilakukan, penulis memberikan saran untuk penelitian di masa depan, antara lain :

1. Menambah data pada setiap kelas yang akan digunakan untuk *training* agar memaksimalkan tingkat akurasi model *weights.*

2. Menambah peringatan untuk pengunjung yang tidak menggunakan masker dan pengunjung yang menggunakan masker secara keliru misalnya dapar berupa *buzzer.*

DAFTAR PUSTAKA

Abdul, M., Irham, R., & Prasetya, D. A. (2020). Prototipe Pendeteksi Masker Pada Ruangan Wajib Masker Untuk Kendali Pintu Otomatis Berbasis Deep Learning Sebagai Pencegahan Penularan Covid-19. *Prototipe Pendeteksi Masker Pada Ruangan Wajib Masker Untuk Kendali Pintu Otomatis Berbasis Deep Learning Sebagai Pencegahan Penularan Covid-19*, 47–55. https://publikasiilmiah.ums.ac.id/xmlui/bitstream/handle/11617/12377/108.pdf?sequence=1&isAllowed=y

Alamsyah, D., & Pratama, D. (2020). Implementasi Convolutional Neural Networks (CNN) untuk Klasifikasi Ekspresi Citra Wajah pada FER-2013 Dataset. *JurTI (Jurnal Teknologi Informasi)*, *4*(2), 350–355.

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). *YOLOv4: Optimal Speed and Accuracy of Object Detection*. http://arxiv.org/abs/2004.10934

Darmanto, H. (2019). Pengenalan Spesies Ikan Berdasarkan Kontur Otolith Menggunakan Convolutional Neural Network. *Joined Journal (Journal of Informatics Education)*, *2*(1), 41. https://doi.org/10.31331/joined.v2i1.847

Demak Kominfo. (2020, October 7). *Masih Ada Masyarakat yang Sepelekan Penggunaan Masker - Pemerintah Provinsi Jawa Tengah*. https://jatengprov.go.id/beritadaerah/masih-ada-masyarakat-yang-sepelekan-penggunaan-masker/

Eural, C. O. N., & Cnn, N. E. (2021). *PREDIKSI KARAKTERISTIK PERSONAL MENGGUNAKAN ANALISIS TANDA TANGAN DENGAN*

MENGGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK (CNN). *15*(1), 123–133.

Fahlevi, F. (2021, July 12). *LaporCovid-19 Ungkap Banyak Warga Enggan ke Rumah Sakit Karena Takut "Dicovidkan" - Tribunnews.com*. https://www.tribunnews.com/corona/2021/07/12/laporcovid-19-ungkap-banyak-warga-enggan-ke-rumah-sakit-karena-takut-dicovidkan

Handono, S. F., Anggraeny, F. T., & Rahmat, B. (2020). Implementasi Convolutional Neural Networks (CNN) untuk Deteksi Retinopati Diabetik. *Jurnal Informatika Dan Sistem Informasi (JIFoSI)*, *1*(1), 669–678.

Hidayatullah, P. (2021). *Buku Sakti Deep Learning*. Academy, Stunning Vision AI.

Ilahiyah, S., & Nilogiri, A. (2018). Implementasi Deep Learning Pada Identifikasi Jenis Tumbuhan Berdasarkan Citra Daun Menggunakan Convolutional Neural Network. *JUSTINDO (Jurnal Sistem Dan Teknologi Informasi Indonesia)*, *3*(2), 49–56.

Kenda, P. (2021). Sistem Presensi Berbasis Wajah Dengan Metode Haar Cascade. *KONSTELASI: Konvergensi Teknologi Dan Sistem Informasi*, *1*(2), 419–429.

Konar, J., Khandelwal, P., & Tripathi, R. (2020). Comparison of Various Learning Rate Scheduling Techniques on Convolutional Neural Network. *2020 IEEE International Students' Conference on Electrical, Electronics and Computer Science, SCEECS 2020*. https://doi.org/10.1109/SCEECS48394.2020.94

Kusuma, T. A. A. H., Usman, K., & Saidah, S. (2021). People Counting for Public

Transportations Using You Only Look Once Method. *Jurnal Teknik Informatika (Jutif)*, *2*(1), 57–66. https://doi.org/10.20884/1.jutif.2021.2.2.77

Ninggar, A. (2021, December 18). *VARIAN Baru Virus Corona, OMICRON Masuk Indonesia, Berikut Gejala dan Cara Cegah Penularannya - Tribunnews.com*. https://www.tribunnews.com/corona/2021/12/18/varian-baru-virus-corona-omicron-masuk-indonesia-berikut-gejala-dan-cara-cegah-penularannya

Noor, M., Isya, R., Endrasmono, J., & Khumaidi, A. (2020). *Rancang Bangun Sistem Peringatan Identifikasi Alat Pelindung Diri (APD) Menggunakan Metode You Only Look Once v4 (YOLOv4)*. *4*(2809), 68–72.

Nugroho, K. S. (2020). *Confusion Matrix untuk Evaluasi Model pada Supervised Learning | by Kuncahyo Setyo Nugroho | Medium*. https://ksnugroho.medium.com/confusion-matrix-untuk-evaluasi-model-pada-unsupervised-machine-learning-bc4b1ae9ae3f

Nugroho, P. A., Fenriana, I., & Arijanto, R. (2020). Implementasi Deep Learning Menggunakan Convolutional Neural Network (CNN) Pada Ekspresi Manusia. *Algor*, *2*, 12–21.

Nurfita, R. D., & Ariyanto, G. (2018). Implementasi Deep Learning Berbasis Tensorflow Untuk Pengenalan Sidik Jari. *Emitor: Jurnal Teknik Elektro*, *18*(01), 22–27. https://doi.org/10.23917/emitor.v18i01.6236

Pelletier, C., Webb, G. I., & Petitjean, F. (2019). Temporal convolutional neural network for the classification of satellite image time series. *Remote Sensing*,

11(5). https://doi.org/10.3390/rs11050523

Prayogo, H., Armanto, & Kristian, D. Y. (2020). Klasifikasi Format Visual Film (Live Action , 3D Animation , 2D Animation) Memanfaatkan Convolutional Neural Network. *Artikel Ilmiah Tugas Akhir Dan Tesis Institut Sains Dan Teknologi Terapan Surabaya*, 1–8.

Qudsi, N. K., Asmara, R. A., & Syulistyo, A. R. (2020). Identifikasi Citra Tulisan Tangan Digital Menggunakan Convolutional Neural Network (CNN). *Seminar Informatika Aplikatif Polinema*, 48–53.

Rachmawati, F., & Widhyaestoeti, D. (2020). Deteksi Jumlah Kendaraan di Jalur SSA Kota Bogor Menggunakan Algoritma Deep Learning YOLO. *Prosiding LPPM UIKA Bogor*, 360–370.

Radiuk, P. M. (2018). Impact of Training Set Batch Size on the Performance of Convolutional Neural Networks for Diverse Datasets. *Information Technology and Management Science*, *20*(1), 20–24. https://doi.org/10.1515/itms-2017-0003

Rahman, A. A., Agustin, S. D., & Ibrahim, N. U. R. (2022). *Perbandingan Algoritma YOLOv4 dan Scaled YOLOv4 untuk Deteksi Objek pada Citra Termal*. *7*(1), 61–71.

Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., & Savarese, S. (2019). Generalized intersection over union: A metric and a loss for bounding box regression. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2019-June*, 658–666.

https://doi.org/10.1109/CVPR.2019.00075

Rokom. (2021, January 19). *Jubir COVID-19 : Kurang Disiplin Pakai Masker Jadi Faktor Utama Kenaikan Kasus – Sehat Negeriku*. https://sehatnegeriku.kemkes.go.id/baca/umum/20200711/2734436/jubir-covid-19-kurang-disiplin-pakai-masker-jadi-faktor-utama-kenaikan-kasus/

Rosebrock, A. (2017). Deep Learning for Computer Vision with Python. *Deep Learning for Computer Vision with Python*, 3.

Santoso, A., & Ariyanto, G. (2018). Implementasi Deep Learning Berbasis Keras Untuk Pengenalan Wajah. *Emitor: Jurnal Teknik Elektro*, *18*(01), 15–21. https://doi.org/10.23917/emitor.v18i01.6235

Septiana, T., Puspita, N., Fikih, M. Al, & Setyawan, N. (2020). Face Mask Detection Covid-19 Using Convolutional Neural Network (Cnn). *Seminar Nasional Teknologi Dan Rekayasa (SENTRA) 2020*, 27–32.

Sudhir Bussa, Ananya Mani, Shruti Bharuka, & Sakshi Kaushik. (2020). Smart Attendance System using OPENCV based on Facial Recognition. *International Journal of Engineering Research And*, *V9*(03), 54–59. https://doi.org/10.17577/ijertv9is030122

Tomás, J., Rego, A., Viciano-Tudela, S., & Lloret, J. (2021). Incorrect facemask-wearing detection using convolutional neural networks with transfer learning. *Healthcare (Switzerland)*, *9*(8). https://doi.org/10.3390/healthcare9081050

Wang, S., Zhao, J., Ta, N., Zhao, X., Xiao, M., & Wei, H. (2021). A real-time deep learning forest fire monitoring algorithm based on an improved Pruned + KD model. *Journal of Real-Time Image Processing*, *18*(6), 2319–2329. https://doi.org/10.1007/s11554-021-01124-9

Wu, Y., Liu, L., Bae, J., Chow, K. H., Iyengar, A., Pu, C., Wei, W., Yu, L., & Zhang, Q. (2019). Demystifying Learning Rate Policies for High Accuracy Training of Deep Neural Networks. *Proceedings - 2019 IEEE International Conference on Big Data, Big Data 2019*, *December*, 1971–1980. https://doi.org/10.1109/BigData47090.2019.9006104

Zhang, G. L., Ge, L. L., Yang, Y. N., Liu, Y. Q., & Sun, K. X. (2019). Fused Confidence for Scene Text Detection via Intersection-over-Union. *International Conference on Communication Technology Proceedings, ICCT*, 1540–1543. https://doi.org/10.1109/ICCT46805.2019.8947307

Zhu, Z., & Cheng, Y. (2020). Application of attitude tracking algorithm for face recognition based on OpenCV in the intelligent door lock. *Computer Communications*, *154*(900), 390–397. https://doi.org/10.1016/j.comcom.2020.02.003

LAMPIRAN

INSTALASI YOLOv4

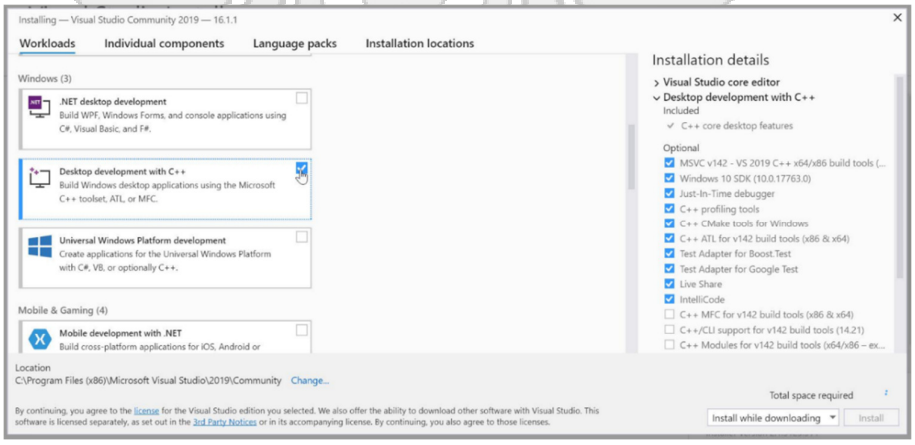
Sebelum melakukan instalasi YOLOv4 terlebih dahulu menginstall *tools* yang menjadi *requirement* untuk instalasi YOLOv4 yang menggunakan mode GPU.

1. **Instalasi Visual Studio**

   a. Sebelum menginstall Visual Studio terlebih dahulu menginstall Microsoft Visual C++

   b. Setelah Microsoft Visual C++ terinstall selanjutnya adalah menginstall Visual Studio 2019 di website https://visualstudio.microsoft.com/vs/. Setelah terdownload *double click file installer*. Pastikan pada saat menginstall Visual Studio 2019 laptop/computer terhubung dengan koneksi internet agar file-file yang dibutuhkan terdownload.

   c. Pilih *workloads* "Desktop Development with C++" kemudian klik Install.



   d. Setelah itu proses download dan proses instalasi *workloads* sesuai yang dipilih. Tunggu beberapa saat.

e. Setelah proses download selesai, akan diminta *sign in* namun jika tidak memiliki akun dapat mengklik *Not now, maybe later.*

2. Instalasi Python

a. Download *installer* Python pada website https://www.python.org/downloads/windows. Pilih installer yang sesuai dengan arsitektur laptop/computer yang digunakan baik itu 32 bit atau 64 bit.

b. Setelah didownload, *double click file installer Python.*

c. Kemudian ceklis bagian *Installer launcher for all users* dan *Add Python to Path* kemudian klik *Install Now*

d. Setelah itu tunggu proses instalasi selesai kemudian cek di cmd dengan perintah *python --version*

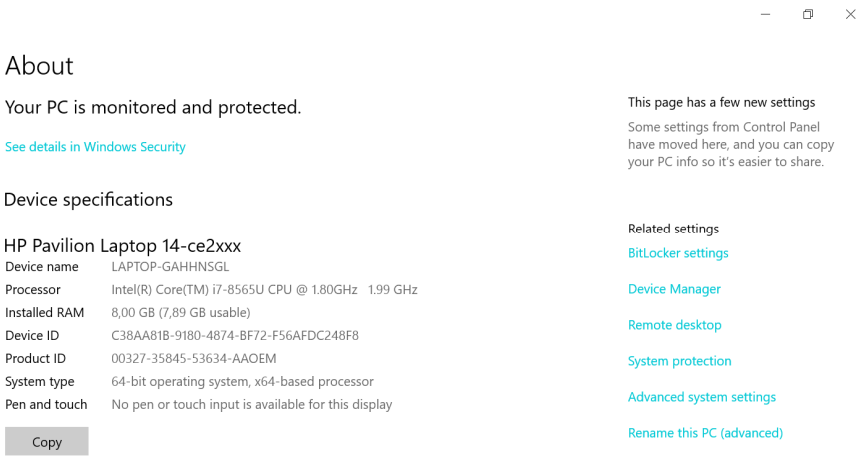
2. **Instalasi OpenCV**

a. *Installer OpenCV* dapat didownload pada website https://opencv.org/releases/

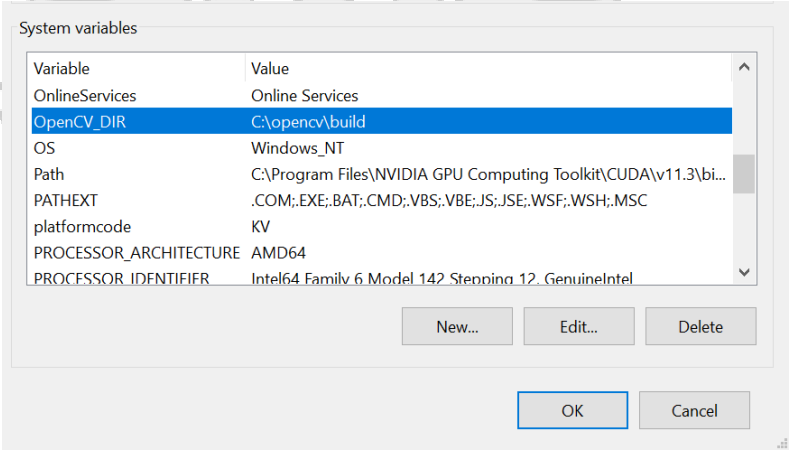
b. Setelah berhasil didownload *double click file installer* yang telah berhasil didownload untuk melakukan penginstalan. Kemudian masukkan *directory* C:/ untuk menyimpan hasil instalasi.

c. Setting *environment* sesuai langkah berikut ini :

1. Klik kanan pada *This PC* di Windows Eksplorer lalu klik Properties
2. Kemudian Klik *Advance system settings* di bagian *related settings*



3. Klik *Environment Variables,* lalu edit variable *Path* pada bagian *System Variable* dengan memilih variable *Path* setelah itu klik *Edit*
4. Klik tombol *New* untuk menambahkan *value* baru kemudian tambahkan path C:/opencv/build/x64/vc15/bin lalu klik OK
5. Tambahkan variable baru dengan cara klik *New* kemudian beri nilai OpenCV_DIR dan Pada *Variable value* beri nilai C:/opencv/build lalu klik OK



6. Jika *setting environment* sudah selesai klik OK kemudian restart laptop/computer.

3. **Instalasi CMake**

   a. Download *installer CMake* pada website https://cmake.org/download/ kemudian pilih installer yang sesuai dengan arsitektur computer baik 32 bit ataupun 64 bit.

   b. Install CMake dengan *double click file installer.*

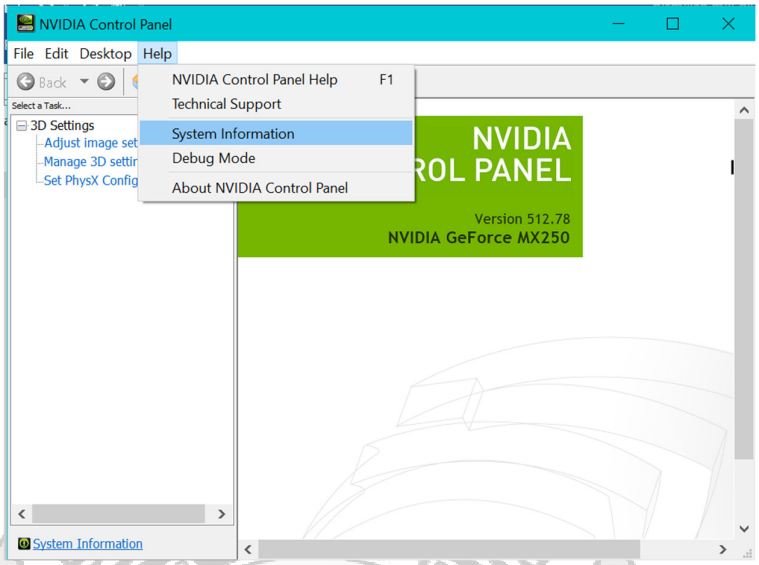
   c. Tentukan *directory* untuk menyimpan hasil instalasi. Kemudian klik untuk memulai instalasi dan tunggu proses instalasi selesai.

4. **Instalasi YOLOv4**

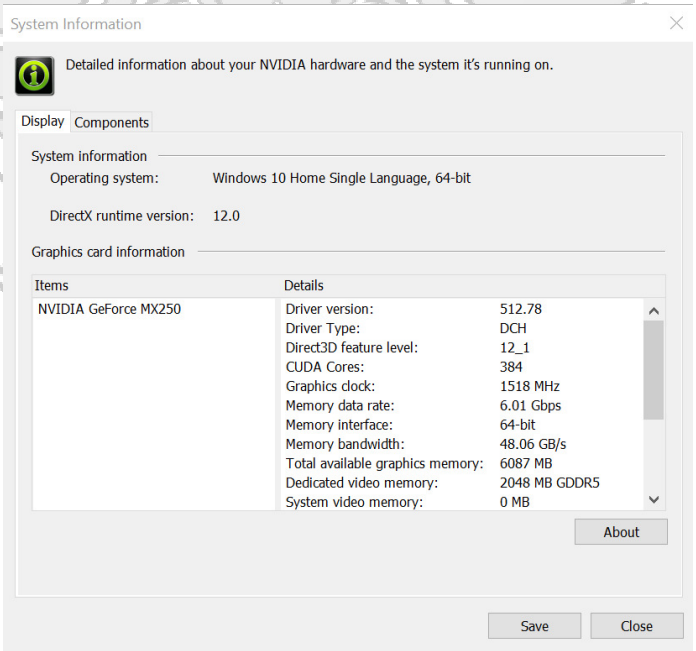
   Instalasi YOLOv4 pada Windows dengan mengaktifkan *Graphics Processing Unit* (GPU) dimana GPU yang digunakan harus berbasis NVIDIA. Penggunaan GPU pada kinerja YOLOv4 akan meningkat secara signifikan dibandingkan menggunakan CPU.

   a. Mengecek versi GPU

      1. Sebelum melakukan instalasi terlebih dahulu memeriksa versi *driver* GPU dengan cara membuka NVIDIA Control Panel, kemudian klik help lalu *System Information*.



2. Setelah itu akan muncul jenis GPU beserta detailnya. Versi driver GPU digunakan untuk menentukan versi CUDA yang akan diinstall karena jika versi tidak sesuai maka akan menyebabkan erorr pada saat proses *build.* Versi GPU yang digunakan adalah 512.78



Adapun versi driver GPU yang digunakan untuk menentukan versi CUDA yaitu sebagai berikut.

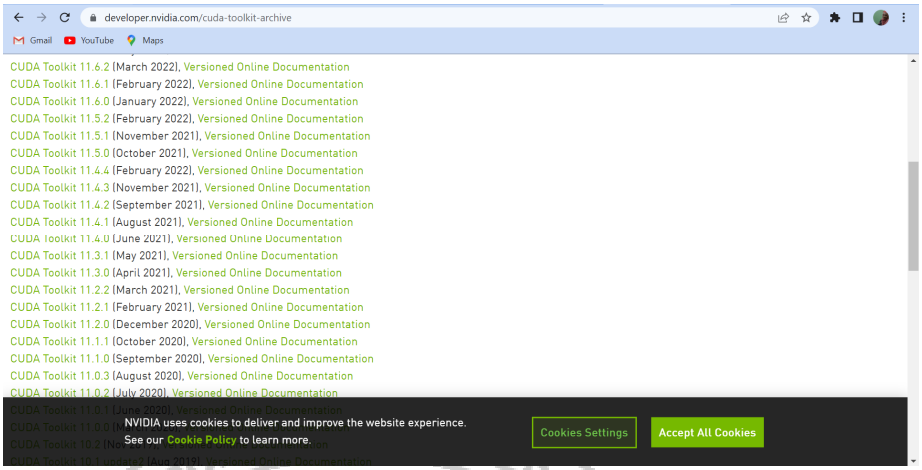
Windows GPU Driver	CUDA Version
>=465.89	CUDA 11.3.0 GA
>=461.33	CUDA 11.2.2 Update 2
>=461.09	CUDA 11.2.1 Update 1
>=460.82	CUDA 11.2.0 GA
>=456.81	CUDA 11.1.1 Update 1
>=456.38	CUDA 11.1 GA
>=451.82	CUDA 11.0.3 Update 1
>=451.48	CUDA 11.0.2 GA
>= 451.22	CUDA 11.0.1 RC
>= 441.22	CUDA 10.2.89

3. Jika *driver* belum terinstall atau versi driver GPU lebih kecil dari versi 441.22 maka driver GPU harus diinstall/update terlebih dahulu pada website https://www.nvidia.com/Download/index.aspx

b. Install CUDA

Setelah mengetahui versi CUDA yang cocok untuk versi GPU NVIDIA yang digunakan selanjutnya adalah menginstall CUDA yaitu sebagai berikut.

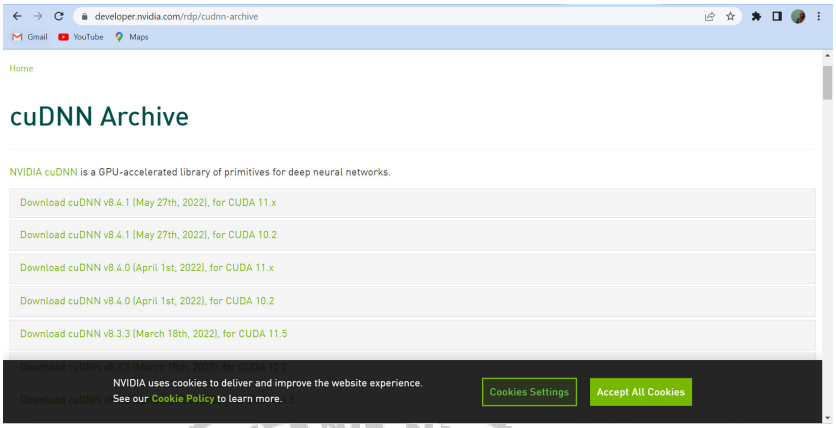
1. Download CUDA *installer* pada https://developer.nvidia.com/cuda-toolkit-archive kemudian pilih OS, versi OS, arsitektur dan tipe installer lalu download.



2. Install CUDA dengan cara *double click file installer* yang sudah didownload kemudian tunggu proses *extract*
3. Setelah itu akan terdapat pemeriksaan *system compatibility* apabila terjadi error maka salah satu penyebabnya adalah versi CUDA yang tidak cocok dengan versi GPU.
4. Setelah itu lakukan proses instalasi sampai selesai.

c. Install cuDNN

1. Download cuDNN pada website https://developer.nvidia.com/cudnn-download-survey
2. Login menggunakan akun NVIDIA atau sign up menggunakan email
3. Setelah login ceklis pada bagian *I Agree To The Terms of The cuDNN* lalu klik *Archived cuDNN Releases* untuk melihat versi cuDNN yang lengkap. Kemudian pilih versi cuDNN yang cocok dengan versi CUDA.



4. Setelah selesai mendownload extract hasil download kemudian *copy* semua file yang ada pada *<folder-hasil-extract>\cuda\bin* dan *paste* pada *C:/Program Files/NVIDIA GPU Computing Toolkit\CUDA\<versi-CUDA>\bin*
5. *Copy* semua file yang ada pada *<folder-hasil-extract>\cuda\include* dan *paste* pada *C:/Program Files/NVIDIA GPU Computing Toolkit\CUDA\<versi-CUDA>\include*
6. *Copy* semua file yang ada pada *<folder-hasil-extract>\cuda\lib\x64* dan *paste* pada *C:/Program Files/NVIDIA GPU Computing Toolkit\CUDA\<versi-CUDA>\lib\x64*

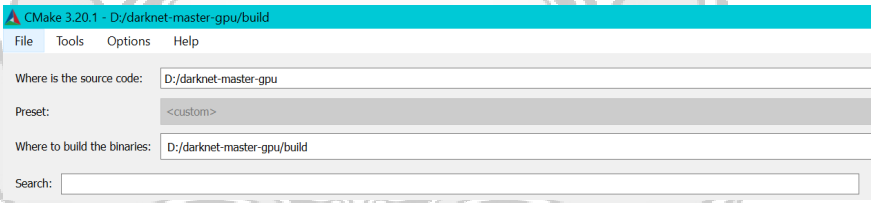
d. *Build* Darknet

1. Download darknet pada repository Github pada link https://github.com/pHidayatullah/darknet
2. Setelah terdownload extract file hasil download tersebut kemudian simpan pada directory yang diinginkan
3. Hasil extract tersebut akan tersimpan dalam folder *darknet-master* kemudian *rename* menjadi *darknet-master-gpu*

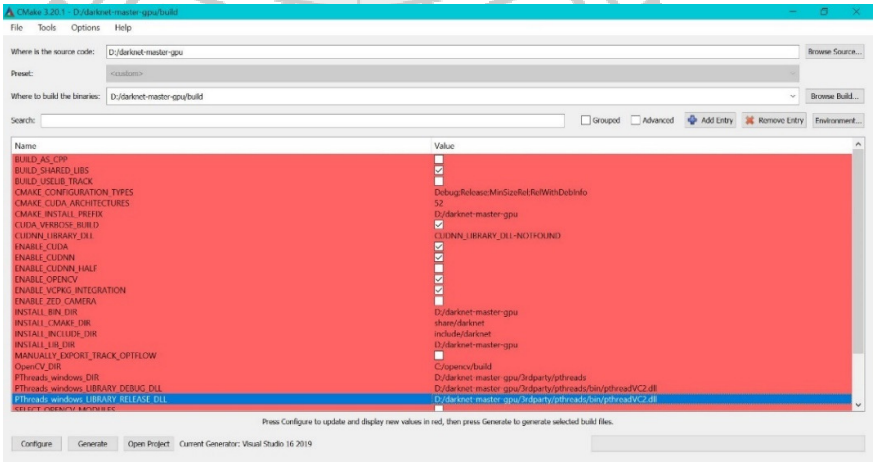
4. Masuk ke folder tersebut lalu buka file yang Bernama CMakeLists.txt lalu edit seperti pada gambar di bawah ini

```
option(CMAKE_VERBOSE_MAKEFILE "Create verbose makefile" ON)
option(CUDA_VERBOSE_BUILD "Create verbose CUDA build" ON)
option(BUILD_SHARED_LIBS "Create dark as a shared library" ON)
option(BUILD_AS_CPP "Build Darknet using C++ compiler also for C files" OFF)
option(BUILD_USELIB_TRACK "Build uselib_track" ON)
option(MANUALLY_EXPORT_TRACK_OPTFLOW "Manually export the TRACK_OPTFLOW=1 define" OFF)
option(ENABLE_OPENCV "Enable OpenCV integration" ON)
option(ENABLE_CUDA "Enable CUDA support" ON)
option(ENABLE_CUDNN "Enable CUDNN" ON)
option(ENABLE_CUDNN_HALF "Enable CUDNN Half precision" OFF)
option(ENABLE_ZED_CAMERA "Enable ZED Camera support" ON)
option(ENABLE_VCPKG_INTEGRATION "Enable VCPKG integration" ON)
option(VCPKG_BUILD_OPENCV_WITH_CUDA "Build OpenCV with CUDA extension integration" ON)
```

5. Buka CMake kemudian pada bagian *Where is the source code* masukkan folder hasil extract kemudian pada bagian *Where to build binaries* masukkan folder *buid* hasil extract kemudian klik *Configure*



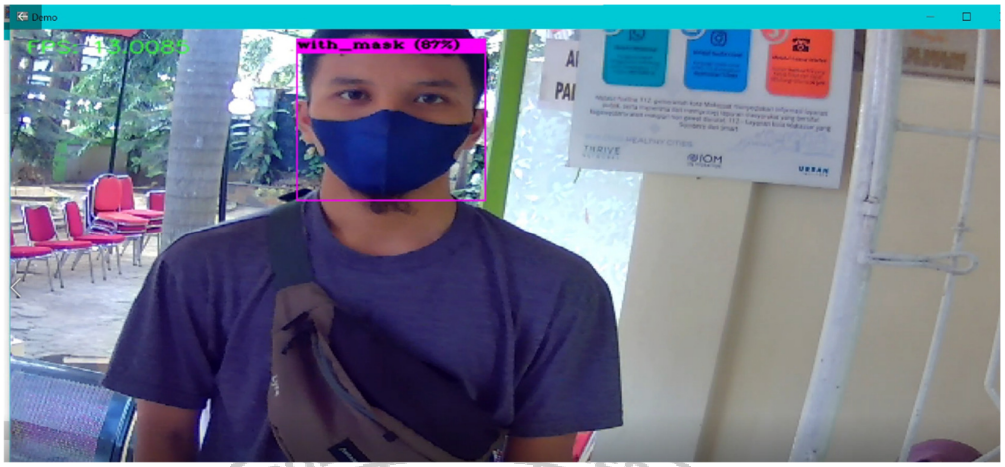
6. Kemudian pilih versi Visual Studio yang digunakan setelah itu akan muncul pesan *Configuring Done* kemudian klik *Generate* untuk membuat *project* pada Visual Studio setelah itu Klik *Open Project* untuk membuka *project* yang telah ter-*generate*

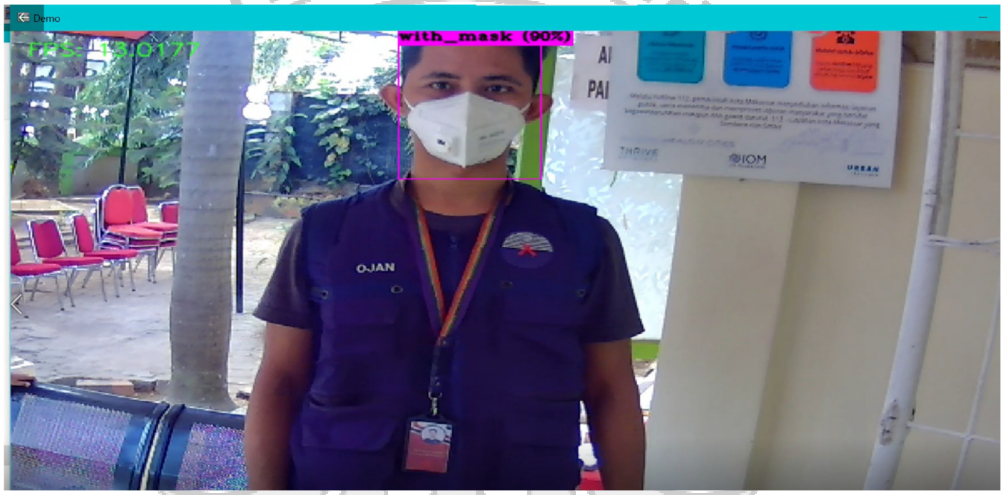


7. Ubah mode *Debug* menjadi *Release* kemudian klik *Build* lalu *Build Solution* untuk *build project* dan hasil *build* tersebut akan tersimpan dalam *D:/darknet-master-gpu/build/Release*
8. *Copy* folder data dan cfg yang ada pada folder *darknet-master-gpu* lalu paste dalam hasil *build* yaitu folder *darknet-master-gpu/build/Release*
9. *Copy file pthreadCV2.dll* dari *darknet-master-gpu/3rdparty/pthreads/bin*, lalu *paste* ke dalam folder *darknet-master-gpu/build/Release*
10. Pengecekan deteksi menggunakan model YOLOv4 sudah dapat digunakan.

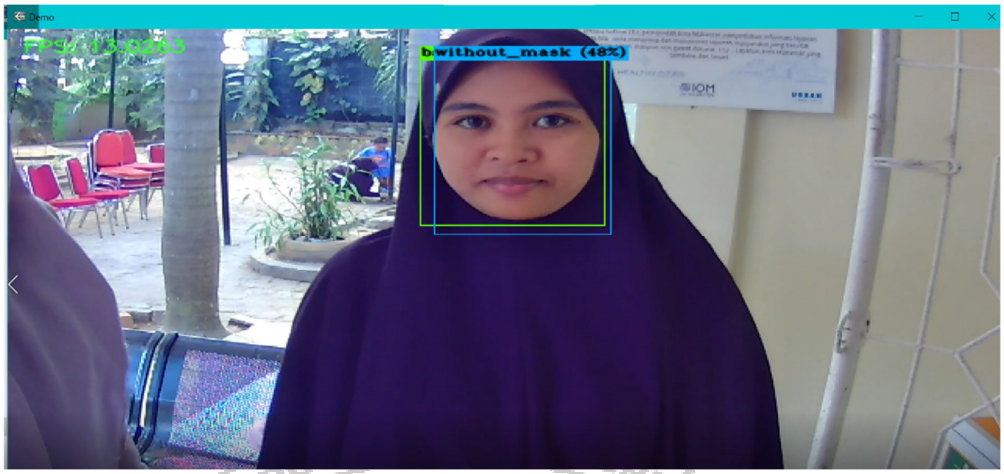
GAMBAR WAJAH PADA SAAT PENGUJIAN

- **WITH MASK**

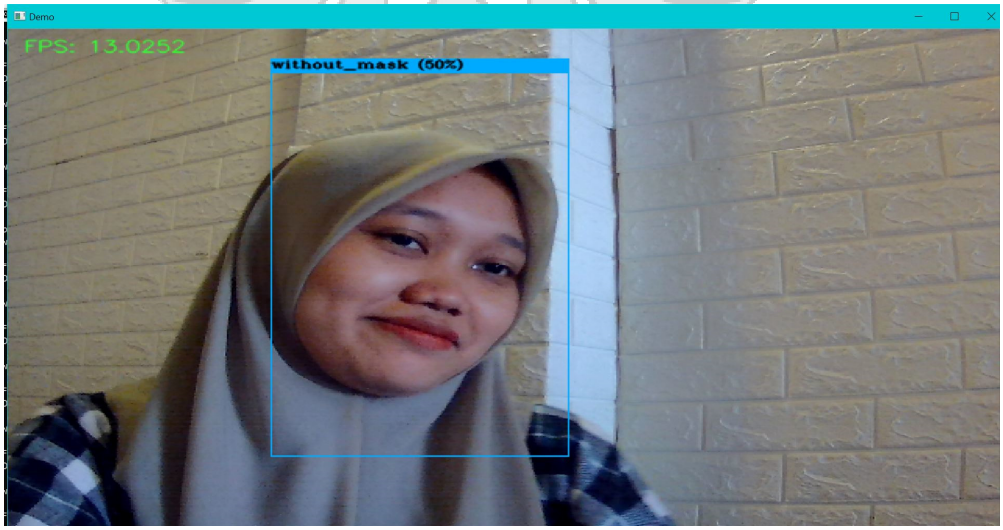




- **WITHOUT MASK**


Demo
without_mask (48%)


Demo
without_mask (65%)


Demo
FPS: 13.0252
without_mask (60%)

- **BAD MASK**