

**PERANCANGAN SISTEM MONITORING PENGGUNAAN  
MASKER KARYAWAN BERBASIS MACHINE LEARNING  
DI KANTOR PT TOWER BERSAMA GROUP**

**LAPORAN MAGANG  
DI  
PT. TOWER BERSAMA INFRASTRUKTUR GROUP**

**Oleh  
MUHAMAD ZHAFRAN MAHENDRA  
NIM: 13218004**



**PROGRAM STUDI SARJANA TEKNIK ELEKTRO  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG  
2022**

## **LEMBAR PENGESAHAN**

### **PERANCANGAN SISTEM MONITORING PENGGUNAAN MASKER KARYAWAN BERBASIS MACHINE LEARNING DI KANTOR PT TOWER BERSAMA GROUP**

Oleh:  
Muhamad Zhafran Mahendra  
13218004

Laporan magang ini telah diterima dan disahkan sebagai persyaratan untuk  
memperoleh nilai  
**MATA KULIAH EL4148 DAN EL4244**

DI  
PROGRAM STUDI SARJANA TEKNIK ELEKTRO  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG

Jakarta Selatan, 31 Januari 2022

Disetujui Oleh:

Dosen Pembimbing  
Mata Kuliah



Dr. Pranoto Hidayah Rusmin, S.T,  
M.T.

Penanggung Jawab  
Di Lokasi Magang



Ficky Julyansyah

## DAFTAR PUSTAKA

- [1] Softscients. "Cara Install PyTorch". softscients.com, 2021. [online]. Available: <https://softscients.com/2020/03/20/cara-install-pytorch/>. [Accessed : 21 Januari 2022]
- [2] Simplilearn. "What is PyTorch, and How Does It Work: All You Need to Know". Simplilearn.com. 2021. [online]. Available : <https://www.simplilearn.com/what-is-pytorch-article> . [Accessed : 21 Januari 2022]
- [3] Karimi, grace. "Introduction to YOLO Algorithm for Object Detection". section.io. 2021. [online]. Available : <https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/#:~:text=YOLO%20is%20an%20algorithm%20that,%2C%20parking%20meters%2C%20and%20animals>. [Accessed : 21 Januari 2022]
- [4] buffml. "YOLO OBJECT DETECTION ALGORITHM". Buffml.com. 2020. [online]. Available : <https://buffml.com/yolo-object-detection-algorithm/> [Accessed : 21 Januari 2022]
- [5] tzutalin. "labelImg". Github.com. 2021. [online]. Available : <https://github.com/tzutalin/labelImg> [Accessed : 21 Januari 2022]
- [6] BhattaCharyya, Jayita. " Step by step Guide To Object Detection Using Roboflow". 2020. [online]. Available : <https://analyticsindiamag.com/step-by-step-guide-to-object-detection-using-roboflow/#:~:text=Roboflow%20is%20a%20Computer%20Vision,Roboflow%20accepts%20various%20annotation%20formats>. [Accessed : 21 Januari 2022]
- [7] Dwyter, Brad. "Getting Started with Roboflow". 2021. [online]. Available : <https://blog.roboflow.com/getting-started-with-roboflow/> [Accessed : 21 Januari 2022]
- [8] Farrah F, Sarifah. "Panduan Cara Install XAMPP di Windows untuk Pertama Kali". 2020. [online]. Available : <https://www.nesabamedia.com/panduan-cara-install-xampp-di-windows-untuk-pertama-kali/> [Accessed : 21 Januari 2022]
- [9] Ashwani K. " What is XAMPP? And How to Install XAMPP". 2021. [online]. Availbale: <https://www.devopsschool.com/blog/what-is-xampp-and-how-to-install-xamp/> [Accessed : 21 Januari 2022]

# Daftar Isi

<b>1 B100 FORMULASI MASALAH .....</b>	<b>6</b>
1.1 MASALAH .....	6
1.1.1 <i>Latar belakang masalah</i> .....	6
1.1.2 <i>Informasi pendukung</i> .....	7
1.1.3 <i>Analisis Masalah</i> .....	8
1.1.4 <i>Kebutuhan yang harus dipenuhi</i> .....	9
1.2 SOLUSI .....	9
1.2.1 <i>Karakteristik Produk</i> .....	9
1.2.2 <i>Usulan Solusi</i> .....	10
1.2.3 <i>Analisis Usulan Solusi</i> .....	11
1.2.4 <i>Solusi yang dipilih</i> .....	14
1.3 PERENCANAAN PASAR .....	14
1.3.1 <i>Perkiraan Biaya</i> .....	14
1.3.2 <i>Analisa Finansial</i> .....	15
1.3.3 <i>Model Bisnis</i> .....	16
<b>2 B200 SPESIFIKASI .....</b>	<b>18</b>
2.1 SPESIFIKASI PRODUK.....	18
2.1.1 <i>Spesifikasi Akurasi Deteksi Objek</i> .....	18
2.1.2 <i>Spesifikasi Jarak Deteksi Objek</i> .....	18
2.1.3 <i>Spesifikasi Ketinggian Kamera</i> .....	18
2.1.4 <i>Spesifikasi Angle Deteksi Objek</i> .....	18
2.1.5 <i>Spesifikasi Jumlah Objek dalam 1 frame</i> .....	19
2.1.6 <i>Spesifikasi Jumlah Video Stream</i> .....	19
2.1.7 <i>Spesifikasi Penyimpanan Data</i> .....	19
2.2 TABEL SPESIFIKASI PRODUK .....	20
2.3 VERIFIKASI.....	22
2.3.1 <i>Spesifikasi Akurasi Deteksi Objek</i> .....	22
2.3.2 <i>Spesifikasi Jarak Deteksi Objek</i> .....	22
2.3.3 <i>Spesifikasi Ketinggian Kamera</i> .....	23
2.3.4 <i>Spesifikasi Angle Deteksi Objek</i> .....	23
2.3.5 <i>Spesifikasi Jumlah Objek dalam 1 frame</i> .....	23
2.3.6 <i>Spesifikasi Jumlah Video Stream</i> .....	24
2.3.7 <i>Spesifikasi Penyimpanan Data</i> .....	24
<b>3 B300 DESAIN SISTEM .....</b>	<b>25</b>
3.1 KONSEP SISTEM .....	25
3.1.1 <i>Arsitektur Utama</i> .....	25
3.1.2 <i>Topologi</i> .....	26
3.1.3 <i>Flow Process Implementasi Sistem</i> .....	27
3.2 DESAIN SISTEM .....	29
3.2.1 <i>Framework Machine Learning</i> .....	29
3.2.2 <i>Algoritma Machine Learning</i> .....	30
3.2.3 <i>Software</i> .....	33
3.2.4 <i>Hardware</i> .....	34
<b>4 B400 IMPLEMENTASI .....</b>	<b>36</b>

4.1	INSTALASI DEPENDENCIES .....	36
4.1.1	<i>Instalasi PyTorch</i> .....	36
4.1.2	<i>Instalasi Yolov5</i> .....	37
4.1.3	<i>Instalasi Dependencies untuk Training dan Running Machine Learning Model</i>	
	39	
4.1.4	<i>Instalasi labelImg</i> .....	40
4.1.5	<i>Instalasi Dependencies untuk labelImg</i> .....	41
4.2	PERSIAPAN DATASET .....	42
4.2.1	<i>Capture dengan webcam</i> .....	42
4.2.2	<i>Mengumpulkan Dataset Melalui Internet</i> .....	44
4.2.3	<i>Memperbaiki Dataset</i> .....	45
4.3	LABELLING DATASET.....	46
4.4	TRAINING MACHINE LEARNING MODEL .....	47
4.4.1	<i>Membuat Pipeline</i> .....	47
4.4.2	<i>Training</i> .....	47
4.4.3	<i>Output Training</i> .....	48
4.5	MEMBUAT DATABASE HASIL DETEksi PROGRAM .....	49
4.5.1	<i>Create Database</i> .....	49
4.5.2	<i>Create Table</i> .....	50
4.6	RUNNING PROGRAM.....	51
4.6.1	<i>Menjalankan Program Pada Foto</i> .....	51
4.6.2	<i>Real-time Detection</i> .....	52
4.6.3	<i>Real-time Detection Multiple Object Tracking</i> .....	53
<b>5</b>	<b>B500 PENGUJIAN.....</b>	<b>62</b>
5.1	PENGUJIAN SPESIFIKASI AKURASI DETEksi OBJEK.....	62
5.1.1	<i>Pengujian Data Dummy</i> .....	63
5.1.2	<i>Pengujian Data Real</i> .....	71
5.2	PENGUJIAN SPESIFIKASI JARAK DETEksi OBJEK .....	108
5.3	PENGUJIAN SPESIFIKASI KETINGGIAN KAMERA .....	108
5.4	PENGUJIAN SPESIFIKASI ANGLE DETEksi OBJEK .....	109
5.5	PENGUJIAN SPESIFIKASI JUMLAH OBJEK DALAM 1 FRAME.....	110
5.6	PENGUJIAN SPESIFIKASI JUMLAH VIDEO STREAM.....	111
5.7	PENGUJIAN SPESIFIKASI PENYIMPANAN DATA.....	113
<b>6</b>	<b>GUIDELINE PENGGUNAAN PROGRAM .....</b>	<b>114</b>
6.1	PROSEDUR AKTIVASI VIRTUAL ENVIRONMENT.....	114
6.2	PROSEDUR LABELLING DATASET .....	114
6.3	PROSEDUR TRAINING MACHINE LEARNING MODEL PROGRAM FACE-MASK DETECTION .....	116
6.4	PROSEDUR AKTIVASI DATABASE SERVER.....	120
6.5	PROSEDUR AKSES DATABASE SERVER.....	121
6.6	PROSEDUR MENGKONEKSIKAN IP CAMERAN DENGAN KOMPUTER.....	123
6.7	PROSEDUR RUNNING PROGRAM FACE-MASK DETECTION .....	124
6.8	SPESIFIKASI SISTEM BERDASARKAN PENGUJIAN .....	130

# 1 B100 FORMULASI MASALAH

## 1.1 *Masalah*

Pada proyek magang ini, masalah yang akan diselesaikan adalah masih kurang baiknya system pengawasan penggunaan masker dalam upaya mencegah penyebaran virus Covid-19. Selain itu, isu mengenai tingginya penyebaran virus Covid-19 di klaster perkantoran dan maraknya penyebaran virus Covid-19 varian omicron perlu menjadi perhatian khusus. Masalah ini didapatkan dengan menimbang berbagai macam faktor yang akan dijelaskan melalui sub bab berikut.

### 1.1.1 Latar belakang masalah

Istilah Industri 5.0 mengacu pada orang-orang yang bekerja bersama robot dan mesin pintar. Ini berkaitan erat dengan robot atau mesin pintar yang membantu manusia bekerja lebih baik dan lebih cepat dengan memanfaatkan teknologi canggih seperti *Internet of Things (IoT)*, *Machine Learning (ML)*, *Big Data*, *Artificial Intelligent (AI)*, dan masih banyak lagi. Ini menambahkan sentuhan pribadi manusia ke pilar otomatisasi dan efisiensi Industri 4.0.

Salah satu sub materi yang merupakan prospek menjanjikan pada era revolusi industry ini ialah video analytics. Pada awalnya, Video Analytics adalah teknologi yang memproses sinyal video digital menggunakan algoritma khusus untuk menjalankan fungsi terkait keamanan. Dengan diperkenalkannya IP CCTV dan perangkat lunak yang lebih cerdas, Video Analytics menjadi lebih kuat dan lebih menjanjikan. Video analytics kini banyak digunakan untuk system control dan monitor untuk berbagai sector bisnis, mulai dari pengenalan wajah otomatis, pengenalan plat nomor kendaraan, mask detection, menghitung objek, hingga mendeteksi objek berkeliaran di suatu area berikut kecepatan dan arah pergerakannya. Ide paling sederhana di balik Video Analytics adalah membuat komputer menemukan peristiwa atau kejadian yang memerlukan perhatian manusia, atau mungkin memperingatkan operator atau membuat tindakan tentang insiden yang mungkin atau telah terjadi. Sangat sering, insiden atau peristiwa diselesaikan jauh setelah waktu dilaporkannya sehingga terkadang tindakan pencegahan tidak dapat dilakukan. Dengan adanya video analytics, sangat banyak informasi yang dapat diperoleh yang selanjutnya informasi tersebut dapat diubah menjadi *knowledge* sebagai acuan komputer untuk melakukan suatu tindakan tertentu dan mempresentasikannya kepada manusia.

Maraknya penyebaran virus Covid-19 varian omicron dan penyebaran melalui klaster perkantoran pada varian delta periode sebelumnya perlu dijadikan perhatian khusus. Banyak masyarakat yang mulai mengabaikan protocol kesehatan seperti tidak menggunakan masker, tidak rutin mencuci tangan, tidak menjaga jarak, dsb. Tidak jarang Satgas (Satuan tugas) covid-19 melakukan Razia keliling untuk memonitor kepatuhan masyarakat akan protocol kesehatan. Masker sekarang ini merupakan suatu komponen yang wajib dibawa dan digunakan ketika seseorang berpergian. Pemakaian masker menjadi wujud disiplin nasional selama pandemi covid-19. Namun system pengawasan terhadap pemakaian masker mayoritas masih sangat minim. Terkadang hanya melalui Satgas Covid-19 saja dilakukan monitoring penggunaan masker terhadap masyarakat yang berada di tempat umum. Keterbatasan SDM (Sumber Daya Manusia) terkhusus Satgas Covid-19 merupakan faktor penghambat proses pemantauan penggunaan masker masyarakat. Oleh karena itu perlu dikembangkan suatu system otomatis yang secara independen melakukan monitoring terhadap penggunaan masker masyarakat terutama di kawasan perkantoran.

## 1.1.2 Informasi pendukung

### 1.1.2.1 Data Klaster Perkantoran COVID-19 DKI Jakarta

Berdasarkan data yang telah dirilis oleh Dinas Tenaga Kerja Transmigrasi dan Energi (Disnakertrans) DKI Jakarta, sejumlah kantor yang ditutup lantaran memiliki kasus COVID-19. Data tersebut dihimpun pada periode 11 Januari 2021 sampai dengan 26 April 2021. Jumlah perusahaan atau perkantoran yang ditutup sementara akibat COVID-19 sebanyak 2.114 sementara keputusan penutupan sementara diambil dari hasil sidak di 3.703 kantor.

Selain karena kasus COVID-19, terdapat 21 perkantoran yang juga ditutup sementara di periode yang sama karena tidak menjalankan protokol kesehatan. Jumlah perusahaan yang ditutup sementara akibat ditemukan kasus COVID-19 ditambah pelanggaran protokol kesehatan mencapai 2.135.

Disnakertrans juga merilis data wilayah dengan jumlah masing-masing temuan kasus COVID-19 di perkantoran.

1. Jakarta Selatan dengan 824 perkantoran
2. Jakarta Pusat dengan 652 perkantoran
3. Jakarta Barat 270 perkantoran
4. Jakarta Utara 201 perkantoran
5. Jakarta Timur 167 perkantoran

Sedangkan perkantoran yang ditutup karena tidak menjalankan protokol kesehatan dengan benar terbanyak juga didominasi Jakarta Selatan.

1. Jakarta Selatan 12 perkantoran
2. Jakarta Timur 4 perkantoran
3. Jakarta Utara 3 perkantoran
4. Jakarta Pusat 2 perkantoran

Sedangkan, tidak ditemukan perkantoran yang ditutup akibat tak menjalankan protokol kesehatan di Jakarta Barat. Kasus COVID-19 dari klaster perkantoran sempat meningkat pada periode 5-18 April 2021. Pada periode 5-11 April, tercatat ada 157 kasus konfirmasi COVID-19 dari 78 perkantoran di DKI Jakarta. Sedangkan pada periode 12-18 April, tercatat ada 425 kasus konfirmasi COVID-19 dari 177 perkantoran. [3].

Meningkatnya kasus COVID-19 dari klaster perkantoran ini disebabkan karena program vaksinasi yang sudah dilakukan dan setiap harinya semakin banyak rakyat Indonesia yang menerima vaksin. Oleh sebab itu, semakin banyak juga perusahaan, perkantoran, hingga sekolah yang mulai memberlakukan kegiatan secara luring di tempatnya masing-masing. Akibatnya, risiko untuk mengalami kontak dengan COVID-19 semakin besar, ditambah dengan protokol kesehatan yang tidak dijalankan dengan baik menimbulkan semakin banyaknya kasus COVID-19 dari klaster perkantoran.

### 1.1.2.2 Covid-19 Varian Omicron

Dilansir dari [www.nasional.kompas.com](http://www.nasional.kompas.com), kasus Covid-19 varian Omicron di Indonesia terus bertambah. Kementerian kesehatan menyatakan, hingga Sabtu (15/1/2022) terdapat 748 kasus Covid-19 varian Omicron yang terdiri dari 569 kasus perjalanan luar negri dan 155 kasus transmisi local. Varian Omicron diperkirakan lebih mudah menular hingga 105 persen dibandingkan varian Delta. Para ilmuwan membandingkan jumlah infeksi varian Omicron, Alpha, dan Delta selama periode 21 hari. Ditemukan, perbedaan tingkat penularan pada orang dengan infeksi varian Delta dan Omicron sekitar 105 persen. Ini merupakan hasil penelitian para ilmuwan Perancis yang telah dipublikasikan di situs

*medRxiv* dengan menganalisis 131.478 sampel di Perancis selama periode 25 Oktober hingga 18 Desember 2021.

### 1.1.3 Analisis Masalah

#### 1.1.3.1 Konstrain Ekonomi

Konstrain ekonomi merupakan biaya total yang diperlukan untuk mengembangkan sistem yang dirancang. Biaya yang diperlukan tentunya bergantung kepada harga dan jumlah komponen serta fitur-fitur yang akan diimplementasikan pada sistem tersebut. Semakin banyak dan mahal komponen yang digunakan maka akan semakin besar biaya yang dibutuhkan untuk merancang dan memproduksi sistem tersebut, selain itu sistem ini tentunya akan ditujukan kepada berbagai perusahaan baik skala kecil, menengah, maupun besar. Maka dari itu perlu diperhatikan pemilihan komponen yang akan digunakan sesuai dengan kebutuhan dan keefektifannya terhadap performa sistem.

#### 1.1.3.2 Konstrain Keberlanjutan (*sustainability*)

Konstrain keberlanjutan merupakan konstrain yang ditinjau dari mudah atau tidaknya produk dapat dikembangkan lebih lanjut. Produk yang dikembangkan dijadikan sebagai *proof of concept* untuk menunjukkan bahwa solusi yang ditawarkan dapat mengatasi permasalahan yang dibawa. Terdapat berbagai batasan masalah pada perancangan produk ini sehingga akan dikembangkan lebih lanjut pada proses selanjutnya.

#### 1.1.3.3 Konstrain Manufakturabilitas (*manufacturability*)

Konstrain manufakturabilitas merupakan konstrain yang ditinjau dari kemudahan proses perancangan sistem. Kemudahan proses perancangan sistem dapat dilihat dari ketersediaan *hardware* di pasaran dan kemudahan instalasi alat.

#### 1.1.3.4 Konstrain Akurasi

Sistem pengawasan penggunaan masker akan secara otomatis melakukan pendekripsi terhadap karyawan yang ada di area perkantoran. Akurasi disini merupakan tingkat keakuratan dari data-data yang masuk kedalam *database* dengan kondisi sebenarnya. Semakin akuratnya sistem yang dikembangkan maka akan semakin merepresentasikan kondisi sebenarnya di area yang dilakukan monitoring.

#### 1.1.3.5 Konstrain Kepraktisan

Sistem yang dirancang harus mudah digunakan karena hal tersebut sangat berpengaruh terhadap *user experience*. Pengguna terbagi menjadi 2 kategori yaitu operator sebagai pihak yang memonitor data dan anggota organisasi dikawasan perkantoran. Semakin sedikit Langkah-langkah yang perlu dilakukan oleh pengguna sehingga sistem dapat melakukan prosesnya maka akan semakin baik parameter kepraktisannya.

#### 1.1.3.6 Konstrain Kesehatan dan Keselamatan

Salah satu tujuan penting dari sistem yang dirancang adalah agar pemantauan terhadap penggunaan masker yang merupakan salah satu dari protocol kesehatan dalam upaya untuk menekan penyebaran virus covid-19 dapat berjalan dengan baik. Tujuan ini sangat berkaitan dengan aspek kesehatan dan keselamatan anggota organisasi di area perkantoran. Semakin ketat sistem pengawasan penggunaan masker yang dikembangkan

maka aspek kesehatan dan keselamatan anggota organisasi di area perkantoran akan semakin tinggi.

#### 1.1.4 Kebutuhan yang harus dipenuhi

1. Sistem deteksi masker wajah (*Face-mask Detection*) harus memiliki akurasi yang tinggi
2. Sistem deteksi dapat melakukan pendekripsi terhadap seluruh jenis dan warna masker yang digunakan oleh karyawan
3. Pengawasan terhadap data dapat dilakukan setiap saat
4. Proses pendekripsi masker wajah dapat dilakukan dengan mudah dan cepat
5. Data yang tersimpan pada sistem hanya dapat diakses oleh pihak pengelola data.

## 1.2 Solusi

Berikut ini akan dijabarkan karakteristik dari produk yang akan dirancang, alternatif alternatif solusi yang memungkinkan dan memenuhi karakteristik produk, serta solusi yang dipilih berdasarkan tingkat penyelesaian masalah yang paling tinggi menggunakan *pairwise comparison matrix* dan *decision matrix*.

### 1.2.1 Karakteristik Produk

Cara penulisan bagian ini bebas, tetapi setidaknya menunjukkan:

- Fitur Utama:

Sistem pendekripsi masker wajah (*Face-mask Detection*) dengan akurasi yang tinggi yang mampu memonitor dan melakukan pendekripsi penggunaan masker wajah secara otomatis.

- Fitur Dasar:

- Mampu mendekripsi penggunaan masker pada kondisi ideal
- Mampu menyimpan hasil pendekripsi masker wajah kedalam *database*
- Mampu mendekripsi banyak wajah dalam frame yang sama
- Mampu dielaborasikan untuk beberapa input kamera dalam satu waktu yang sama

- Fitur Tambahan:

- Mampu mendekripsi semua jenis dan warna masker
- Mampu mendekripsi penggunaan masker terhadap wajah yang menggunakan kacamata
- Pendekripsi dapat dilakukan dari jarak yang jauh untuk berbagai sisi wajah

- Sifat solusi yang diharapkan

- Proses deteksi memiliki akurasi yang tinggi
- Pengawasan terhadap data hasil pendekripsi dapat dilakukan tiap saat.
- Sistem memiliki harga yang dapat bersaing dengan produk sejenis yang terdapat di pasaran
- Sistem yang dirancang harus mudah diproduksi dan mudah diinstalasi
- Sistem yang dirancang tidak memerlukan perawatan yang intensif
- Data yang tersimpan pada sistem hanya dapat diakses oleh pihak pengelola data.

## 1.2.2 Usulan Solusi

### 1.2.2.1 Solusi 1

Solusi pertama yang ditawarkan adalah solusi pendekripsi penggunaan masker sebagai persyaratan memasuki area perkantoran. Implementasi system seperti ini biasanya menggunakan suatu komponen yang berdiri secara independent pada pintu-pintu masuk area perkantoran ataupun fasilitas-fasilitas umum lainnya. Pengguna harus menghadap ke arah kamera agar kamera dapat melakukan pendekripsi masker wajah. Apabila wajah yang terdeteksi menggunakan masker, maka karyawan tersebut memperoleh izin untuk memasuki area perkantoran tersebut. Sedangkan Ketika wajah yang terdeteksi tidak menggunakan masker maka karyawan tersebut tidak diizinkan untuk memasuki area perkantoran tersebut.

Wajah yang dideteksi hanya merupakan wajah-wajah karyawan yang menghadap ke depan kamera (*Frontal-face*). Pengguna diharuskan untuk mendekatkan dan menghadapkan wajahnya ke kamera pada jarak tertentu dan kemudian system akan melakukan proses pendekripsi.

### 1.2.2.2 Solusi 2

Solusi kedua yang ditawarkan adalah solusi pendekripsi penggunaan masker pada Lorong-lorong suatu lantai di perkantoran ataupun tempat karyawan melintas. Implementasi system ini dapat menggunakan CCTV *Existing* yang ditambahkan proses analitik yang disorot dari bagian atas Lorong. Pengguna tidak perlu menghadap ke kamera agar system dapat mendekripsi penggunaan masker pada wajah karyawan yang lewat. Karyawan hanya perlu berjalan melewati Lorong tersebut dan system secara otomatis melakukan pendekripsi.

Ketika wajah yang terdeteksi tidak menggunakan masker, kamera akan secara otomatis melakukan *capture* terhadap wajah tersebut dan mengirimkannya ke operator. Selanjutnya operator akan memberitahukan / menegur karyawan (dapat dengan system atau manual) yang terdeteksi tersebut untuk menggunakan masker dengan baik Ketika berada di area perkantoran

### 1.2.2.3 Solusi 3

Solusi ketiga yang ditawarkan adalah solusi pendekripsi penggunaan masker pada meja kerja / tempat karyawan bekerja. Seperti pada solusi 2, implementasi system ini dapat menggunakan CCTV *Existing* yang ditambahkan proses analitik yang disorot dari bagian atas ruang kerja karyawan. Pengguna tidak perlu menghadap ke kamera agar system dapat mendekripsi penggunaan masker pada wajah karyawan yang lewat. Karyawan hanya perlu berjalan melewati Lorong tersebut dan system secara otomatis melakukan pendekripsi.

Pada solusi 3 ini wajah yang dideteksi dapat sangat bervariasi dikarenakan monitoring yang dilakukan adalah aktivitas karyawan selama berada di ruang kerja. Tidak seperti solusi 1 dan 2 yang hanya perlu mendekripsi wajah bagian depan saja (*Frontal face*), pada solusi ini wajah akan terdeteksi pada kemiringan-kemiringan wajah tertentu sesuai batas pada spesifikasi yang disebutkan. Parameter hasil pendekripsi merupakan persentase wajah yang menggunakan masker dan tidak pada ruangan kerja tertentu. Apabila persentase batas maksimum wajah tidak menggunakan masker telah tercapai, maka operator akan menghubungi petugas keamanan untuk menertibkan karyawan yang ada di ruangan tersebut agar selalu menggunakan masker.

### 1.2.3 Analisis Usulan Solusi

Pemilihan solusi dari ketiga alternatif solusi diatas akan mempertimbangkan aspek-aspek yang telah disebutkan di subbab 2.1.3. Masing-masing dari aspek tersebut akan dianalisis terlebih dahulu dan diberikan pembobotan sesuai urgensinya menggunakan *pair-wise comparison matrix*. Kemudian pemilihan solusi akan dilakukan menggunakan *decision matrix* sesuai pembobotan aspek yang telah dilakukan.

Berikut adalah analisis usulan solusi terhadap aspek-aspek yang telah ditentukan:

- Aspek Ekonomi

Dilihat dari biaya, untuk solusi pertama memerlukan biaya yang lebih mahal dibandingkan solusi kedua dan ketiga dikarenakan pada solusi pertama diperlukan suatu *interface* kepada pengguna untuk tampilan bahwa karyawan sudah diizinkan memasuki area perkantoran atau belum. Pada solusi kedua dan ketiga hanya memerlukan kamera dan computer saja sebagai alat monitoring deteksi masker dan tidak memerlukan *interface* ke pengguna sebagai tampilan apapun.

- Aspek Manufakturabilitas

Untuk aspek manufakturabilitas dilihat dari segi instalasi perangkat kerasnya (*Hardware*). Solusi 2 dan 3 memiliki tingkat kesulitan instalasi alat lebih sulit dibandingkan dengan konsep 1. Hal ini dikarenakan pada konsep 1 perangkat yang dipasang merupakan suatu bagian yang terpisah dari bangunan perkantoran sedangkan pada konsep 2 dan 3 memerlukan kamera CCTV sebagai modul kamera.

- Aspek Keberlanjutan

Dari aspek keberlanjutan, solusi 1 lebih baik dibandingkan dengan solusi 2 dan 3. Sepertinya yang sudah dijelaskan pada bagian sebelumnya, solusi 1 memiliki *hardware* yang secara umum berdiri sendiri dan terpisah dari bagian bangunan. Sehingga dapat ditambahkan fitur-fitur pengembangan lainnya contohnya seperti sensor suhu, RFID atau fingerprint sensor untuk proses presensi, dsb. Pada konsep 2 dan 3 tidak dapat banyak hal yang dapat dikembangkan dari sisi *hardware*. Untuk pengembangan lebih lanjut mengenai *software* ketiga alternatif solusi tersebut relative sama.

- Aspek Akurasi

Aspek akurasi merupakan salah satu faktor penting ketika merancang suatu system. Solusi 1 akan memiliki akurasi yang paling tinggi dibandingkan konsep 2 dan 3. Hal ini dikarenakan pada konsep 1 wajah yang dideteksi berjarak dekat dan menghadap depan (*Frontal-face*). Kemudian konsep 2 akan memiliki akurasi yang lebih tinggi dibandingkan dengan konsep 3 dikarenakan pada konsep 2 walaupun sama-sama memiliki jarak yang jauh seperti konsep 3, namun ini diimplementasikan pada suatu Lorong dimana *angle* wajah yang dideteksi relative konsisten. Sedangkan pada konsep 3 variasi *angle* wajah yang dideteksi sangat bervariasi sehingga akurasi akan semakin rendah.

- Aspek Kepraktisan

Dilihat dari ketiga solusi yang ditawarkan, solusi pertama memiliki tingkat kepraktisan yang paling rendah. Hal ini dikarenakan karyawan perlu mengarahkan wajahnya ke arah lensa kamera terlebih dahulu agar system dapat mendeteksi wajah pengguna. Berbeda dengan solusi kedua dan ketiga dimana karyawan tidak perlu melakukan apapun kemudian system tetap dapat melakukan pendekripsi penggunaan masker dari wajah karyawan tersebut.

- Aspek Kesehatan dan Keselamatan

Solusi pertama memiliki nilai paling rendah pada aspek kesehatan dan keselamatan. Pendekripsi masker pada konsep 1 ini hanya mengambil sampel pada kondisi ketika karyawan memasuki pintu masuk area perkantoran saja. Hal tersebut tidak menjamin bahwa karyawan tersebut menggunakan maskernya selama berada di area perkantoran. Solusi kedua memiliki nilai aspek Kesehatan dan keselamatan yang sama kurang baiknya seperti solusi pertama karena hanya mengambil sampel pada Lorong-lorong saja dan tidak menjamin ketika karyawan tersebut bekerja mereka menggunakan masker. Solusi ketiga memiliki nilai paling baik pada aspek ini dikarenakan pemantauan penggunaan masker karyawan dilakukan setiap saat bahkan ketika karyawan tersebut sedang berada di meja kerjanya. Hal ini memungkinkan data yang diperoleh sistem lebih sesuai dengan data aktual penggunaan masker karyawan yang ada di area perkantoran.

Untuk memilih solusi yang tepat, akan dianalisis menggunakan *pair-wise comparison matrix* dan *analytic hierarchy process*. Penentuan ini akan didasarkan dari konstrain yang telah ditentukan pada subbab sebelumnya. Pertama, akan ditentukan bobot dari setiap konstrain yang ada terhadap solusi dengan *pair-wise comparison matrix*. Pembobotan memiliki rentang nilai yaitu 0.33, 0.5, 1, 2, dan 3.

**Table 1.2-1 Pair-wise Comparison Matrix**

Aspek	Ekonomi	Manufakturabilitas	Keberlanjutan	Akurasi	Kepraktisan	Kesehatan dan Keselamatan	Total Nilai
Ekonomi	1	1	0.5	0.5	0.5	0.33	2
Manufakturabilitas	1	1	0.5	0.5	1	0.33	3
Keberlanjutan	2	2	1	0.5	2	0.5	7
Akurasi	2	2	2	1	2	1	10
Kepraktisan	2	1	0.5	0.5	1	0.5	4
Kesehatan dan Keselamatan	3	3	2	1	2	1	12
Total Nilai							38

Berikut adalah tabel hasil perhitungan bobot menggunakan *pair-wise comparison matrix*.

**Table 1.2-2 Hasil Perhitungan Bobot**

Aspek	Normalisasi Aspek	Bobot (%)	Prioritas
Ekonomi	0,052631579	5,263157895	6
Manufakturabilitas	0,078947368	7,894736842	5
Keberlanjutan	0,184210526	18,42105263	3

Akurasi	0,263157895	26,31578947	2
Kepraktisan	0,105263158	10,52631579	4
Kesehatan dan Keselemanatan	0,315789474	31,57894737	1

Berikut adalah tabel *decision matrix* dari setiap aspek yang dianalisis.

**Table 1.2-3 Decision Matrix**

No	Aspek	Bobot (%)	Nilai Solusi 1	Hasil Perkalian Bobot Solusi 1	Solusi 2	Hasil Perkalian Bobot Solusi 2	Solusi 3	Hasil Perkalian Bobot Solusi 3
1	Ekonomi	5.26	1	5.26	2	10.52	2	10.52
2	Manufaktur abilitas	7.89	3	23.67	2	15.78	2	15.78
3	Keberlanjutan	18.42	3	55.26	2	36.84	2	36.84
4	Akurasi	26.32	3	78.96	2	52.64	1	26.32
5	Kepraktisan	10.53	1	10.53	3	31.59	3	31.59
6	Kesehatan dan Keselamatan	31.58	1	31.58	2	63.16	3	94.74
Total			12	205.26	13	210.53	13	215.79

Dibawah ini merupakan penjelasan tiap nilai dari tiap aspek yang dianalisis.

- Aspek Ekonomi
  1. Membeli seluruh komponen diantaranya computer, perangkat komunikasi system, dan modul kamera secara terpisah
  2. Hanya membeli *hardware* berupa computer dan perangkat komunikasi sistem. Kamera menggunakan kamera *existing*
  3. Semua *hardware* menggunakan komponen *existing*
- Aspek Manufakturabilitas
  1. Proses instalasi system rumit (diperlukan perizinan yang rumit atas regulasi tertentu)
  2. Proses instalasi system cukup mudah (diperlukan perizinan yang mudah atas regulasi tertentu)
  3. Proses instalasi sistem sangat mudah (*plug and play*)
- Aspek Keberlanjutan
  1. Produk tidak dapat dikembangkan lebih lanjut
  2. Produk dapat dikembangkan lebih lanjut dari sisi *Core Program*
  3. Produk dapat dikembangkan lebih lanjut dari sisi *Core Program* dan *User Interface*

- Aspek Akurasi
  1. Sistem dapat mendeteksi objek dari jarak yang jauh dan tidak perlu menghadap modul kamera
  2. Sistem dapat mendeteksi objek dari jarak yang jauh dan menghadap modul kamera
  3. Sistem dapat mendeteksi objek dari jarak yang dekat dan menghadap modul kamera
- Aspek Kepraktisan
  1. Karyawan mendekatkan dan menghadapkan wajah pada modul kamera
  2. Karyawan perlu menghadapkan wajahnya pada modul kamera
  3. Karyawan tidak perlu menghadapkan dan menghadapkan wajah pada modul kamera
- Aspek Kesehatan dan Keselamatan
  1. Sistem deteksi penggunaan masker karyawan hanya dilakukan ketika karyawan memasuki area perkantoran saja
  2. Sistem deteksi penggunaan masker karyawan dilakukan pada Lorong-lorong tempat karyawan melintas pada area perkantoran
  3. Sistem deteksi penggunaan masker karyawan dilakukan selama karyawan beraktivitas di area perkantoran

#### 1.2.4 Solusi yang dipilih

Dari ketiga usulan solusi yang telah dianalisis menggunakan *pair-wise comparison matrix* dan *decision matrix*, usulan solusi yang terpilih adalah solusi ke 3 yaitu sistem pendekripsi masker yang dilakukan selama karyawan beraktivitas di area perkantoran. Dipilih solusi ini sebab memiliki bobot yang lebih besar dibandingkan dengan kedua alternatif solusi lainnya.

### 1.3 Perencanaan Pasar

#### 1.3.1 Perkiraan Biaya

Berikut akan dijabarkan mengenai perkiraan biaya produksi, biaya pengembangan, dan biaya *engineering* dari solusi yang dipilih.

#### Production Cost

**Table 1.3-1 Production Cost**

No	Komponen	Harga Satuan	Jumlah	Total Harga
1	On-premise Computer for production (CPU, GPU, LCD, Mouse)	Rp 25.000.000	1	Rp 25.000.000
Total				Rp 25.000.000

## Development Cost

Table 1.3-2 Development Cost

No	Komponen	Harga Satuan	Jumlah	Total Harga
1	Computer for development (CPU, GPU, LCD, Mouse)	Rp 25.000.000	1	Rp 25.000.000
2	IP Camera	Rp 700.000	1	Rp 700.000
3	LAN Cable 20m	Rp 40.000	1	Rp 40.000
4	Meteran Digital	Rp 6.650.000	1	Rp 6.650.000
5	Camera Tripod	Rp 600.000	1	Rp 600.000
	<i>Software Developer</i>	Rp 17.680.000	2	Rp 35.360.000
Total				Rp 68.350.000

## Engineering Cost

Table 1.3-3 Engineering Cost

No	Komponen	Harga Satuan	Jumlah	Total Harga
1	Instalasi Alat	Rp 3.000.000	1	Rp 3.000.000
Total				Rp 3.000.000

### 1.3.2 Analisa Finansial

Pada bagian analisis finansial ini, akan digunakan indicator perkiraan biaya untuk menghitung prospek finansial produk. Dipisahkan terlebih dahulu bagian yang menjadi *fixed cost* dan *variable cost*. *Fixed cost* merupakan biaya tetap yang harus dikeluarkan perusahaan tanpa adanya intervensi dari biaya produksi sedangkan *variable cost* merupakan biaya yang harus dikeluarkan perusahaan seiring dengan pertambahan laju produksi. Sehingga dapat disegmentasikan *fixed cost* meliputi *development cost* sedangkan *variable cost* merupakan *production cost* dan *engineering cost*.

Ongkos pembuatan satu produk yang sudah memperhitungkan *engineering cost* adalah Rp28.000.000,00. Persentase keuntungan untuk penjualan 1 unit produk yang digunakan adalah 50%. Sehingga setelah dihitung dengan biaya produksi, *engineering* dan keuntungan, harga penjualan adalah Rp 42.000.000,00. Diasumsikan akan terjual 1 unit produk untuk setiap 2 bulannya. Kemudian perhitungan juga menggunakan acuan suku bunga tahunan yang dimiliki oleh Bank Indonesia yaitu sebesar 4%. Maka nilai NPV yang diperoleh adalah:

$$NPV = -\text{Investasi Awal} + \sum_{t=1}^n \frac{\text{Cash Flow} \times q}{(1+k)^t}$$

$$NPV = -68.350.000 + \sum_{t=1}^1 \frac{(42.000.000 - 25.000.000) \times 6}{(1+0.04)^t}$$

$$NPV = -68.350.000 + 98.076.923$$

$$NPV = 29.726.923 (\text{positif})$$

Dengan:

Investasi Awal = Development Cost

n = Tahun Peninjauan

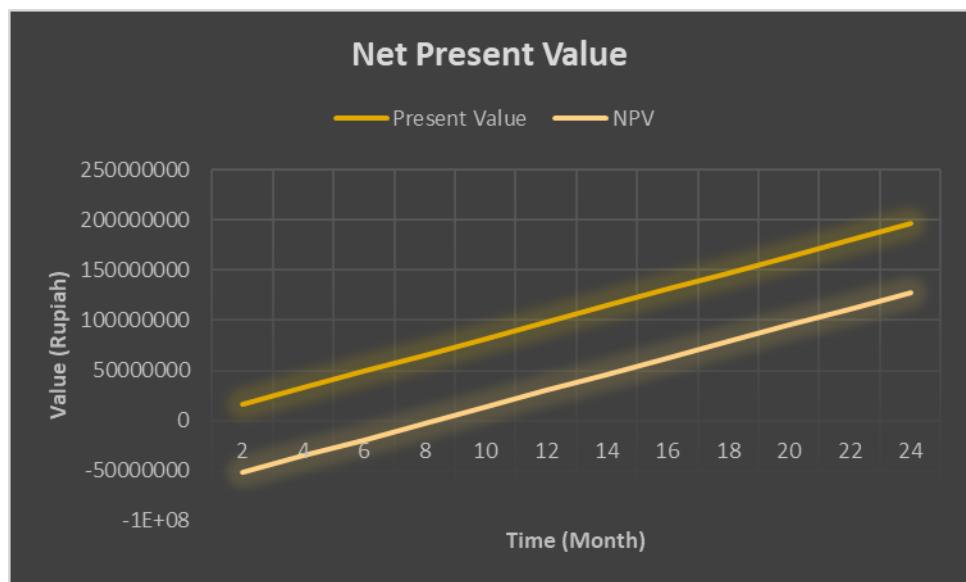
t = Periode 1 tahun

q = Jumlah Produk terjual dalam 1 tahun

Cash flow = Harga Jual – Harga Beli

k = Suku Bunga Pinjaman Bank Indonesia

Berikut adalah grafik Net Present Value produk yang dirancang.



Gambar 1.3-1 Net Present Value (NPV)

Dapat dilihat pada gambar diatas yaitu grafik *Net Present Value* (NPV) produk yang dirancang. Dapat terlihat Ketika diasumsikan terjual 1 unit produk setiap 2 bulannya, maka NPV akan bernilai positif pada bulan ke 10. Dengan demikian dapat diartikan juga *Break Even Point* (BEP) produk adalah 5 unit.

### 1.3.3 Model Bisnis

Setidaknya membahas siapa yang akan membeli produk tersebut, siapa yang membiayai pengembangannya, operatornya, dan pihak-pihak mana yang mendapatkan keuntungan.

Setidaknya membahas siapa yang diharapkan memproduksi, berinvestasi, memasarkan produk, perijinan, dan lainnya.

Produk yang dirancang merupakan system monitoring penggunaan masker wajah berbasis *Machine Learning* di area perkantoran. Oleh karena itu, target utama konsumen yang diharapkan ialah seluruh organisasi ataupun lembaga berskala kecil hingga besar yang ingin meningkatkan sistem pengawasan terhadap protocol kesehatan yaitu penggunaan masker di area perkantoran. Dengan adanya system pengawasan ini, diharapkan penggunaan masker karyawan dapat terawasi lebih baik sehingga karyawan yang berada di area perkantoran lebih patuh dalam menggunakan masker dan pada akhirnya angka penyebaran virus covid-19 di area pekantoran dapat dikurangi. Sistem ini juga merupakan langkah nyata perusahaan untuk terus menjaga kesehatan dan keselamatan karyawannya agar terhindar dari wabah virus covid-19.

Proses implementasi pertama akan dilakukan pada area perkantoran perusahaan pengembang system ini yaitu PT Tower Bersama Infrastruktur Group (TBIG). Perangkat keras yang perlu dibeli hanyalah sebuah computer tempat program akan berjalan. Input modul kamera akan berasal dari kamera CCTV *existing* PT Tower Bersama Infrastruktur Group. Computer tersebut akan beroperasi pada beberapa kamera yang jumlahnya akan dibahas pada bab selanjutnya. Sistem ini diharapkan juga dapat dipasarkan lebih lanjut kepada perusahaan-perusahaan lain yang ingin meningkatkan standard protocol kesehatan di area kantornya dengan memanfaatkan kamera CCTV *existing* perusahaan pembeli system ini.

## 2 B200 SPESIFIKASI

### 2.1 Spesifikasi Produk

#### 2.1.1 Spesifikasi Akurasi Deteksi Objek

Akurasi deteksi objek merupakan tingkat ketepatan system dalam mendeteksi objek wajah menggunakan masker maupun wajah tanpa menggunakan masker. Perhitungannya memperhitungkan juga Batasan-batasan pada spesifikasi lain seperti jarak, ketinggian, sudut, dsb. Sehingga sample yang diambil untuk dihitung akurasinya hanya sampel yang berada pada batas keseluruhan spesifikasi saja. Berdasarkan projek-projek yang telah dikerjakan oleh PT Tower Bersama Group, akurasi minimum yang masih dapat diterima untuk pemodelan *machine learning* berada di angka 80%. Akurasi ini merupakan akurasi yang sampelnya diambil secara real di lapangan. Sehingga nilai ini dijadikan acuan oleh penulis untuk batas akurasi minimum *real system* yang akan dikembangkan. Namun penulis juga membuat akurasi minimum untuk input data *dummy*. Data-data ini merupakan data-data yang diperoleh penulis melalui internet untuk dilakukan pengujian system. Parameter yang dijadikan acuan adalah Studi *Face in Video Evaluation* (FIVE) tertanggal Maret 2017 dilakukan tes algoritma diterapkan pada *video* yang diambil di gerbang keberangkatan dan tempat olahraga memiliki akurasi terbaik sekitar 94.4%. Sehingga nilai ini dijadikan acuan oleh penulis untuk batas akurasi minimum *dummy system* yang akan dikembangkan.

#### 2.1.2 Spesifikasi Jarak Deteksi Objek

Jarak deteksi objek merupakan seberapa jauhnya wajah terhadap lensa kamera. Parameter ini diambil dari ukuran ruangan kerja yang digunakan di PT Tower Bersama Group yang berada di lantai 15. Ukuran ruangan tersebut adalah sekitar 3 m x 5 m. Sehingga penulis mengambil jarak terjauh yang merupakan lebar dari ruangan kerja tersebut yaitu 5 meter sebagai acuan jarak system masih dapat melakukan pendekslan penggunaan masker karyawan.

#### 2.1.3 Spesifikasi Ketinggian Kamera

Ketinggian kamera merupakan sebaring tingginya lensa kamera terhadap *ground*. Pada implementasi system ini diinginkan digunakan kamera CCTV *existing* pada ketinggian 3 meter. Namun terdapat kendala yaitu belu diizinkannya akses terhadap CCTV *existing* kantor tersebut untuk proses pengembangan produk. Sehingga alternatif yang digunakan ialah digunakan tripod yang ditempatkan setinggi mungkin yang bisa dilakukan. Dikarenakan keterbatasan alat, ketinggian maksimum yang dapat diperoleh untuk pengujian system adalah 2.3 meter. Sehingga nilai ini dijadikan acuan oleh penulis untuk batas ketinggian CCTV pada proses pengujian system.

#### 2.1.4 Spesifikasi Angle Deteksi Objek

Angle merupakan sudut kemiringan sisi wajah yang tampak pada kamera.  $0^\circ$  merupakan titik dimana wajah sepenuhnya mengarah ke arah lensa kamera. Sudut positif merupakan titik dimana wajah menampakan sisi wajah bagian kiri dan kebalikannya sudut negative merupakan titik dimana wajah menampakan sisi wajah bagian kanan. Penulis mengambil acuan agar system masih dapat mendekripsi dengan akurasi yang baik adalah bagian dari *landmark* bibir dan batang hidung masih terlihat. Sehingga dipilih Batasan sudut kemiringan wajah yaitu  $\pm 60^\circ$  yang mempertimbangkan bahwa garis bibir dan batang hidung masih cukup terlihat.

### **2.1.5 Spesifikasi Jumlah Objek dalam 1 *frame***

Spesifikasi ini merupakan spesifikasi batasan jumlah wajah yang dapat dideteksi secara bersamaan (dalam 1 frame). Terdapat 5 pekerja di ruangan kerja tempat produk ini dilakukan pengujian. Penulis mengasumsikan bahwa kasus terburuk adalah bahwa setiap pekerja tersebut melakukan diskusi dan membawa masuk karyawan lainnya sejumlah 1 orang. Sehingga diperoleh Batasan spesifikasi jumlah objek dalam 1 frame yang digunakan adalah lebih dari 10 wajah.

### **2.1.6 Spesifikasi Jumlah Video Stream**

Spesifikasi ini merupakan spesifikasi batasan jumlah jalur video dari beberapa input kamera CCTV yang masih dapat berjalan dengan baik. Penulis mengacu pada jumlah ruangan yangberada pada lantai 15 kantor PT Tower Bersama Infrstruktur Group dimana tempat penulis bekerja. Terdapat 5 ruangan (2 ruang kerja dan 3 ruang meeting). Nilai ini dijadikan acuan oleh penulis sebagai jumlah jalur video CCTV yang masih dapat digunakan oleh system untuk berjalan dengan baik.

### **2.1.7 Spesifikasi Penyimpanan Data**

Spesifikasi ini merupakan spesifikasi untuk penyimpanan data hasil proses analytic system. Data yang dimasukan ke database merupakan data menggunakan masker atau tidak dari wajah yang terdeteksi. Namun perlu dilakukannya objek *tracking* sehingga untuk objek yang sama tidak terhitung 2 kali atau lebih. Sehingga data yang masuk ke database hanyalah data-data pada kondisi yang berbeda. Spesifikasinya adalah Sistem dapat menyimpan data yang perlu saja dimana objek dengan id dan label yang sama hanya dapat 1 kali tersimpan pada database. Id merupakan suatu kondisi Ketika objek berhasil dikenali dan dilakukan id-ing sehingga objek tersebut merupakan objek yang sama selama beberapa waktu. Label terdiri dari 2 kondidi yaitu Mask (wajah menggunakan masker) dan NoMask (wajah tanpa masker).

## 2.2 Tabel Spesifikasi Produk

Table 2.2-1 Spesifikasi Produk

No	Karakteristik Produk	Spesifikasi	Rincian
1	<ul style="list-style-type: none"> <li>- Mampu mendeteksi penggunaan masker pada kondisi ideal</li> <li>- Mampu mendeteksi penggunaan masker terhadap wajah yang menggunakan kacamata</li> <li>- Mampu mendeteksi semua jenis dan warna masker</li> </ul>	Akurasi Deteksi Objek	<ul style="list-style-type: none"> <li>- Produk memiliki akurasi lebih dari 80% untuk sampel <i>real</i> yang diambil</li> <li>- Produk memiliki akurasi lebih dari 94.4% untuk sampel data <i>dummy</i></li> </ul>
3	Pendeteksian dapat dilakukan dari jarak yang jauh untuk berbagai sisi wajah	Jarak Deteksi Objek	Objek wajah yang masih dapat dideteksi berjarak kurang dari 5 m dari lensa kamera
		Angle Sisi Wajah	Objek sisi wajah yang masih dapat dideteksi berada dalam rentang sudut $\pm 60^\circ$
		Ketinggian Kamera	Ketinggian lensa kamera kurang dari 2.3 m

4	Mampu mendeteksi banyak wajah dalam frame yang sama	Jumlah Objek dalam 1 <i>frame</i>	Wajah yang terdeteksi dalam 1 frame lebih dari 10 wajah
5	Mampu dielaborasikan untuk beberapa input kamera dalam satu waktu yang sama	Jumlah Video Stream	Sistem dapat menggunakan sampai dengan 5 <i>stream</i> kamera
6	Mampu menyimpan hasil pendekripsi masker wajah kedalam <i>database</i>	Penyimpanan Data	Sistem dapat menyimpan data yang perlu saja dimana objek dengan id dan label yang sama hanya dapat 1 kali tersimpan pada database

**Table 2.2-2 Karakteristik Produk**

No	Karakteristik Produk
1	Mampu mendeteksi penggunaan masker pada kondisi ideal
2	Mampu menyimpan hasil pendekripsi masker wajah kedalam <i>database</i>
3	Mampu mendeteksi banyak wajah dalam frame yang sama
4	Pendekripsi dapat dilakukan dari jarak yang jauh untuk berbagai sisi wajah
5	Mampu dielaborasikan untuk beberapa input kamera dalam satu waktu yang sama
6	Mampu mendeteksi semua jenis dan warna masker
7	Mampu mendeteksi penggunaan masker terhadap wajah yang menggunakan kacamata

## 2.3 Verifikasi

### 2.3.1 Spesifikasi Akurasi Deteksi Objek

Table 2.3-1 Verifikasi Spesifikasi Deteksi Objek

Hal	1. Akurasi
Rincian	1. $\geq 80\%$ Data Real 2. $\geq 94.4\%$ Data <i>Dummy</i>
Metode Pengukuran	Pengujian dilakukan pada wajah manusia berupa data <i>real</i> (gambar wajah yang diambil di kantor) dan data <i>dummy</i> (gambar wajah dengan masker dan tanpa masker dari internet)
Prosedur Pengujian	<ul style="list-style-type: none"><li>- Pengujian data <i>real</i> dilakukan dengan mengambil frame gambar pada ketinggian CCTV pada 2.3 meter dan jarak objek bervariasi pada rentang 0 sampai 5 meter (sesuai spesifikasi jarak dan ketinggian). Kemudian dicuplik frame selama 20 detik sekali kemudian dilihat hasil pendektsian. Hanya wajah dengan sudut kemiringan wajah <math>\pm 60^\circ</math> saja yang diperhitungkan pada perhitungan akurasi (sesuai spesifikasi <i>angle</i> wajah). Sampel yang diambil lebih dari 100 frame gambar tangkapan CCTV pada berbagai variasi posisi kamera. Dihitung akurasi berdasarkan rumus perhitungan akurasi model <i>machine learning</i> dengan parameter TP (True Positive), TN (True Negatuve), FP (False Positive), FN (False Negative).</li><li>- Pengujian data <i>dummy</i> dilakukan dengan memanfaatkan gambar-gambar wajah menggunakan dan tanpa masker dari internet. Diujikan pada lebih dari 100 gambar pada <i>angle</i> wajah yang berbeda-beda. Hanya wajah dengan sudut kemiringan wajah <math>\pm 60^\circ</math> saja yang diperhitungkan pada perhitungan akurasi (sesuai spesifikasi <i>angle</i> wajah). Dihitung akurasi berdasarkan rumus perhitungan akurasi model <i>machine learning</i> dengan parameter TP (True Positive), TN (True Negatuve), FP (False Positive), FN (False Negative).</li></ul>

### 2.3.2 Spesifikasi Jarak Deteksi Objek

Table 2.3-2 Verifikasi Spesifikasi Jarak Deteksi Objek

Hal	1. Jarak
Rincian	1. $< 5$ Meter
Metode Pengukuran	Pengujian dilakukan dengan menggunakan meteran digital (meteran laser)

Prosedur Pengujian	<ul style="list-style-type: none"> <li>- Pengujian dilakukan dengan menembakkan laser meteran ke arah lensa IP Camera pada ketinggian 2.3 meter (sesuai spesifikasi ketinggian) dari ujung sisi terjauh dari kamera pada ruangan kerja. Nilai yang terbaca pada meteran harus lebih dari atau sama dengan 5 meter agar spesifikasi jarak ini terpenuhi.</li> </ul>
--------------------	--

### 2.3.3 Spesifikasi Ketinggian Kamera

**Table 2.3-3 Verifikasi Spesifikasi Ketinggian Kamera**

Hal	1. Ketinggian
Rincian	1. $< 2.3$ Meter
Metode Pengukuran	Pengujian dilakukan dengan menggunakan meteran digital (meteran laser)
Prosedur Pengujian	<ul style="list-style-type: none"> <li>- Pengujian dilakukan dengan menembakkan laser meteran ke arah lensa IP Camera dari tanah dengan sudut kemiringan <math>0^\circ</math>. Nilai yang terbaca pada meteran harus lebih dari atau sama dengan 2.3 meter agar spesifikasi ketinggian ini terpenuhi.</li> </ul>

### 2.3.4 Spesifikasi Angle Deteksi Objek

**Table 2.3-4 Verifikasi Spesifikasi Angle Deteksi Objek**

Hal	1. Angle Sisi Wajah (a)
Rincian	$-60^\circ < a < 60^\circ$
Metode Pengukuran	Pengujian dilakukan dengan data <i>real</i> yaitu wajah penulis dari berbagai variasi sisi wajah.
Prosedur Pengujian	Pengujian dilakukan dengan menghadapkan wajah pada berbagai sisi. Pada pengujian data <i>real</i> , ini dilakukan pada kondisi ketinggian kamera 2.3 meter dan jarak objek maksimum adalah 5 meter (sesuai spesifikasi jarak dan ketinggian). Patokan 60 derajat ialah selama kedua landmark mata pada wajah masih terlihat.

### 2.3.5 Spesifikasi Jumlah Objek dalam 1 frame

**Table 2.3-5 Verifikasi Jumlah Objek dalam 1 Frame**

Hal	1. Jumlah Objek dalam 1 frame
Rincian	$> 10$ wajah
Metode Pengukuran	Pengujian dilakukan dengan data <i>dummy</i> (gambar wajah dengan masker dan tanpa masker dari internet)
Prosedur Pengujian	Dipersiapkan terlebih dahulu gambar-gambar yang dalam gambar tersebut lebih dari 10 wajah. Diujikan program pada input gambar tersebut dan dilihat apakah program dapat mendeteksi lebih dari 10 wajah atau tidak. Parameter

	keberhasilannya ketika lebih dari 10 wajah dapat dideteksi dengan baik. Diujikan pada 5 gambar.
--	---

### 2.3.6 Spesifikasi Jumlah Video Stream

**Table 2.3-6 Verifikasi Spesifikasi Jumlah Video Stream**

Hal	1. Jumlah Video Stream
Rincian	1. > 5 Video Stream
Metode Pengukuran	Pengujian dilakukan dengan meninjau pemakaian GPU, CPU, dan RAM computer terhadap 1 video stream
Prosedur Pengujian	Dijalankan program <i>face-mask detection</i> untuk input 1 IP Camera. Kemudian dilihat pada persentase pemakaian GPU, CPU, dan RAM. Untuk masing-masing komponen tersebut dikalikan dengan 5 maka hasilnya tidak boleh lebih dari 80% pemakaian total komponen (85% GPU, 80% CPU, 80% RAM). Dengan kata lain diberikan <i>spare</i> sebesar 20% agar computer tidak mengalami <i>overload</i> .

### 2.3.7 Spesifikasi Penyimpanan Data

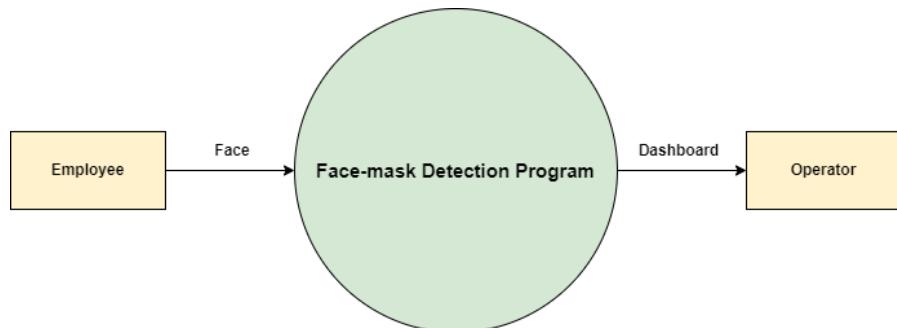
**Table 2.3-7 Verifikasi Spesifikasi Penyimpanan Data**

Hal	1. Penyimpanan Data
Rincian	1. Data dengan id dan label yang sama hanya dapat 1 kali tersimpan pada database
Metode Pengukuran	Pengujian dilakukan dengan menjalankan program <i>face-mask detection</i> kemudian melihat data hasil deteksi pada database
Prosedur Pengujian	Dijalankan program <i>face-mask detection</i> . Data hasil deteksi program kemudian tersimpan pada database. Ditinjau lebih lanjut apakah terdapat 2 data dengan id dan label yang sama tersimpan pada database. Diujikan pada 20 data yang tersimpan.

### 3 B300 DESAIN SISTEM

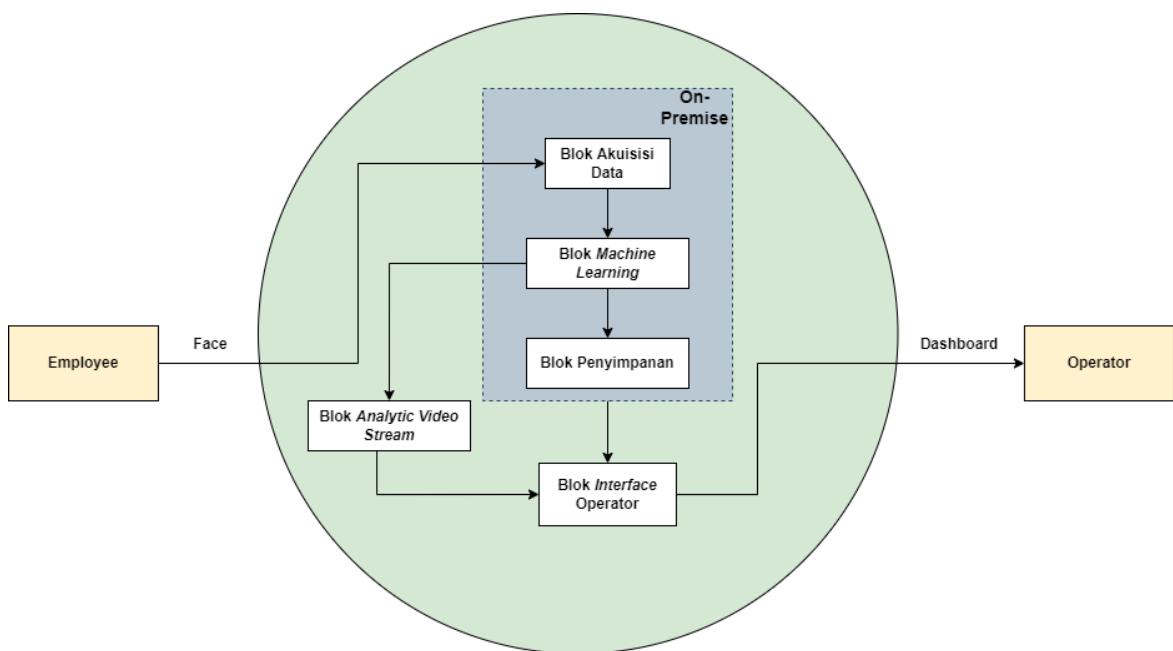
#### 3.1 Konsep Sistem

##### 3.1.1 Arsitektur Utama



Gambar 3.1-1 DFD Level 0 Sistem

Dapat dilihat pada Gambar 3.1-1 yaitu Data Flow Diagram (DFD) level 0 sistem. Terdapat 2 bagian penting yaitu Employee sebagai input sistem dan operator sebagai pihak yang menggunakan output dari sistem. Input yang dibawa merupakan sebuah frame gambar hasil tangkapan modul kamera. Selanjutnya hasil tangkapan gambar tersebut dideteksi oleh program terkait menggunakan atau tidaknya masker oleh karyawan. Output yang dihasilkan akan berupa web-based dashboard yang menunjukkan grafik, tabel, live-stream modul kamera, dll, yang bertujuan untuk menampilkan hasil rekapitulasi data keseluruhan program kepada operator.



Gambar 3.1-2 DFD Level 1

Dapat dilihat pada Gambar 3.1-2 yaitu DFD level 1 sistem. Dijelaskan lebih detail mengenai blok penyusun program *Face-mask Detection*. Terdapat 5 blok penyusun program tersebut. Berikut adalah penjelasan tiap bloknya.

### 1) Blok Akuisi Data

Blok ini digunakan untuk memperoleh data yang diperlukan sistem untuk berfungsi dengan baik. Data yang diperoleh berupa frame gambar yang ditangkap oleh modul kamera. Data ini akan digunakan blok machine learning untuk dilakukan deteksi objek.

### 2) Blok Machine Learning

Blok ini digunakan untuk melakukan deteksi dari frame gambar hasil tangkapan blok akuisisi data. Blok ini memanfaatkan pemodelan machine learning yang telah dilakukan training dari dataset yang sudah dipersiapkan. Output dari blok ini adalah label, id, dan timestamp untuk dilakukan proses penyimpanan data pada blok penyimpanan. Selain itu, dihasilkan juga koordinat bounding box objek yang terdeteksi untuk dapat digunakan pada blok analytic video stream. Di dalam blok ini juga ditambahkan sub-blok Multiple Object Tracking untuk melakukan tracking terhadap objek yang dideteksi.

### 3) Blok Analytic Video Stream

Blok ini digunakan untuk menampilkan video secara real-time dari rekaman kamera yang sudah digabungkan dengan proses machine learning. Sehingga pada video sudah menampilkan bounding box dan klasifikasi dari objek-objek yang terdeteksi pada frame.

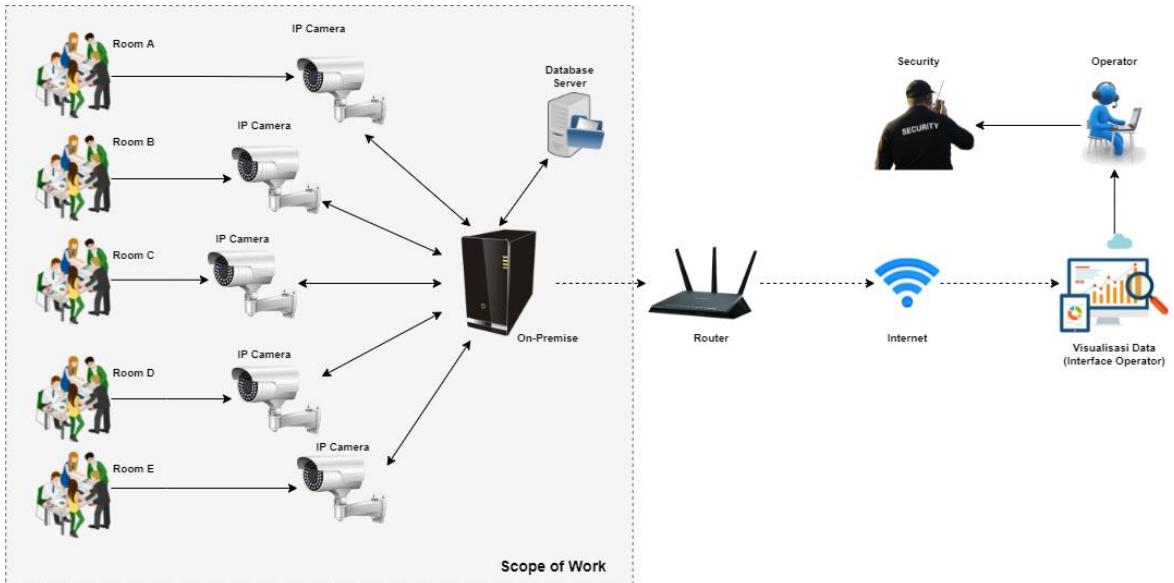
### 4) Blok Penyimpanan

Blok ini digunakan untuk melakukan penyimpanan data secara local pada on-premise komputer hasil proses machine learning berupa label, id, dan timestamp. Penyimpanan dilakukan lokal pada on-premise komputer.

### 6) Blok Interface Operator

Blok ini digunakan untuk menampilkan hasil rekap data keseluruhan dalam bentuk table, grafik, dsb. Data yang diambil dari blok ini merupakan data yang disimpan pada blok penyimpanan server. Disisipkan juga tempat untuk menampilkan live-streaming video dari blok Analytic Video Stream.

#### 3.1.2 Topologi

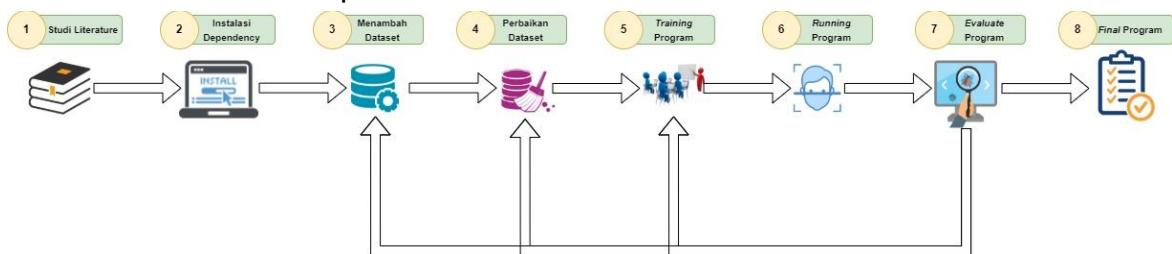


Gambar 3.1-3 Topologi Sistem

Dapat dilihat pada Gambar 3.1-3 yaitu topologi system keseluruhan. Terlihat terdapat 5 IP Camera yang terhubung dengan computer. Jumlah ini sesuai dengan batasan pada spesifikasi jumlah video stream. Masing-masing kamera tersebut di tempatkan di tiap-tiap ruangan tempat karyawan beraktivitas. Data hasil pemrosesan deteksi penggunaan masker oleh computer disimpan pada *database* local computer. Database dengan computer berkomunikasi secara 2 arah dimana arah dari computer ke database ialah menyimpan data-data hasil proses deteksi sedangkan arah sebaliknya yaitu dari database ke computer digunakan melakukan pengiriman data untuk visualisasi data ke *dashboard* operator. *Dashboard* operator ini merupakan *dashboard* berbasis *website* (Web-based Dashboard). Operator dapat menggunakan *dashboard* ini untuk melakukan monitor terhadap penggunaan masker karyawan pada ruangan kerja tertentu. Apabila persentase *threshold* tidak menggunakan masker terlewati maka operator dapat menghubungi petugas penganan protocol Kesehatan untuk menegur karyawan pada ruangan tersebut agar segera menggunakan maskernya.

Selain itu dapat dilihat mengenai *Scope of Work* penulis, terlihat bahwa penulis hanya mengerjakan sampai bagian data tersimpan ke database saja (*Back-end*). Penulis hanya berfokus pada perancangan model *Machine Learning* agar dihasilkan program sesuai dengan spesifikasi yang diinginkan dan dapat menyimpan hasil pendektsiannya ke database. Permasalahan *front-end* seperti pembuatan *dashboard* akan dikerjakan oleh penerus projek ini.

### 3.1.3 Flow Process Implementasi Sistem



Gambar 3.1-4 Flow Process Implementasi Sistem

Dapat dilihat pada Gambar 3.1-4 yaitu flow process implementasi system. Diagram tersebut merupakan seluruh tahapan yang akan dilakukan oleh penulis untuk mengembangkan program *face-mask detection*. Terdapat 8 tahapan dalam proses implementasinya. Berikut adalah penjelasan tiap tahapnya.

#### 1. Studi Literatur

Pada tahap ini dipelajari bagaimana merancang sebuah projek *machine learning* dan *computer vision* yang meliputi *library* apa saja yang harus di-instal, algoritma dan *framework* apa saja yang dapat digunakan untuk mengembangkan pemodelan *machine learning*, bahasan pemrograman yang dapat digunakan, system database, dsb. Selain itu dipelajari juga *operating system* yang dapat digunakan dan berkaitan dengan *resource* yang ada.

#### 2. Instalasi Dependency

Setelah mengetahui *tools* apa saja yang akan digunakan untuk mengembangkan projek, dilakukan instalasi terhadap *tools* tersebut. Dilakukan instalasi *library*, *framework*, *software*, dsb.

### 3. Menambah Dataset

Untuk melakukan *training* pemodelan *machine learning*, salah satu komponen paling krusial adalah dataset. Dataset yang ada merupakan gambar-gambar wajah menggunakan masker dan tidak menggunakan masker. Karena pada spesifikasi diinginkan wajah dapat dideteksi dari berbagai sudut hingga batas kemiringan tertentu, sehingga dataset yang ada juga harus memiliki variasi terhadap *angle* wajah. Dataset yang dipersiapkan dapat diperoleh melalui platform-platform gratis di internet atau dapat mengambil secara mandiri dengan men-*capture* wajah-wajah orang yang ada di sekitar.

### 4. Perbaikan Dataset

Perbaikan dataset disini merupakan proses untuk memperbaiki dataset yang sudah dipersiapkan pada tahap selanjutnya. Dapat dilakukan dengan cara melakukan augmentasi dataset. Augmentasi merupakan proses penambahan dataset dari dataset yang sudah ada dengan cara memvariasikannya. Dapat dilakukan dengan melakukan rotasi, translasi, mengubah resolusi, flip, blur, memberikan noise, mengubah warna, dsb. Hal ini bertujuan agar pemodelan hasil training dapat akurat dalam mendeteksi objek.

Pada tahap ini juga dilakukan pelabelan dataset. Pelabelan adalah proses membuat *bounding box* sebagai *Region of Interest* (RoI) dari dataset yang sudah siap digunakan. Pelabelan merupakan ciri khas dari program *machine learning* tipe *supervised learning*. Akan dihasilkan file anotasi dari hasil pelabelan. ekstensi file anotasi ini berbeda-beda bergantung pada algoritma *machine learning* yang digunakan. Contohnya apabila algoritma yang digunakan adalah YOLO (You Only Look Once) maka file anotasi hasil pelabelan yang akan dihasilkan memiliki ekstensi .txt. Sedangkan apabila algoritma yang digunakan adalah SSD (Single Shot Detector) maka file anotasi hasil pelabelan yang akan dihasilkan memiliki ekstensi .xml.

### 5. Training Program

Setelah mempersiapkan dan memperbaiki seluruh dataset, dilakukan *training* untuk menghasilkan pemodelan *machine learning* yang akan digunakan program untuk melakukan proses deteksi. Proses *training machine learning* memakan waktu cukup lama bergantung dengan perangkat computer yang digunakan. Terdapat beberapa alternatif *framework* untuk mengembangkan program *machine learning* seperti Tensorflow, PyTorch, Darknet, dsb. Selain itu, terdapat juga pilihan algoritma *machine learning* seperti SSD (Single Shot Detector), YOLO (You Only Look Once), dsb. Parameter yang biasanya dijadikan acuan memilih *framework* dan algoritma *machine learning* adalah akurasi, *latency*, dan perangkat yang akan digunakan. *Training* memanfaatkan file yang disebut *pre-trained model*. *Pre-trained model* merupakan pemodelan *machine learning* yang sudah dirancang sebelumnya oleh pengembang program pada dataset yang sangat banyak. Selanjutnya *pre-trained* model ini dapat digunakan oleh pengembang lain untuk digunakan membuat pemodelan *machine learning* lainnya berdasarkan dataset yang sudah dipersiapkan tersebut.

### 6. Running Program

Setelah proses *training* selesai, akan dihasilkan file pemodelan *machine learning* . File ini digunakan untuk proses *running* program *face-mask detection*. Pada proses ini dibuatkan program akuisisi data yaitu mengambil frame gambar dari modul kamera dan mendeteksi frame tersebut menggunakan model *machine learning* yang telah dibuat sebelumnya. Pada proses ini juga dibuatkan program penyimpanan database hasil pendekripsi, program *Multiple Object Tracking* (MOT), konfigurasi warna modul kamera, dll.

## 7. Evaluate Program

Proses evaluasi program merupakan proses peninjauan kualitas pemodelan *machine learning* yang sudah dilatih sebelumnya. Disini diujikan secara langsung dengan menjalankan program dan diuji kekurangan-kekurangan apa saja yang masih dapat diminimalisir. Dapat dilakukan seperti menambah dataset untuk variasi-variasi sisi wajah, warna masker, memperbaiki pelabelan dataset, memperbanyak iterasi *training*, mengatur Kembali konfigurasi warna modul kamera, modifikasi pada program, dsb.

## 8. Final Program

Final program merupakan program akhir Ketika dirasa program sudah memenuhi spesifikasi yang diinginkan.

### 3.2 Desain Sistem

#### 3.2.1 Framework Machine Learning



Gambar 3.2-1 PyTorch

Sejak dimulai oleh tim Facebook AI Research (FAIR) pada tahun 2017, PyTorch telah menjadi framework yang sangat populer dan efisien untuk membuat model Deep Learning (DL). Pustaka *open-source machine learning* ini didasarkan pada Torch dan dirancang untuk memberikan fleksibilitas yang lebih besar dan peningkatan kecepatan untuk implementasi *deep neural network*. Saat ini, PyTorch adalah *library* paling disukai untuk peneliti dan praktisi AI (Artificial Intelligence) di seluruh dunia dalam industri dan akademisi.

PyTorch adalah library tensor Deep Learning yang dioptimalkan berdasarkan Python dan Torch terutama digunakan untuk aplikasi yang menggunakan GPU dan CPU. PyTorch lebih disukai daripada *framework* Deep Learning lainnya seperti TensorFlow dan Keras karena menggunakan grafik komputasi dinamis dan sepenuhnya Pythonic. Hal ini memungkinkan ilmuwan, pengembang, dan debugger *neural network* untuk menjalankan dan menguji bagian dari kode secara real-time. Dengan demikian, pengguna tidak perlu menunggu seluruh kode diimplementasikan untuk memeriksa apakah sebagian kode berfungsi atau tidak.

### 3.2.2 Algoritma Machine Learning



Gambar 3.2-2 YOLO Object Detection

YOLO adalah singkatan dari istilah 'You Only Look Once'. Ini adalah algoritma yang mendeteksi dan mengenali berbagai objek dalam gambar (secara real-time). Deteksi objek di YOLO dilakukan sebagai masalah regresi dan memberikan probabilitas kelas dari gambar yang terdeteksi. Algoritma YOLO menggunakan *Convolutional Neural Network* (CNN) untuk mendeteksi objek secara real-time. Seperti namanya, algoritma ini hanya membutuhkan propagasi maju tunggal melalui *neural network* untuk mendeteksi objek. Artinya prediksi pada keseluruhan citra dilakukan dalam satu algoritma yang dijalankan. CNN digunakan untuk memprediksi berbagai probabilitas kelas dan *bounding box* secara bersamaan. Algoritma YOLO terdiri dari berbagai varian. Pada proyek *face-mask detection* ini, yang digunakan adalah YOLOv5.

Algoritma YOLO bekerja menggunakan tiga teknik berikut:

#### A. Residual Blocks

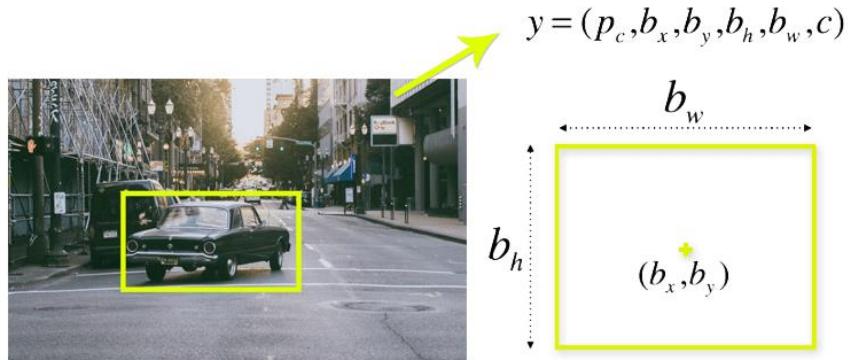
Pertama, gambar dibagi menjadi berbagai grid. Setiap grid memiliki dimensi  $S \times S$ . Gambar berikut menunjukkan bagaimana gambar input dibagi menjadi beberapa grid.



Gambar 3.2-3 Residual Blocks

Pada gambar di atas, ada banyak sel grid dengan dimensi yang sama. Setiap sel grid akan mendeteksi objek yang muncul di dalamnya. Misalnya, jika pusat objek muncul dalam sel grid tertentu, maka sel ini akan bertanggung jawab untuk mendeteksinya.

### B. Bounding Box Regression



**Gambar 3.2-4 Bounding Box Regression**

Setiap *bounding box* pada gambar merepresentasikan parameter sebagai berikut :

- Width ( $b_w$ )
- Height ( $b_h$ )
- Class (e.g. person, car, traffic light, etc) – ini merepresentasikan parameter  $c$
- Bounding box center ( $b_x, b_y$ )

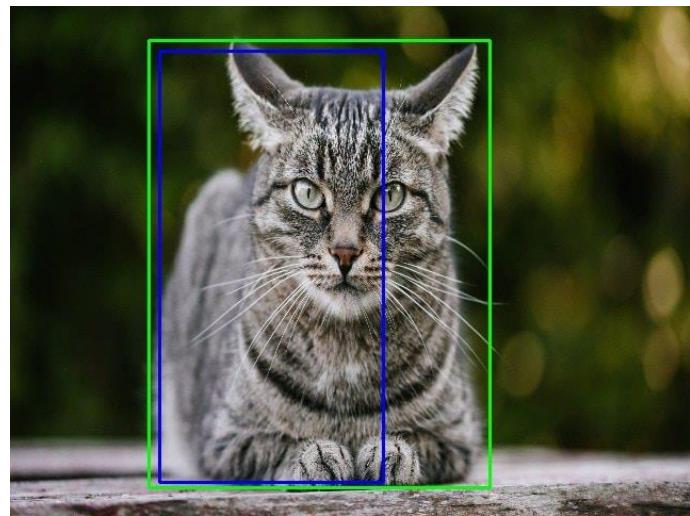
YOLO menggunakan regresi *bounding box* tunggal untuk memprediksi ketinggian, lebar, pusat, dan kelas objek. Pada gambar di atas, direpresentasikan probabilitas suatu objek yang muncul.

### C. Intersection Over Union (IOU)

Intersection over union (IOU) adalah fenomena dalam deteksi objek yang menggambarkan bagaimana kotak beririsan. YOLO menggunakan IOU untuk membentuk bounding box yang mengelilingi objek dengan sempurna.

Setiap sel grid bertanggung jawab untuk memprediksi kotak pembatas dan skor *confidence*. IOU sama dengan 1 jika kotak pembatas yang diprediksi sama dengan kotak sebenarnya. Mekanisme ini menghilangkan kotak pembatas yang tidak sama dengan kotak sebenarnya.

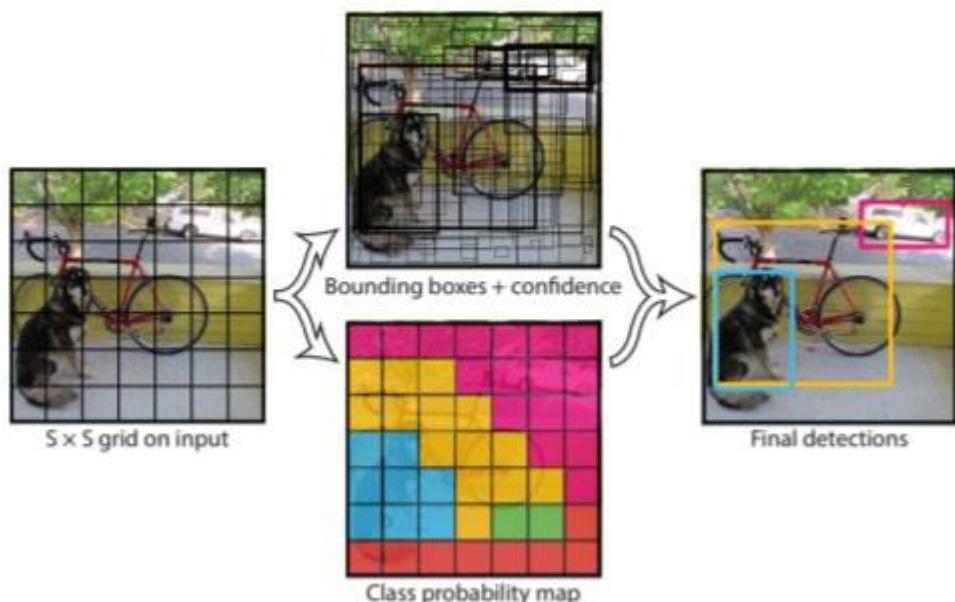
Gambar berikut memberikan contoh sederhana tentang cara kerja IOU.



Gambar 3.2-5 Intersection Over Union (IOU)

Pada gambar di atas, ada dua kotak pembatas, satu berwarna hijau dan yang lainnya berwarna biru. Kotak biru adalah kotak prediksi sedangkan kotak hijau adalah kotak asli. YOLO memastikan bahwa kedua kotak pembatas itu sama sehingga hanya kotak hijau saja yang dipertimbangkan.

Gambar berikut adalah bagaimana ketiga Teknik yang sudah dijelaskan sebelumnya dikombinasikan.



Gambar 3.2-6 Kombinasi Teknik YOLO

### 3.2.3 Software

#### 3.2.3.1 LabelImg

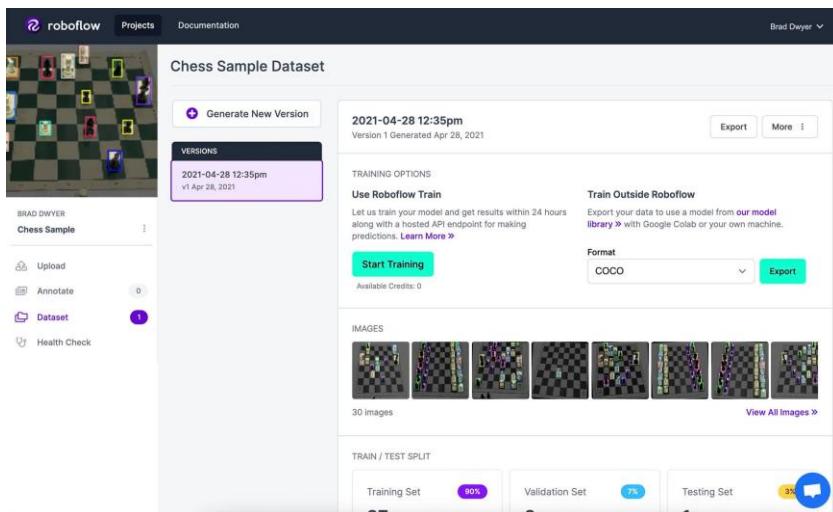


Gambar 3.2-7 Tampilan labelImg

LabelImg adalah aplikasi *open-source* untuk melabeli gambar secara grafis. Aplikasi ditulis dengan Python dan menggunakan QT untuk antarmuka grafisnya. Ini merupakan cara mudah dan gratis untuk memberi label pada beberapa gambar untuk mencoba proyek pendektsian objek.

Pada proyek ini, akan dilakukan pelabelan terhadap dataset wajah menggunakan masker (kelas Mask) dan data wajah tanpa masker (kelas NoMask). Karena digunakan algoritma *machine learning* YOLO, sehingga output hasil pelabelan dataset merupakan file dengan ekstensi .txt.

#### 3.2.3.2 Roboflow



Gambar 3.2-8 Tampilan Roboflow

Roboflow merupakan aplikasi computer vision yang bertujuan untuk melakukan *pre-processing* dan model *training* dataset. Roboflow memiliki kumpulan data publik yang tersedia bagi pengguna dan juga pengguna dapat mengunggah data kustom mereka sendiri. Roboflow menerima berbagai format anotasi. Dalam pra-pemrosesan data, dapat dilakukan beberapa teknik seperti orientasi gambar, pengubahan ukuran, kontras, dan augmentasi data. Pada proyek ini, roboflow penulis gunakan hanya untuk melakukan *pre-processing* dan augmentasi dataset saja.

### 3.2.3.3 XAMPP



Gambar 3.2-9 XAMPP

XAMPP merupakan singkatan dari cross(X)-platform, Apache(A), MariaDB(M), PHP(P), dan Perl(P). Aplikasi ini bersifat *open-source* yang menyediakan pengiriman Apache untuk berbagai server dan perintah-perintah yang dapat di eksekusi beserta API Apache, MariaDB, PHP, dan Perl.

XAMPP mengizinkan host atau server local untuk melakukan validasi terhadap computer. Ini merupakan *framework* yang menyediakan lingkungan yang sesuai untuk menguji dan memverifikasi fungsionalitas proyek berdasarkan Apache, MySQL, PHP, dan Perl menggunakan kerangka kerja host. Pada proyek ini, penulis menggunakan MySQL sebagai database server untuk menyimpan hasil deteksi objek. Untuk melihat dalam GUI, digunakan PHPMyadmin agar data-data yang ditampilkan lebih mudah dilihat dan dipahami.

### 3.2.4 Hardware

#### 3.2.4.1 Komputer



Gambar 3.2-10 On-premise Komputer

**Table 3.2-1 Spesifikasi On-premise Komputer**

Spesifikasi	
CPU	Intel Core i7-9700F 3.0 Ghz Up To 4.7 Ghz – Cache 12 MB [Box] Socket LGA 1151V2 – Coffeeflake Series
GPU	MSI GeForce RTS 2070 SUPER 8 GB DDR6 – Ventus OC1
ROM	WDC 1TB SATA3 64MB - Blue
RAM	DDR4 PC24000 3000MHz Dual Channel 16 GB (2X18GB)
<i>Operating System</i>	Linux Ubuntu

### 3.2.4.2 Modul Kamera



**Gambar 3.2-11 Modul Kamera**

**Table 3.2-2 Spesifikasi Modul Kamera**

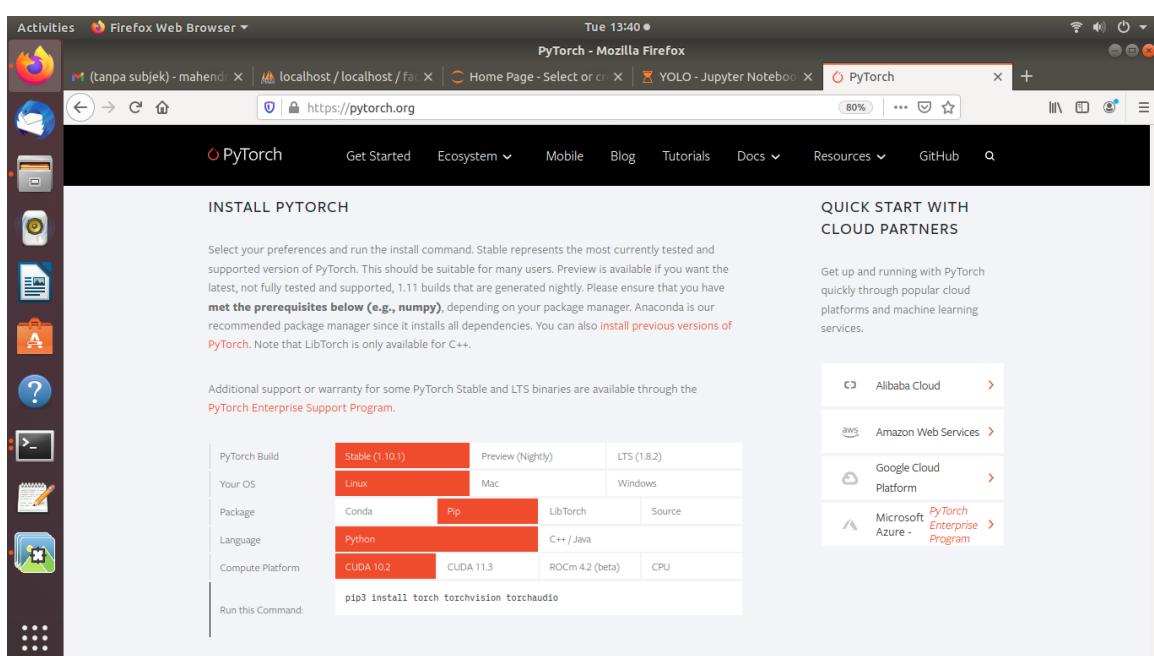
Nama Barang	Spesifikasi
Hikvision IP camera IR mini Bullet Cam DS-2CD1002-I	<ul style="list-style-type: none"><li>○ 1.0 Megapixel CMOS IR Bullet Camera</li><li>○ 1/4 Progressive CMOS / 25(P)/30(N)fps @ 1280 720</li><li>○ 0.01Lux @ F1.2/3D-DNR</li><li>○ Lens 4mm @F2.0,Angle of View : 73.1 (6/8/12mm Optional)</li><li>○ True Day &amp; Night / IR disance :30 m / IP66</li><li>○ DC 12V, 6W MAX / POE )802.3af)</li></ul>

## 4 B400 IMPLEMENTASI

### 4.1 Instalasi Dependencies

#### 4.1.1 Instalasi PyTorch

Framework *Machine Learning* yang digunakan pada proyek ini adalah PyTorch. PyTorch merupakan *library* yang digunakan untuk aplikasi *Deep Learning* menggunakan GPU dan CPU. Ini merupakan *library machine learning* yang bersifat *open-source* untuk Bahasa pemrograman python. *Library* ini dikembangkan oleh tim riset AI Facebook. Untuk melakukan instalasi library, perlu diketahui terlebih dahulu spesifikasi *Operating System*, *Package*, Bahasa Pemrograman, dan *Compute Platform* dari komputer yang digunakan. Langkah pertama untuk instalasi adalah dengan mengakses website PyTorch di <https://pytorch.org>. Berikut adalah tampilan website PyTorch.



Gambar 4.1-1 Tampilan Website PyTorch

Dapat dilihat pada Gambar 4.1-1 yaitu halaman web PyTorch, bagian PyTorch Build pilih opsi Stable (1.10.1). *Operating System* komputer yang digunakan merupakan Linux. *Environment Package* yang digunakan untuk proses instalasi merupakan Pip sehingga bahasa pemrograman yang dipilih adalah Python. Untuk mengetahui informasi *Compute Platform* komputer yang digunakan, buka terminal dan tuliskan kode perintah seperti berikut.

```
nvidia-smi
```

A screenshot of a terminal window titled 'Terminal'. The title bar shows 'Activities Terminal'. The terminal window displays the command 'nvidia-smi' and its output. The output includes system information like 'cctv@cctv-analytic:~\$ nvidia-smi' and performance metrics for multiple GPUs.

Gambar 4.1-2 Perintah Mengetahui Informasi GPU Komputer

Berikut adalah gambar tampilan informasi GPU komputer yang digunakan.

```

Tue 13:56 •
cctv@cctv-analytic: ~

File Edit View Search Terminal Help
cctv@cctv-analytic:~$ nvidia-smi
Tue Jan 4 13:55:35 2022
+-----+
| NVIDIA-SMI 440.59     Driver Version: 440.59      CUDA Version: 10.2 |
| Persistence-M| Bus-Id      Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap| Memory-Usage | GPU-Util Compute M. |
+-----+
| 0  GeForce RTX 207... Off | 00000000:01:00.0 On |          N/A |
| 24% 27C   P8    9W / 215W |  226MiB /  7981MiB |      2% Default |
+-----+
Processes:
GPU PID Type Process name          GPU Memory Usage
+-----+
0 2717 G  /usr/lib/xorg/Xorg          102MiB
0 5179 G  /usr/bin/gnome-shell        115MiB
0 6404 G  /usr/lib/firefox/firefox    2MiB
0 19077 G  /usr/lib/firefox/firefox    2MiB

```

Gambar 4.1-3 Tampilan Informasi GPU Komputer

Dapat dilihat pada Gambar 4.1-3 yaitu tampilan informasi GPU computer. Informasi *Compute Platform* terletak pada bagian atas kanan (lingkaran merah). Pada kasus ini, digunakan CUDA versi 10.2.

Sehingga pada bagian bawah Gambar 4.1-1, diperoleh kode perintah untuk dituliskan pada terminal. Berikut adalah kode perintah pada terminal untuk melakukan instalasi PyTorch.

```
Pip3 install torch torchvision torchaudio
```

```

Tue 14:12 •
cctv@cctv-analytic: ~

File Edit View Search Terminal Help
cctv@cctv-analytic:~$ pip3 install torch torchvision torchaudio
Requirement already satisfied: torch in ./virtualenvs/yolo/lib/python3.6/site-packages (1.8.2+cu102)
Requirement already satisfied: torchvision in ./virtualenvs/yolo/lib/python3.6/site-packages (0.9.2+cu102)
Requirement already satisfied: torchaudio in ./virtualenvs/yolo/lib/python3.6/site-packages (0.8.2)
Requirement already satisfied: numpy in ./virtualenvs/yolo/lib/python3.6/site-packages (from torch) (1.19.5)
Requirement already satisfied: typing-extensions in ./virtualenvs/yolo/lib/python3.6/site-packages (from torch) (3.10.0.2)
Requirement already satisfied: dataclasses; python_version < "3.7" in ./virtualenvs/yolo/lib/python3.6/site-packages (from torch) (0.8)
Requirement already satisfied: pillow>=4.1.1 in ./virtualenvs/yolo/lib/python3.6/site-packages (from torchvision) (8.4.0)
WARNING: You are using pip version 20.0.2; however, version 21.3.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
(yolo) cctv@cctv-analytic:~$ 
```

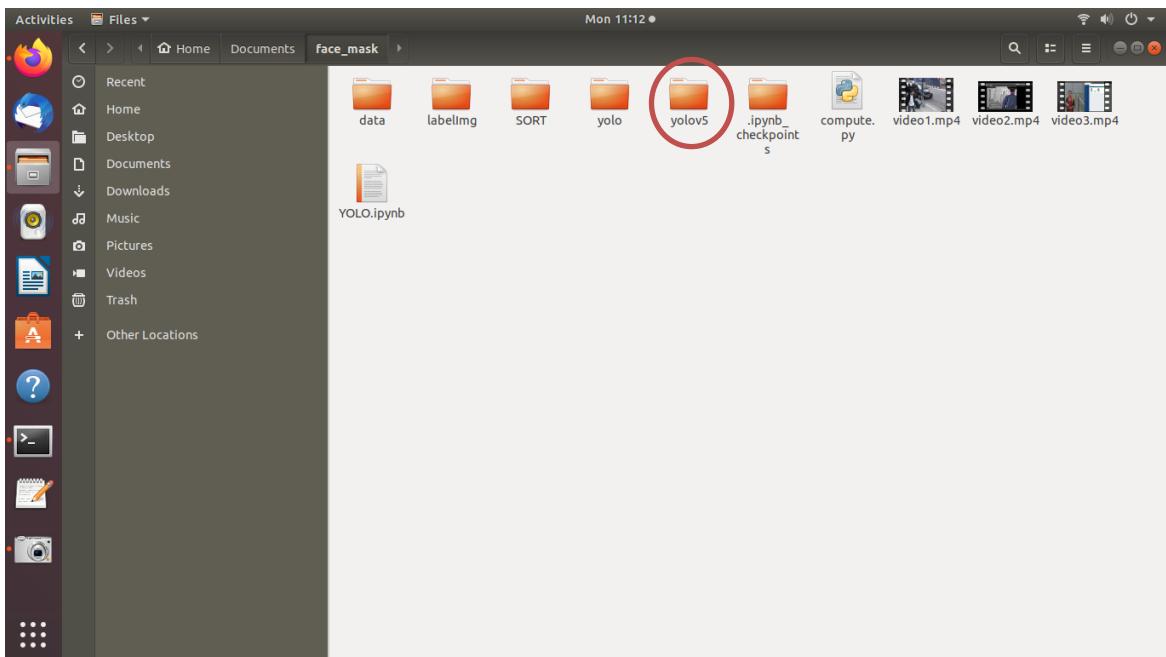
Gambar 4.1-4 Instalasi PyTorch pada Terminal

Dapat dilihat pada gambar 4.1-4, karena tidak terdapat notifikasi error dari terminal, proses instalasi PyTorch sudah berhasil dilakukan.

#### 4.1.2 Instalasi Yolov5

Pada proyek *Face-mask Detection* ini, digunakan YOLO (You Only Look Once) sebagai algoritma deteksi objek. Ini adalah algoritma yang mendeteksi dan mengenali berbagai objek dalam sebuah gambar (secara real-time). Deteksi objek di YOLO dilakukan sebagai masalah regresi dan memberikan probabilitas kelas dari gambar yang terdeteksi.

Versi YOLO yang digunakan adalah YOLOv5. Versi YOLOv5 adalah keluarga arsitektur dan model deteksi objek yang telah dilatih sebelumnya pada kumpulan COCO Dataset. Berikut adalah link untuk melakukan *clone* repositori YOLOv5 : <https://github.com/ultralytics/yolov5> . Setelah melakukan *download* repositori pada link tersebut, diekstrak folder hasil *download*. Pada kasus ini, folder yolov5 disimpan di alamat **/Documents/face\_mask** . Berikut adalah tampilan folder YOLOv5 yang telah di-*download*.

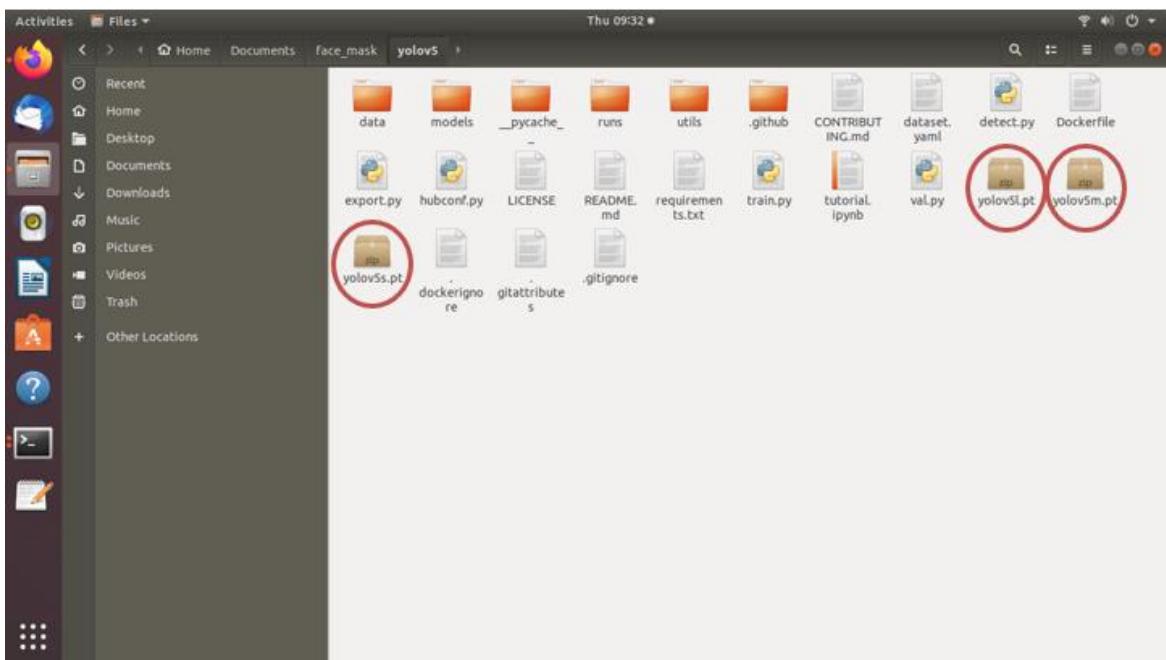


Gambar 4.1-5 Tampilan Folder yolov5

Setelah berhasil melakukan *clone* folder yolov5, proses *clone* belum sepenuhnya selesai dilakukan. Terdapat tahapan lanjutan yaitu melakukan *clone* dari YOLOv5 *Pre-Trained Model*. Model ini merupakan suatu file dengan ekstensi .pt yang merupakan file hasil pelatihan *Machine Learning* dengan dataset dari COCO dataset. Model ini dapat juga digunakan untuk membuat pemodelan *machine learning* lain yang salah satunya digunakan untuk membuat proyek *face-mask detection* ini.

Terdapat beberapa opsi file pemodelan YOLOv5 yang dapat digunakan diantaranya yolov5s.pt, yolov5m.pt, yolov5l.pt, dan yolov5x.pt, dll. Pada proyek ini, *pre-trained model* yang digunakan adalah yolov5s.pt. Berikut adalah link untuk men-*download pre-trained model* YOLOv5 : <https://github.com/ultralytics/yolov5/releases> .

Setelah file berhasil di-*download*, diekstrak file tersebut dan disimpan pada alamat **/Documents/face\_mask/yolov5**. Berikut adalah tampilan *pre-trained* model yolov5 yang telah di-*download*.



Gambar 4.1-6 Tampilan File *Pre-trained Model YOLOv5*

#### 4.1.3 Instalasi *Dependencies* untuk *Training* dan *Running Machine Learning Model*

Terdapat beberapa *library* lain yang perlu dipastikan sudah terinstall untuk dapat melakukan *training* dan *running* program. Berikut adalah kode program untuk melakukan instalasi *library* yang diperlukan.

```
# pip install -r requirements.txt

# Base -----
matplotlib>=3.2.2
numpy>=1.18.5
opencv-python>=4.1.2
Pillow>=7.1.2
PyYAML>=5.3.1
requests>=2.23.0
scipy>=1.4.1
torch>=1.7.0
torchvision>=0.8.1
tqdm>=4.41.0

# Logging -----
tensorboard>=2.4.1
# wandb

# Plotting -----
pandas>=1.1.4
seaborn>=0.11.0

# Export -----
# coremltools>=4.1 # CoreML export
# onnx>=1.9.0 # ONNX export
# onnx-simplifier>=0.3.6 # ONNX simplifier
# scikit-learn==0.19.2 # CoreML quantization
```

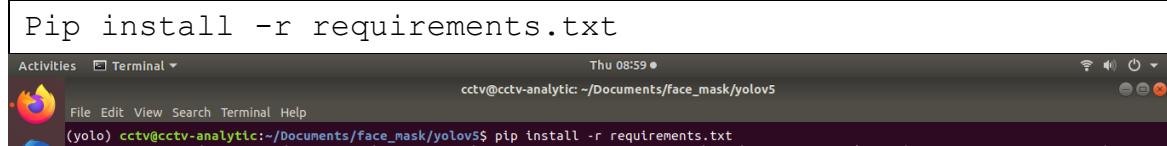
```

# tensorflow>=2.4.1 # TFLite export
# tensorflowjs>=3.9.0 # TF.js export

# Extras -----
# albumentations>=1.0.3
# Cython # for pycocotools
https://github.com/cocodataset/cocoapi/issues/172
# pycocotools>=2.0 # COCO mAP
# roboflow
thop # FLOPs computation

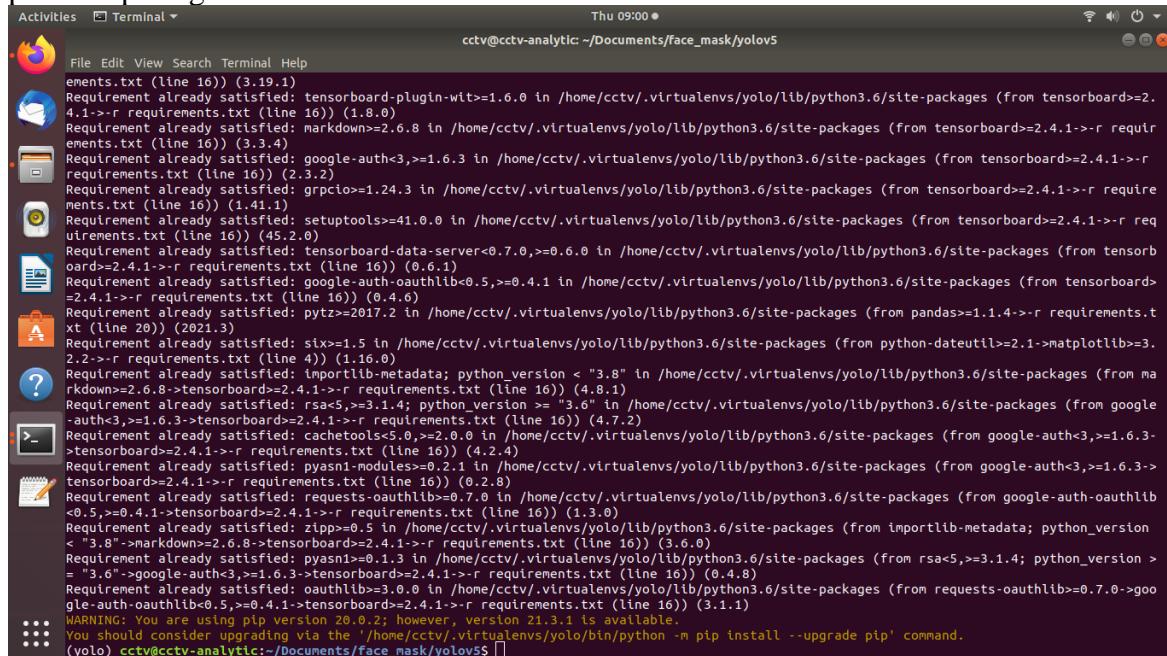
```

Program diatas disimpan dengan nama requirements.txt. Sehingga untuk melakukan proses instalasi seluruh kode diatas, hanya diperlukan satu baris perintah saja yang mengacu pada file requirements.txt. Berikut adalah kode perintah untuk melakukan instalasi *dependenciesnya*.



**Gambar 4.1-7 Perintah Instalasi Dependencies untuk Training dan Running Machine Learning Model**

Perintah pada Gambar 4.1-7 diatas menginstruksikan untuk menjalankan instalasi yang ada pada file bernama requirements.txt. Berikut adalah tampilan proses instalasi dari kode perintah pada gambar 4.1-7 diatas.



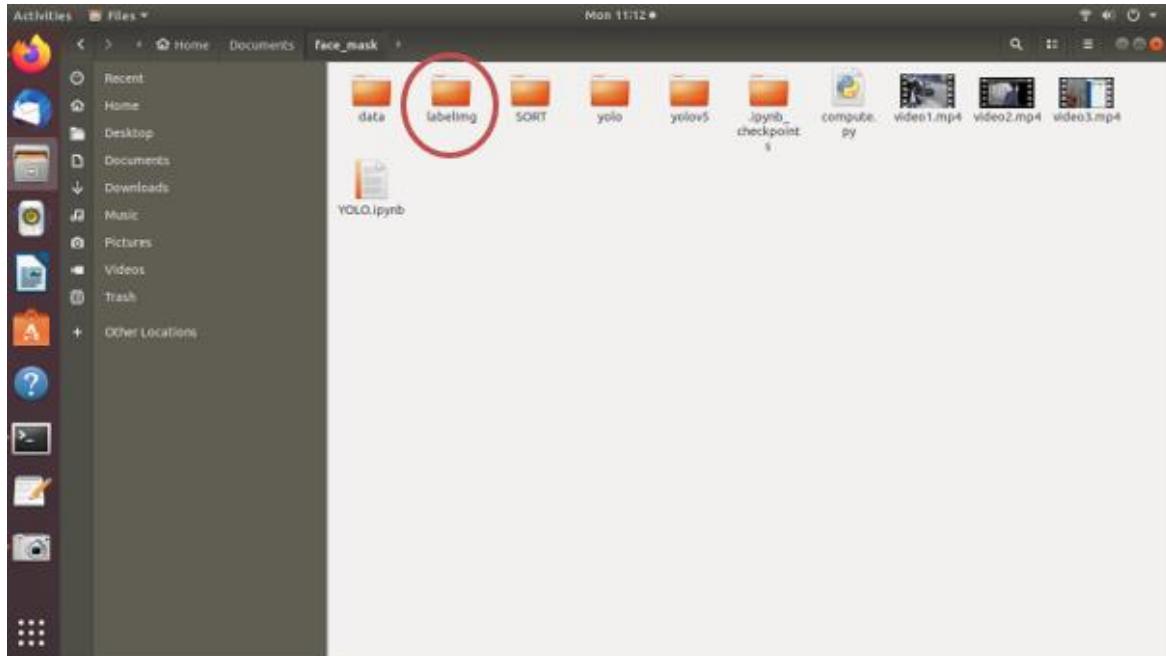
**Gambar 4.1-8 Tampilan Proses Instalasi Dependencies file requirements.txt**

Dapat dilihat pada Gambar 4.1-8, tidak terdapat pesan error selama proses instalasi. Ini mengindikasikan bahwa proses instalasi sudah berjalan dengan baik.

#### 4.1.4 Instalasi labellmg

Pada proyek *Face-mask Detection* ini, digunakan *software* labelImg untuk melakukan labelling dataset. LabelImg adalah bersifat *open-source* untuk melabeli gambar secara grafis. *Software* ini ditulis dengan Python dan menggunakan QT untuk antarmuka grafisnya. Cara ini adalah cara mudah dan gratis untuk memberi label pada gambar untuk

keperluan proyek pendekripsi objek. Berikut adalah link untuk melakukan *clone* repositori labelImg : <https://github.com/tzutalin/labelImg> . Setelah melakukan *download* repositori pada link tersebut, diekstrak folder hasil *download*. Pada kasus ini, folder yolov5 disimpan di alamat **/Documents/face\_mask** . Berikut adalah tampilan folder labelImg yang telah di-*download*.

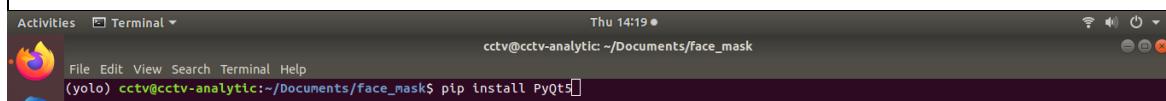


**Gambar 4.1-9 Tampilan Folder labelImg**

#### 4.1.5 Instalasi *Dependencies* untuk labelImg

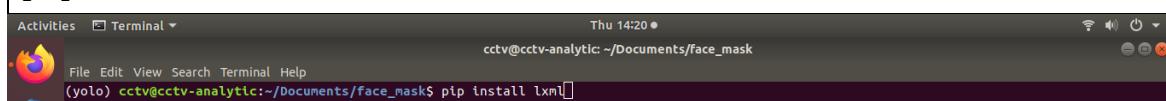
Berikut adalah *dependency* yang harus di-*install* untuk labelImg.

```
pip install PyQt5
```



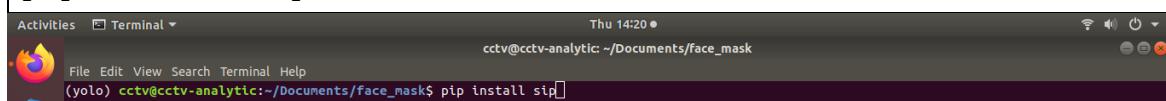
**Gambar 4.1-10 Instalasi PyQt5**

```
pip install lxml
```



**Gambar 4.1-11 Instalasi lxml**

```
pip install sip
```



**Gambar 4.1-12 Instalasi sip**

Apabila tidak terdapat pesan *error* maka proses instalasi berjalan dengan baik. Berikut adalah tampilan proses instalasi *dependencies* untuk labelImg.

```

Activities Terminal Thu 14:17 *
cctv@cctv-analytic: ~/Documents/face_mask
File Edit View Search Terminal Help
cctv@cctv-analytic:~$ cd Documents
cctv@cctv-analytic:~/Documents$ cd face_mask
cctv@cctv-analytic:~/Documents/face_mask$ workon yolo
(yolo) cctv@cctv-analytic:~/Documents/face_mask$ pip install PyQt5
Requirement already satisfied: PyQt5 in /home/cctv/.virtualenvs/yolo/lib/python3.6/site-packages (5.15.6)
Requirement already satisfied: PyQt5-sip<13,>=12.8 in /home/cctv/.virtualenvs/yolo/lib/python3.6/site-packages (from PyQt5) (12.9.0)
Requirement already satisfied: PyQt5-Qt5>=5.15.2 in /home/cctv/.virtualenvs/yolo/lib/python3.6/site-packages (from PyQt5) (5.15.2)
WARNING: You are using pip version 20.0.2; however, version 21.3.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
(yolo) cctv@cctv-analytic:~/Documents/face_mask$ pip install lxml
Requirement already satisfied: lxml in /home/cctv/.virtualenvs/yolo/lib/python3.6/site-packages (4.7.1)
WARNING: You are using pip version 20.0.2; however, version 21.3.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
(yolo) cctv@cctv-analytic:~/Documents/face_mask$ pip install sip
Requirement already satisfied: sip in /home/cctv/.virtualenvs/yolo/lib/python3.6/site-packages (6.5.0)
Requirement already satisfied: packaging in /home/cctv/.virtualenvs/yolo/lib/python3.6/site-packages (from sip) (21.3)
Requirement already satisfied: toml in /home/cctv/.virtualenvs/yolo/lib/python3.6/site-packages (from sip) (0.10.2)
Requirement already satisfied: setuptools in /home/cctv/.virtualenvs/yolo/lib/python3.6/site-packages (from sip) (45.2.0)
Requirement already satisfied: pyParsing!=3.0.5,>=2.0.2 in /home/cctv/.virtualenvs/yolo/lib/python3.6/site-packages (from packaging->sip) (3.0.3)
WARNING: You are using pip version 20.0.2; however, version 21.3.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.

```

Gambar 4.1-13 Tampilan Proses Instalasi *Dependencies* untuk labelImg

## 4.2 Persiapan Dataset

Dataset pada proyek ini diperoleh melalui 2 cara yaitu dengan melakukan pemrograman sehingga diperoleh *capture* dari *webcam*, dan melalui platform-platform gratis yang ada di internet.

### 4.2.1 Capture dengan *webcam*

Cara ini merupakan cara yang dapat digunakan untuk memperoleh dataset. Dilakukan dengan memprogram sehingga *frame* gambar hasil tangkapan *webcam* dapat diambil kemudian disimpan pada rentang periode tertentu. Terdapat 2 jenis dataset yang diambil berdasarkan kelasnya yaitu Mask (wajah dengan masker) dan NoMask (wajah tanpa masker). Untuk tiap kelasnya diberikan variasi juga terhadap sisi wajah. Berikut adalah kode program untuk melakukan *capture* dan menyimpan hasil *capture* tersebut ke suatu folder.

```

import torch
from matplotlib import pyplot as plt
import numpy as np
import cv2
import uuid
import os
import time

IMAGES_PATH = os.path.join('data', 'collected_images')
labels = ['Masked', 'NoMasked']
position = ['Facing Front!', 'Facing Right!', 'Facing Left!', 'Facing Down!']
number_imgs = 2

cap = cv2.VideoCapture(0)
for label in labels:
    for pos in position:
        if label == 'Masked':
            print('Please Wear Your Face-Mask and {}'.format(pos))
        if label == 'NoMasked':
            print('Please Take-off Your Face-Mask and {}'.format(pos))
        time.sleep(3)
        print('Collecting image for {} and {}'.format(label, pos))
        time.sleep(1)
        for img_num in range(number_imgs):
            print('Collecting image number {}'.format(img_num))

            ret, frame = cap.read()

```

```

        imgname = os.path.join(IMAGES_PATH,
label+'.'+pos+'. '+str(uuid.uuid1())+'.jpg')
cv2.imwrite(imgname,frame)
cv2.imshow('Image Collection', frame)
time.sleep(1)
if cv2.waitKey(10) & 0xFF == ord('q'):
    break
print('\n')
cap.release()

```

Pada kode diatas dapat dilihat pengambilan *capture* gambar mengacu pada label, posisi, dan jumlah gambar yang diinginkan. Terdapat 2 label yaitu Masked (wajah menggunakan masker) dan NoMasked (wajah tanpa masker). Kemudian setiap label tersebut terdapat variasi sisi wajah yaitu depan, kanan, kiri, dan bawah. Setiap variasi sisi dari label tersebut akan dilakukan *capture* sebanyak 2 kali. Sehingga total akan dilakukan *capture* sebanyak 16 gambar. Tiap gambar tersebut akan disimpan sesuai dengan alamat pada variable `IMAGES_PATH` dengan diberikan kode unik tiap gambar sebagai nama gambar dan dalam ekstensi `.jpg`. Input kamera yang digunakan adalah *webcam* ditandai dengan kode ‘0’ pada parameter `VideoCapture`. Berikut adalah tampilan terminal dan hasil pada folder Ketika program dijalankan.

```

Please Wear Your Face-Mask and Facing Front!
Collecting image for Masked and Facing Front!
Collecting image number 0
Collecting image number 1

Please Wear Your Face-Mask and Facing Right!
Collecting image for Masked and Facing Right!
Collecting image number 0
Collecting image number 1

Please Wear Your Face-Mask and Facing Left!
Collecting image for Masked and Facing Left!
Collecting image number 0
Collecting image number 1

Please Wear Your Face-Mask and Facing Down!
Collecting image for Masked and Facing Down!
Collecting image number 0
Collecting image number 1

Please Take-off Your Face-Mask and Facing Front!
Collecting image for NoMasked and Facing Front!
Collecting image number 0
Collecting image number 1

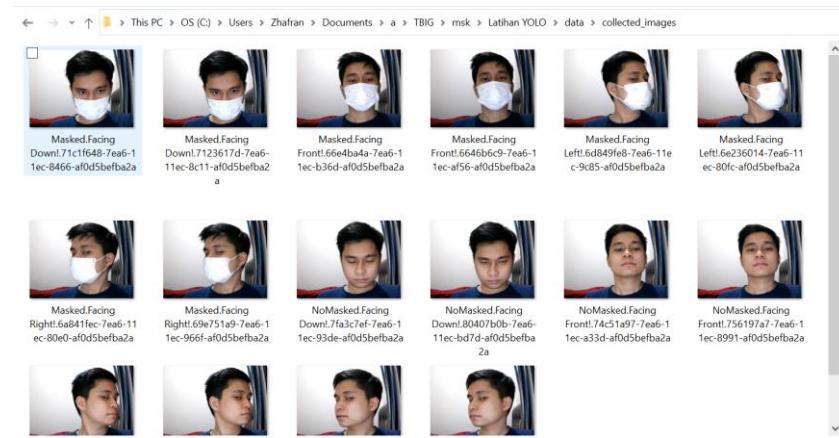
Please Take-off Your Face-Mask and Facing Right!
Collecting image for NoMasked and Facing Right!
Collecting image number 0
Collecting image number 1

Please Take-off Your Face-Mask and Facing Left!
Collecting image for NoMasked and Facing Left!
Collecting image number 0
Collecting image number 1

Please Take-off Your Face-Mask and Facing Down!
Collecting image for NoMasked and Facing Down!
Collecting image number 0
Collecting image number 1

```

**Gambar 4.2-1 Tampilan Terminal Ketika Collecting Images**

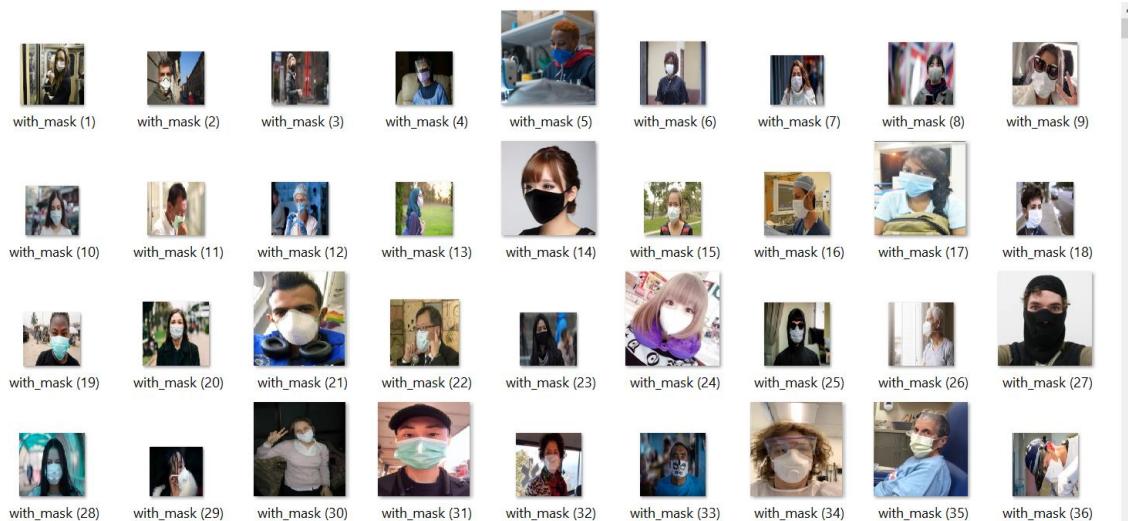


**Gambar 4.2-2 Hasil Data Capture Tersimpan**

Dapat dilihat pada gambar 4.2-1 dan 4.2-2 ketika program dijalankan. Terlihat pada folder sudah berhasil disimpan sejumlah 16 gambar. Penulis menjadikan foto-foto hasil *capture webcam* ini sebagai dataset ketika awal mengembangkan program untuk melakukan pengetesan apakah proses *training* dan *running* program sudah berjalan dengan baik atau belum.

#### 4.2.2 Mengumpulkan Dataset Melalui Internet

Pada metode pengumpulan dataset melalui *capture* menurut penulis cukup memakan waktu dan dipenuhi oleh keterbatasan-keterbatasan jenis wajah yang dapat diperoleh. Diperlukan variasi-variasi bentuk wajah, jenis masker, warna masker, sisi wajah, dan kombinasi-kombinasi lainnya sehingga metode tersebut kurang efektif apabila digunakan. Sehingga penulis memutuskan untuk mencari sebanyak-banyaknya dataset melalui internet. Penulis melakukan penambahan dataset secara bertahap. Setiap dilakukan beberapa penambahan dataset, penulis langsung melakukan pelabelan dataset kemudian melakukan *training*. Selanjutnya hasil *training* tersebut penulis uji kekurangan-kekurangan apa saja yang dapat ditingkatkan melalui penambahan dataset. Berikut adalah cuplikan dataset yang diambil dari internet.



**Gambar 4.2-3 Cuplikan Dataset**

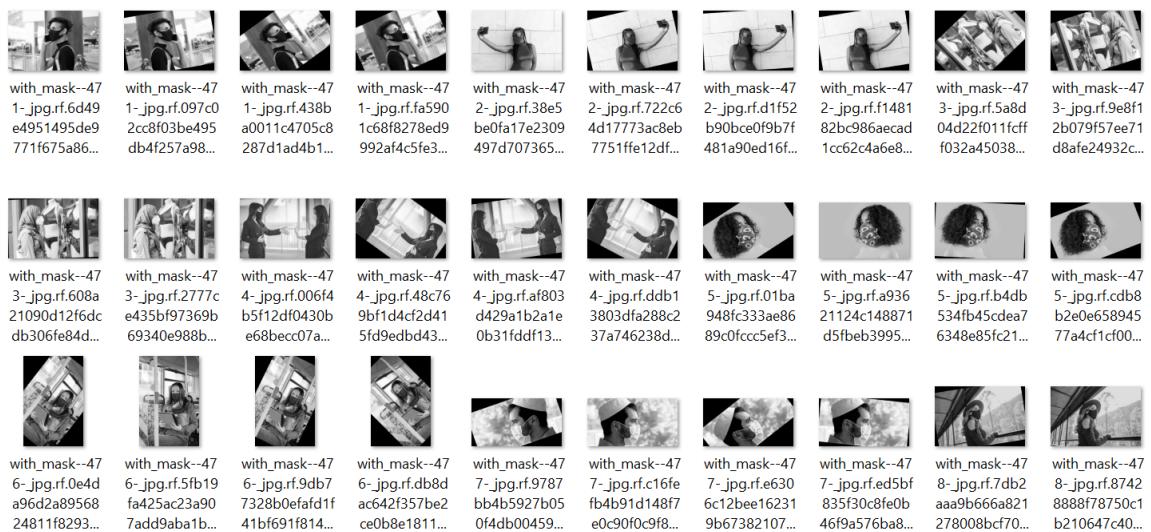
Berikut adalah table rekapan jumlah dataset yang penulis peroleh melalui internet.

**Table 4.2-1 Jumlah Dataset Mentah**

Mask (Wajah Dengan masker)	1221
NoMask (Wajah Tanpa Masker)	2048
Total	3269

#### 4.2.3 Memperbaiki Dataset

Dataset yang dikumpulkan sebelumnya merupakan dataset mentah yang masih belum dilakukan perbaikan agar menghasilkan pemodelan *machine learning* yang semakin akurat. Pada proses ini melakukan augmentasi dengan cara rotasi dan flip. Selain itu penulis juga mengubah warna dataset menjadi *grayscale* dikarenakan pada proyek ini tidak diperlukan identifikasi warna. Sehingga program yang dijalankan dapat berfungsi dengan baik untuk seluruh warna kulit wajah dan warna masker. Berikut adalah cuplikan dataset hasil perbaikan yang penulis lakukan.



**Gambar 4.2-4 Tampilan Dataset Akhir**

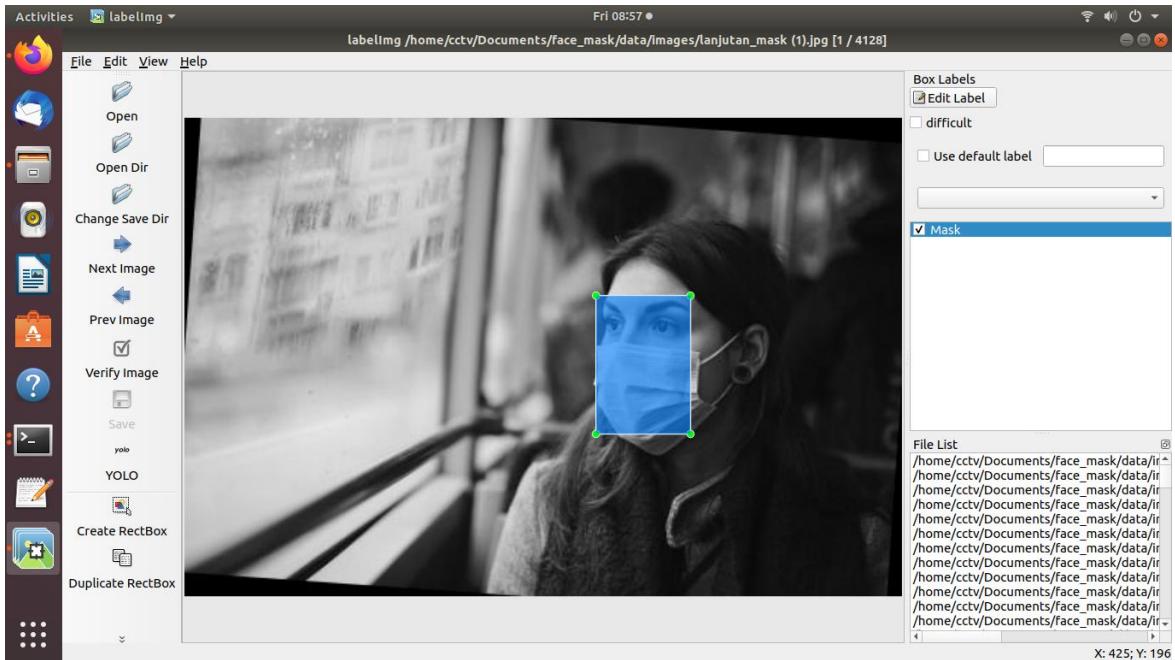
Berikut adalah table rekapan jumlah dataset keseluruhan program.

**Table 4.2-2 Jumlah Dataset Kesuruan**

Mask (Wajah Dengan masker)	1531
NoMask (Wajah Tanpa Masker)	2597
Total	4128

### 4.3 Labelling Dataset

Proyek ini merupakan proyek *machine learning* tipe *supervised learning* sehingga perlu dilakukannya labelling (membuatkan anotasi) dari dataset yang telah dipersiapkan sebelumnya. Proses pelabelan data ini dilakukan pada semua data yang ada pada dataset. Proses labelling menggunakan aplikasi Bernama labelImg. Berikut adalah tampilan proses melakukan labelling dataset.



Gambar 4.3-1 Labelling Dataset dengan labelImg

Label mengidentifikasi vektor data yang sesuai untuk ditarik sebagai acuan pelatihan model, di mana model, kemudian, belajar membuat prediksi terbaik. Dikarenakan algoritma pemrograman ML menggunakan YOLO, maka output dari hasil labelling data ini adalah berupa file anotasi dengan ekstensi .txt.

Berikut cuplikan isi dari file anotasi hasil labelling data yang dibuat.

```
with_mask--1.jpg.rf.bff0b7f683681fc1e6d3e4500616a29e.txt
1 0 0.63375 0.3298076923076923 0.13 0.2673076923076923
2 |
```

Gambar 4.3-2 Cuplikan Isi File Anotasi Data Wajah Menggunakan Masker

```
without_mask--217-.jpg.rf.637ad748550d9f20db44ce704bad3a09.txt
1 2 0.540625 0.5572429906542056 0.53125 0.39953271028037385
2 |
```

Gambar 4.3-3 Cuplikan Isi File Anotasi Data Wajah Tanpa Masker

```
classes.txt
1 Mask
2 MaskIncorrect
3 NoMask|
```

Gambar 4.3-4 Isi File classes.txt

Gambar diatas merupakan gambar file anotasi dengan format YOLO. Akan secara otomatis terbuat file-file anotasi dari tiap data gambar (Gambar 4.3-2, Gambar 4.3-3) dan 1 file yang berisikan rekapan dari kelas-kelas yang ada pada proses pelabelan keseluruhan

(Gambar 4.3-4). Format YOLO memiliki spesifikasi yaitu [class] [x\_center] [y\_center] [width] [height]. Perhitungan kelas dimulai dari indeks 0 sehingga indeks 0 merupakan representasi dari kelas Mask, indeks 1 merupakan representasi dari kelas MaskIncorrect, dan indeks 2 merupakan representasi dari kelas NoMask. Namun pada kasus ini penulis tidak menyiapkan dataset wajah pada kelas MaskIncorrect (wajah dengan penggunaan masker yang tidak tepat) dikarenakan dataset yang kurang banyak. Apabila dipaksa untuk digunakan akan memperlambat proses *training* dan akurasi yang menurun dikarenakan data yang kurang seimbang jumlahnya (*Data Imbalance*).

## 4.4 Training Machine Learning Model

### 4.4.1 Membuat Pipeline

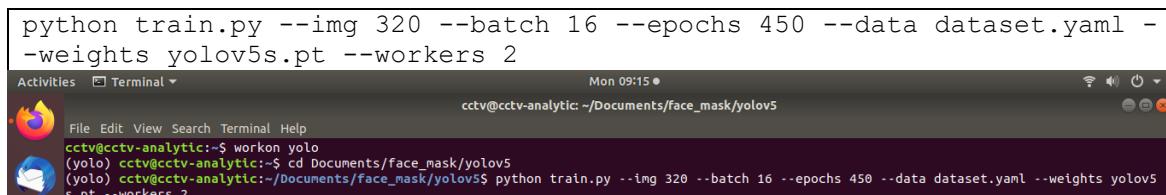
Untuk melakukan *training*, perlu dibuatkan kode program terlebih dahulu sebagai jalur input dataset. Kode program yang penulis buat disimpan dalam nama file dataset.yaml. File dataset.yaml digunakan sebagai *pipeline training* yang disimpan pada alamat **Documents/face\_mask/yolov5**. Isi dari file ini kode untuk mengakses data-data gambar dan pelabelan, serta *list* dari kelas yang ada pada program. *List* kelas yang ada dapat diambil dari output file hasil pelabelan (Gambar 4.3-4). Berikut isi file dataset.yaml yang telah disesuaikan dengan program *face-mask detection*.

```
# Train/val/test sets as 1) dir: path/to/imgs, 2) file:  
path/to/imgs.txt, or 3) list: [path/to/imgs1, path/to/imgs2, ...]  
path: ../data  
train: images  
val: images  
  
# Classes  
nc: 3 # number of classes  
names: ['Mask', 'MaskIncorrect', 'NoMask'] # class names
```

### 4.4.2 Training

Setelah direktori kerja sudah sesuai dan file dataset.yaml sudah disesuaikan, selanjutnya adalah proses *training* model *machine learning*. Proses *training* ini menggunakan dataset wajah dan pelabelan yang sudah dilakukan, dan file *pre-trained* model YOLO. Pada proyek ini, penulis menggunakan yolov5s sebagai *pre-trained* model program. Berikut adalah kode perintah untuk melakukan *training* pemodelan *machine learning* program.

```
python train.py --img 320 --batch 16 --epochs 450 --data dataset.yaml -  
-weights yolov5s.pt --workers 2
```



Gambar 4.4-1 Perintah Melakukan Training Machine Learning Program Face-mask Detection

Dapat dilihat pada Gambar 4.4-1 diatas yaitu kode perintah untuk melakukan *training*. Penulis menggunakan *image size* 320, *batch size* 16, dan *epochs* 450. Parameter ini dapat diubah sesuai dengan nilai yang diinginkan agar diperoleh hasil pemodelan yang terbaik. Berikut adalah tampilan ketika proses *training* sedang berlangsung.

```

Activities Terminal Mon 09:32 •
cctv@cctv-analytic: ~/Documents/face_mask/yolov5

File Edit View Search Terminal Help

Epoch gpu_mem box obj cls labels img_size
3/449 1.53G 0.05447 0.01292 0.01313 27 320: 100% | 169/169 [01:16<00:00, 2.20it/s]
Class Images Labels P R mAP@.5 mAP@.5:.95: 100% | 85/85 [00:25<00:00, 3.29it/s]
all 2703 2862 0.674 0.775 0.792 0.36

Epoch gpu_mem box obj cls labels img_size
4/449 1.53G 0.04983 0.0124 0.0109 24 320: 100% | 169/169 [01:16<00:00, 2.20it/s]
Class Images Labels P R mAP@.5 mAP@.5:.95: 100% | 85/85 [00:25<00:00, 3.29it/s]
all 2703 2862 0.837 0.825 0.892 0.382

Epoch gpu_mem box obj cls labels img_size
5/449 1.53G 0.04863 0.0123 0.01111 22 320: 100% | 169/169 [01:16<00:00, 2.21it/s]
Class Images Labels P R mAP@.5 mAP@.5:.95: 100% | 85/85 [00:25<00:00, 3.28it/s]
all 2703 2862 0.887 0.866 0.936 0.56

Epoch gpu_mem box obj cls labels img_size
6/449 1.53G 0.04327 0.01226 0.0105 25 320: 100% | 169/169 [01:15<00:00, 2.23it/s]
Class Images Labels P R mAP@.5 mAP@.5:.95: 100% | 85/85 [00:25<00:00, 3.29it/s]
all 2703 2862 0.92 0.885 0.943 0.567

Epoch gpu_mem box obj cls labels img_size
7/449 1.53G 0.04155 0.01219 0.009889 36 320: 100% | 169/169 [01:14<00:00, 2.27it/s]
Class Images Labels P R mAP@.5 mAP@.5:.95: 100% | 85/85 [00:25<00:00, 3.30it/s]
all 2703 2862 0.921 0.912 0.958 0.545

Epoch gpu_mem box obj cls labels img_size
8/449 1.53G 0.0385 0.01152 0.008802 21 320: 100% | 169/169 [01:17<00:00, 2.18it/s]
Class Images Labels P R mAP@.5 mAP@.5:.95: 100% | 85/85 [00:25<00:00, 3.30it/s]
all 2703 2862 0.854 0.853 0.912 0.554

Epoch gpu_mem box obj cls labels img_size
9/449 1.53G 0.03758 0.01165 0.008524 24 320: 100% | 169/169 [01:14<00:00, 2.28it/s]
Class Images Labels P R mAP@.5 mAP@.5:.95: 100% | 85/85 [00:25<00:00, 3.30it/s]
all 2703 2862 0.944 0.937 0.97 0.538

Epoch gpu_mem box obj cls labels img_size
10/449 1.53G 0.03712 0.01115 0.008293 37 320: 50% | 84/169 [00:45<00:35, 2.41it/s]

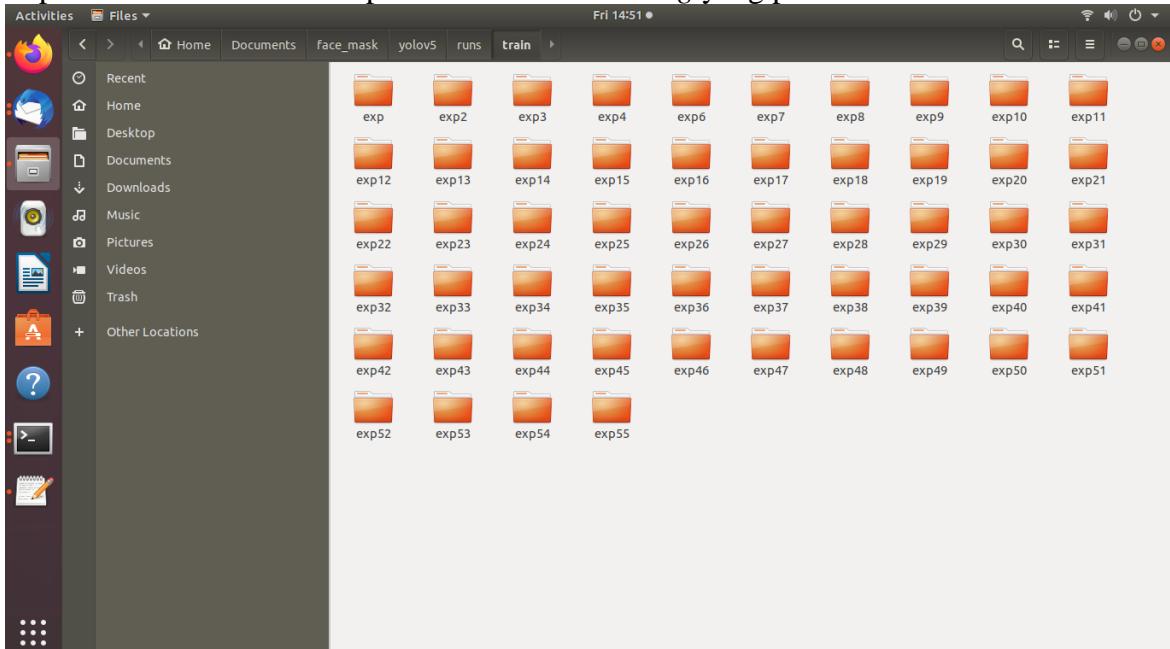
```

**Gambar 4.4-2 Proses Training Program**

Dapat dilihat pada Gambar 4.4-2 yaitu Ketika proses *training* sedang berlangsung. Akan dilakukan proses pelatihan sebanyak epochs yang dituliskan pada kode perintah terminal. Pada kasus ini, dilakukan iterasi sebanyak 450 kali. Semakin banyak iterasi yang diinginkan maka akan semakin lama waktu yang dibutuhkan untuk melakukan *training*.

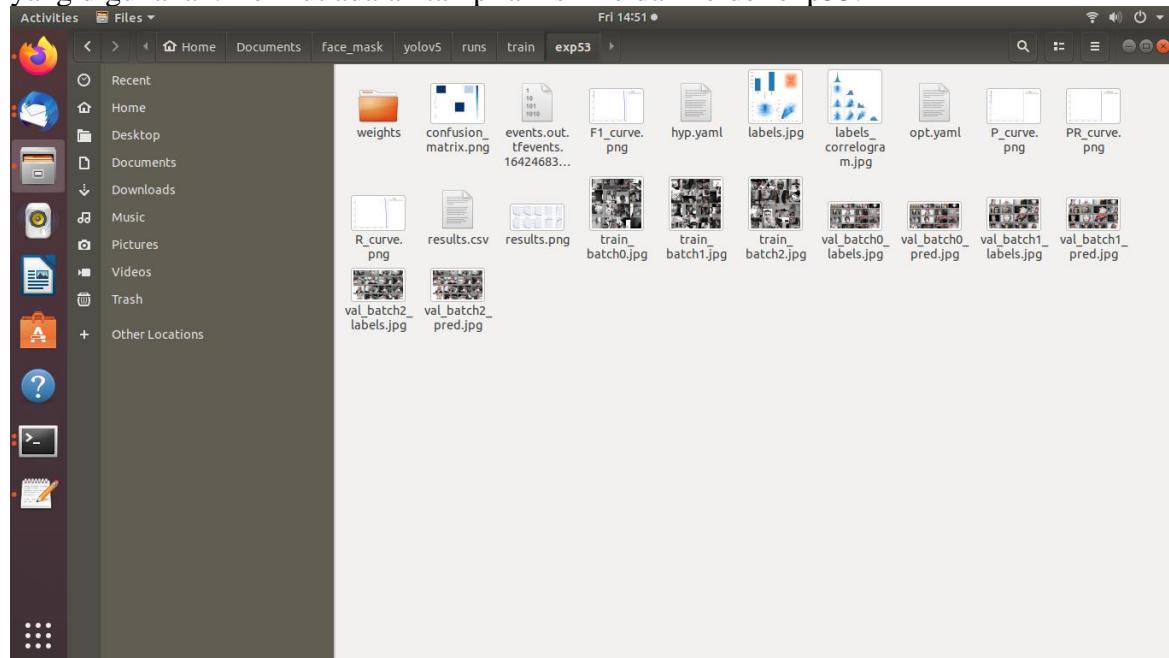
#### 4.4.3 Output Training

Output hasil *training* secara otomatis disimpan kedalam folder dengan alamat **Documents/face\_mask/yolov5/runs/train**. Pada folder ini disimpan folder-folder hasil *training* untuk 1 kali pelatihan. Dengan kata lain setiap pelatihan disimpan pada folder yang terpisah. Berikut adalah tampilan folder hasil *training* yang pernah dilakukan.



**Gambar 4.4-3 Folder Hasil Training**

Dapat dilihat pada gambar 4.4-3 yaitu folder-folder hasil *training* yang pernah dilakukan penulis. Hasil *Training* itu merupakan tahapan-tahapan *training* yang pernah penulis lakukan dengan melakukan perubahan-perubahan terhadap jumlah dataset, jumlah iterasi, pewarnaan dataset, jenis *pre-trained* model, dan variasi-variasi lainnya. Penulis melakukan hal tersebut untuk mencapai hasil terbaik model *machine learning*. Folder hasil *training* akhir penulis yaitu pada folder **exp53** (menggunakan *pre-trained model* yolov5s), **exp54** (menggunakan *pre-trained model* yolov5m), dan **exp55** (menggunakan *pre-trained model* yolov5l). Ketiga folder tersebut berupa file akhir pemodelan yang hanya membedakan jenis *pre-trained model* nya saja. Jumlah dataset dan iterasi *training* menggunakan jumlah yang sama. Variasi ini bertujuan untuk dilakukan pengujian yang paling cocok dengan *hardware* yang digunakan. Berikut adalah tampilan isi file dari folder exp53.



Gambar 4.4-4 Tampilan Isi Folder Hasil *Training*

Dapat dilihat pada Gambar 4.4-4 yaitu isi folder hasil 1 kali *training*. Terdapat beberapa file yang meliputi hasil mAP pemodelan, *confusion matrix*, *loss*, *prediction*, dan yang paling penting adalah file model itu sendiri yang terletak dalam folder weights. File pemodelan ini merupakan file dengan ekstensi .pt yang digunakan program untuk menghasilkan prediksi *machine learning*.

## 4.5 Membuat Database Hasil Deteksi Program

### 4.5.1 Create Database

Hasil pendekripsi program memerlukan suatu tempat penyimpanan (database). Penulis menggunakan MySQL sebagai database server hasil deteksi program. Kode program yang ditulis menggunakan python dengan memanfaatkan *library* mysql konektor. Berikut adalah kode program yang dibuat untuk membuat database.

```
import mysql.connector
# Connect to mysql database
db = mysql.connector.connect(
    host = "localhost",
    user = "root",
)
mycursor = db.cursor()
# Create Table
```

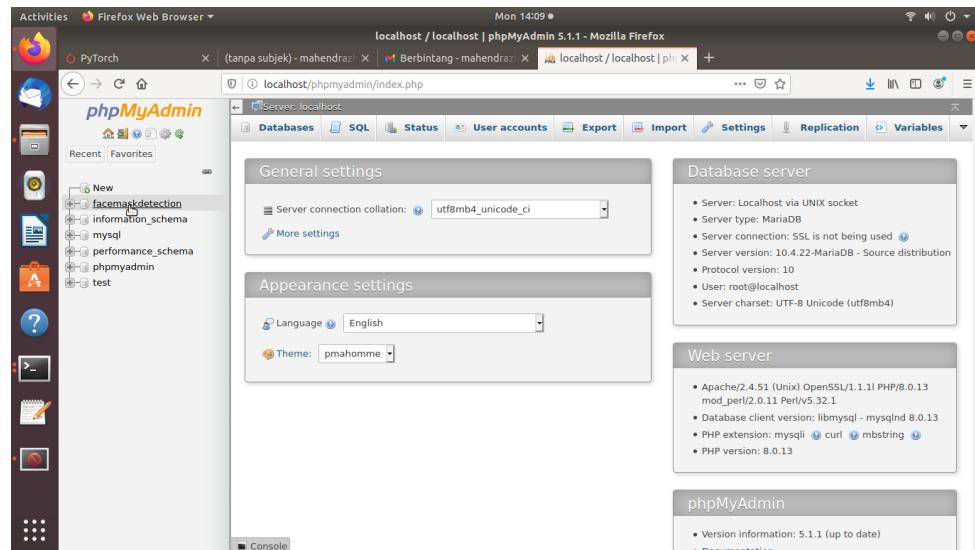
```

mycursor.execute("CREATE DATABASE facemaskdetection")

db.commit()
db.close()

```

Dapat dilihat pada kode program penulis membuat suatu database Bernama facemaskdetection. Setelah menjalankan program tersebut, dilihat pada GUI phpMyAdmin untuk mengecek apakah database tersebut sudah terbuat atau belum. Berikut adalah tampilan *database* yang telah dibuat.



**Gambar 4.5-1 Database Sistem Face-mask Detection**

Terlihat pada gambar 4.5-1 bagian kiri sudah terdapat database Bernama facemaskdetection. Ini mengindikasikan bahwa database sudah berhasil dibuat.

#### 4.5.2 Create Table

Setelah *database* berhasil dibuat, selanjutnya adalah mengisi database tersebut dengan suatu table. Tabel inilah yang akan dijadikan tempat menyimpan secara langsung data-data yang diperoleh dari hasil deteksi program. Berikut adalah kode program yang dibuat.

```

import mysql.connector
# Connect to mysql database
db = mysql.connector.connect(
    host = "localhost",
    user = "root",
    database = "facemaskdetection"
)

mycursor = db.cursor()
# Create Table

mycursor.execute("""CREATE TABLE table_list
(
    no varchar(20),
    `date` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    id int,
    cls varchar(20),
    bbox_left varchar(20),
    bbox_top varchar(20),
    bbox_right varchar(20),
    bbox_bottom varchar(20)
)""")

```

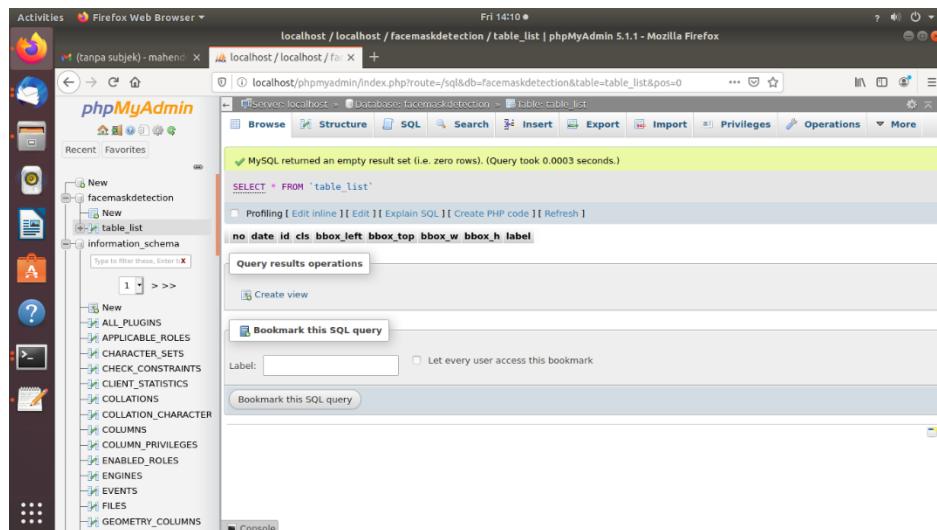
```

bbox_w varchar(20),
bbox_h varchar(20),
label varchar(20)

)
""")
```

db.commit()  
db.close()

Berikut adalah hasil Ketika program diatas dijalankan.



**Gambar 4.5-2 Tabel Program Face-mask Detection**

Dapat terlihat pada Gambar 4.5-2 tabel Bernama table\_list untuk menampung seluruh data hasil deteksi telah berhasil dibuat. Tabel ini berada dibawah facemaskdetection database. Terdapat informasi-informasi mengenai jumlah data (no), timestamp (date), id, nomor kelas (cls), koordinat *bounding box* (bbox\_left, bbox\_top, bbox\_w, b box\_h), dan nama kelas. Pada kondisi ini, data-data pada tabel ini masih kosong. Pada kondisi ini database sudah siap untuk menampung seluruh data yang akan dikirimkan oleh program untuk disimpan.

## 4.6 Running Program

### 4.6.1 Menjalankan Program Pada Foto

Untuk melakukan pengecekan terhadap akurasi program, penulis mempersiapkan program untuk menjalankan model *machine learning* kepada data *dummy* yang sudah penulis persiapkan sebelumnya. Pada spesifikasi diinginkan akurasi program yang menggunakan data *dummy* diatas 95%. Berikut adalah kode program yang dibuat.

```

import torch
from matplotlib import pyplot as plt
import numpy as np
import cv2

img = cv2.imread('tes (56).png')
rgb = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
results = model(rgb)
%matplotlib inline
plt.imshow(np.squeeze(results.render()))
plt.show()
```

Foto yang dipersiapkan untuk diuji masih dalam warna RGB. Karena program yang dirancang ini merupakan program dengan dataset berwarna *grayscale*, sehingga program akan efektif pada frame berwarna *grayscale*. Oleh karena itu pada kode program diubah terlebih dahulu warna foto ke *grayscale*. Kemudian menampilkan foto tersebut setelah melalui proses pendekripsi masker. Berikut adalah cuplikan hasil Ketika program dijalankan.



Gambar 4.6-1 Foto Sebelum (Kiri) dan Sesudah (Kanan) Pemrosesan

Dapat dilihat pada gambar 4.6-1 bagian kiri yaitu gambar mentah sebelum dilakukan deteksi dan bagian kanan yaitu gambar foto setelah dilakukan deteksi. Pada gambar tersebut terlihat bahwa program dapat berjalan dengan baik. Program akan diujikan pada beberapa sampel gambar untuk memperoleh parameter akurasi dengan *data dummy* yang sesuai dengan spesifikasi produk.

#### 4.6.2 Real-time Detection

Program yang dikembangkan selanjutnya diujikan untuk berjalan mendekripsi secara *Real-time* (*Real-time Detection*). Penulis menggunakan *webcam* sebagai input modul kamera untuk menguji tahap awal program secara *real-time*. Implementasi akhir program akan tetap menggunakan IP Camera sebagai modul kamera system namun penulis ingin mencoba terlebih dahulu melalui cara yang lebih mudah yaitu menggunakan *webcam*. Berikut adalah kode program yang dibuat.

```
import torch
from matplotlib import pyplot as plt
import numpy as np
import cv2

model = torch.hub.load('ultralytics/yolov5', 'custom', path =
'yolov5/runs/train/exp53/weights/last.pt', force_reload=True)
def rescaleFrame(frame):
    #berlaku untuk foto, video, dan live video
    scale = 1.5
    width = int(frame.shape[1] *scale)
    height = int(frame.shape[0] *scale)
    dimensions = (width,height)

    return cv2.resize(frame, dimensions, interpolation =
cv2.INTER_AREA)

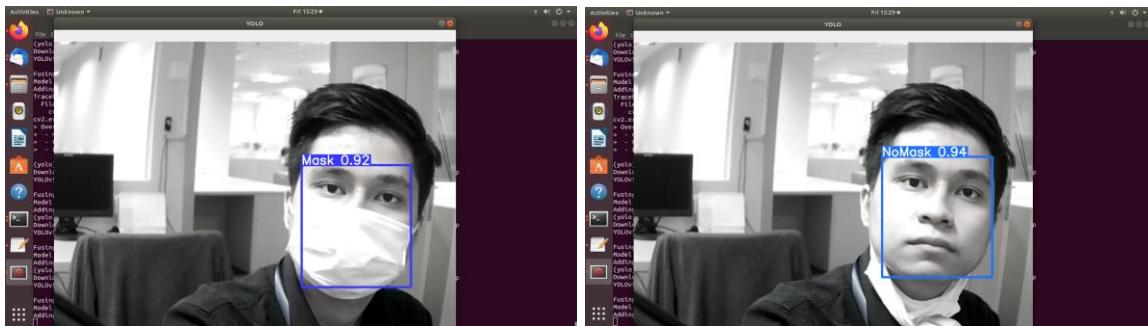
cap = cv2.VideoCapture(0)
while cap.isOpened():
    ret, frame = cap.read()
    frame = rescaleFrame(frame)
```

```

analytics = model(frame)
cv2.imshow('YOLO', np.squeeze(analytics.render()))

if cv2.waitKey(10) & 0xFF == ord('q'):
    break
cap.release()
cv2.destroyAllWindows()

```



Gambar 4.6-2 Tampilan *Real-time Detection* Menggunakan *Webcam*

Dapat dilihat pada Gambar 4.6-2 yaitu output program diatas. Terlihat bahwa program menampilkan deteksi secara *real-time* terhadap wajah penulis. Sampai tahap ini program sudah dapat membedakan wajah seseorang yang menggunakan masker dan tidak menggunakan masker secara *real-time*.

#### 4.6.3 *Real-time Detection Multiple Object Tracking*

Pada tahap sebelumnya sudah berhasil dilakukan pendekripsi penggunaan masker dari suatu wajah. Namun sampai tahap tersebut data yang diperoleh masih sangat mentah dan perlu dilakukannya *tracking* terhadap objek yang terdeteksi tersebut. Hal ini dilakukan agar objek yang terdeteksi memiliki suatu identitas sehingga tidak terhitung lebih dari satu kali. Ketika masuk kedalam database. Metode ini disebut dengan *Multiple Object Tracking* (MOT). Terdapat beberapa metode MOT yang dapat digunakan. Penulis menggunakan metode yang disebut dengan DeepSORT. DeepSORT merupakan singkatan dari Deep Learning Simple Object Real-time Tracking. Penulis menggunakan repositori yang dibuat oleh Mikel Brostrom melalui link [https://github.com/mikel-brostrom/Yolov5\\_DeepSort\\_Pytorch](https://github.com/mikel-brostrom/Yolov5_DeepSort_Pytorch). Penulis melakukan *cloning* terhadap folder tersebut dan melakukan modifikasi pada program track.py. Penulis melakukan modifikasi untuk menuliskan data-data yang diperlukan untuk masuk kedalam database yang telah dibuat sebelumnya. Sehingga data yang dituliskan pada database hanya dapat 1 kali saja masuk ke database (tidak ada data dengan id dan label yang sama pada database). Berikut adalah hasil modifikasi kode program track.py yang penulis buat.

```

# limit the number of cpus used by high performance libraries
import os
os.environ["OMP_NUM_THREADS"] = "1"
os.environ["OPENBLAS_NUM_THREADS"] = "1"
os.environ["MKL_NUM_THREADS"] = "1"
os.environ["VECLIB_MAXIMUM_THREADS"] = "1"
os.environ["NUMEXPR_NUM_THREADS"] = "1"

import sys
sys.path.insert(0, './yolov5')

from yolov5.models.experimental import attempt_load
from yolov5.utils.downloads import attempt_download
from yolov5.models.common import DetectMultiBackend
from yolov5.utils.datasets import LoadImages, LoadStreams

```

```

from yolov5.utils.general import LOGGER, check_img_size,
non_max_suppression, scale_coords, check_imshow, xyxy2xywh
from yolov5.utils.torch_utils import select_device, time_sync
from yolov5.utils.plots import Annotator, colors
from deep_sort_pytorch.utils.parser import get_config
from deep_sort_pytorch.deep_sort import DeepSort
import argparse
import os
import platform
import shutil
import time
from pathlib import Path
import cv2
import torch
import torch.backends.cudnn as cudnn
from matplotlib import pyplot as plt
import numpy as np

import mysql.connector
# Connect to mysql database
db = mysql.connector.connect(
    host = "localhost",
    user = "root",
    database = "facemaskdetection"
)
mycursor = db.cursor()
idList = []
counterr =[]
labelList =[]
tempp = np.zeros(10000000)
sementara =[]

def detect(opt):
    out, source, yolo_weights, deep_sort_weights, show_vid, save_vid,
    save_txt, imgsz, evaluate, half = \
        opt.output, opt.source, opt.yolo_weights,
    opt.deep_sort_weights, opt.show_vid, opt.save_vid, \
        opt.save_txt, opt.imgsz, opt.evaluate, opt.half
    webcam = source == '0' or source.startswith(
        'rtsp') or source.startswith('http') or
    source.endswith('.txt')
    maksval = 0
    d = 0
    no = 1
    e = 0
    f =0
    # initialize deepsort
    cfg = get_config()
    cfg.merge_from_file(opt.config_deepsort)
    attempt_download(deep_sort_weights, repo='mikel-
brostrom/Yolov5_DeepSort_Pytorch')
    deepsort = DeepSort(cfg.DEEPSORT.REID_CKPT,
                        max_dist=cfg.DEEPSORT.MAX_DIST,
    min_confidence=cfg.DEEPSORT.MIN_CONFIDENCE,
    max_iou_distance=cfg.DEEPSORT.MAX_IOU_DISTANCE,
    max_age=cfg.DEEPSORT.MAX_AGE,
    n_init=cfg.DEEPSORT.N_INIT, nn_budget=cfg.DEEPSORT.NN_BUDGET,
    use_cuda=True)

```

```

# Initialize
device = select_device(opt.device)
half &= device.type != 'cpu' # half precision only supported on CUDA

# The MOT16 evaluation runs multiple inference streams in parallel, each one writing to
# its own .txt file. Hence, in that case, the output folder is not restored
if not evaluate:
    if os.path.exists(out):
        pass
        shutil.rmtree(out) # delete output folder
    os.makedirs(out) # make new output folder

# Load model
device = select_device(device)
model = DetectMultiBackend(opt.yolo_weights, device=device,
dnn=opt.dnn)
stride, names, pt, jit, onnx = model.stride, model.names,
model.pt, model.jit, model.onnx
imgsz = check_img_size(imgsz, s=stride) # check image size

# Half
half &= pt and device.type != 'cpu' # half precision only supported by PyTorch on CUDA
if pt:
    model.model.half() if half else model.model.float()

# Set Dataloader
vid_path, vid_writer = None, None
# Check if environment supports image displays
if show_vid:
    show_vid = check_imshow()

# Dataloader
if webcam:
    view_img = check_imshow()
    cudnn.benchmark = True # set True to speed up constant image size inference
    dataset = LoadStreams(source, img_size=imgsz, stride=stride,
auto=pt and not jit)
    bs = len(dataset) # batch_size
else:
    dataset = LoadImages(source, img_size=imgsz, stride=stride,
auto=pt and not jit)
    bs = 1 # batch_size
vid_path, vid_writer = [None] * bs, [None] * bs

# Get names and colors
names = model.module.names if hasattr(model, 'module') else
model.names

save_path = str(Path(out))
# extract what is in between the last '/' and last '.'
txt_file_name = source.split('/')[-1].split('.')[0]
txt_path = str(Path(out)) + '/' + txt_file_name + '.txt'

if pt and device.type != 'cpu':

```

```

        model(torch.zeros(1, 3,
*imgsz).to(device).type_as(next(model.model.parameters())))) # warmup
        dt, seen = [0.0, 0.0, 0.0], 0
        for frame_idx, (path, img, im0s, vid_cap, s) in
enumerate(dataset):
            t1 = time_sync()
            img = torch.from_numpy(img).to(device)
            img = img.half() if half else img.float() # uint8 to fp16/32
            img /= 255.0 # 0 - 255 to 0.0 - 1.0
            if img.ndim == 3:
                img = img.unsqueeze(0)
            t2 = time_sync()
            dt[0] += t2 - t1

            # Inference
            visualize = increment_path(save_dir / Path(path).stem,
mkdir=True) if opt.visualize else False
            pred = model(img, augment=opt.augment, visualize=visualize)
            t3 = time_sync()
            dt[1] += t3 - t2

            # Apply NMS
            pred = non_max_suppression(pred, opt.conf_thres,
opt.iou_thres, opt.classes, opt.agnostic_nms, max_det=opt.max_det)
            dt[2] += time_sync() - t3

            # Process detections
            for i, det in enumerate(pred): # detections per image
                seen += 1
                if webcam: # batch_size >= 1
                    p, im0, frame = path[i], im0s[i].copy(), dataset.count
                    s += f'{i}: '
                else:
                    p, im0, frame = path, im0s.copy(), getattr(dataset,
'frame', 0)
                    #im0 = cv2.cvtColor(im0, cv2.COLOR_BGR2GRAY)
                    s += '%gx%g ' % img.shape[2:] # print string
                    save_path = str(Path(out) / Path(p).name)

                    annotator = Annotator(im0, line_width=2, pil=not ascii)

                    if det is not None and len(det):
                        # Rescale boxes from img_size to im0 size
                        det[:, :4] = scale_coords(
                            img.shape[2:], det[:, :4], im0.shape).round()

                        # Print results
                        for c in det[:, -1].unique():
                            n = (det[:, -1] == c).sum() # detections per
class
                            s += f"{n} {names[int(c)]}{'' * (n > 1)}, " # add to string

                        xywhs = xyxy2xywh(det[:, 0:4])
                        confs = det[:, 4]
                        clss = det[:, 5]

                        # pass detections to deepsort
                        outputs = deepsort.update(xywhs.cpu(), confs.cpu(),
clss.cpu(), im0)

```

```

        # draw boxes for visualization
        if len(outputs) > 0:
            for j, (output, conf) in enumerate(zip(outputs,
confss)):

                bboxes = output[0:4]
                id = output[4]
                cls = output[5]

                c = int(cls) # integer class

                if save_txt:
                    # to MOT format
                    bbox_left = output[0]
                    bbox_top = output[1]
                    bbox_w = output[2] - output[0]
                    bbox_h = output[3] - output[1]
                    # Write MOT compliant results to file
                    maksval = maksval + 0
                    with open('count_result.txt', 'r+') as f:
                        myDataList = f.readlines()
                        a =0

                        if len(myDataList) >0 and d == 0 :
                            d = 1
                            for i in myDataList :
                                entry = i.split(' ')
                                if entry[1] not in idList :
                                    idList.append(entry[1])
                                    if (maksval <=
int(entry[1])) :
                                        maksval =
int(entry[1])
                                        id = id +maksval
                                        no = no + len(myDataList)
                                else :
                                    id = id +maksval

                            d = 1
                            for line in myDataList:
                                entry = line.split(' ')
                                if entry[1] not in idList :
                                    idList.append(entry[1])
                                    temp[ int(id) ] =
temp[int(id)] + c

                                if temp[int(id)] != 0 and
temp[int(id)] != 4 and temp[int(id)] > 0 and temp[int(id)] <=3:
                                    if (temp[int(id)] == 2 and c
==0):
                                        temp[int(id)] =
temp[int(id)] - c

                                else:
                                    a = 1
                            else :

```

```

temp[int(id)] =
temp[int(id)] - c
a = 0

if ((str(id) not in idList) or
a==1) :
    f.write((%g * 7) % (no,
id, cls, bbox_left, bbox_top, bbox_w, bbox_h)) # label format
    f.write(str(names[c])+'\n')
    mycursor.execute("INSERT INTO
table_list(no,id,cls,bbox_left,bbox_top,bbox_w,bbox_h,label)
VALUES ('"+str(no)+"','"+str(id)+"','"+str(cls)+"','"+str(bbox_left)+"',
'+str(bbox_top)+"','"+str(bbox_w)+"','"+str(bbox_h)+"','"+str(names[c])+"');")
    db.commit()

a = 0
no = no +1
label = f'{id} {names[c]} {conf:.2f}'
annotator.box_label(bboxes, label,
color=colors(c, True))
else:
    deepsort.increment_ages()

# Print time (inference-only)
LOGGER.info(f'{s}Done. ({t3 - t2:.3f}s)')

# Stream results
im0 = annotator.result()

cv2.imshow(p, im0)
if cv2.waitKey(1) == ord('q'): # q to quit
    raise StopIteration

# Save results (image with detections)
if save_vid:
    if vid_path != save_path: # new video
        vid_path = save_path
    if isinstance(vid_writer, cv2.VideoWriter):
        vid_writer.release() # release previous video
writer
    if vid_cap: # video
        fps = vid_cap.get(cv2.CAP_PROP_FPS)
        w = int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
        h =
int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
    else: # stream
        fps, w, h = 60, im0.shape[1], im0.shape[0]
        save_path += '.mp4'

    vid_writer = cv2.VideoWriter(save_path,
cv2.VideoWriter_fourcc(*'mp4v'), fps, (w, h))
    vid_writer.write(im0)

# Print results
t = tuple(x / seen * 1E3 for x in dt) # speeds per image
LOGGER.info(f'Speed: %.1fms pre-process, %.1fms inference, %.1fms
NMS per image at shape {(1, 3, *imgsz)}' % t)
if save_txt or save_vid:

```

```

print('Results saved to %s' % os.getcwd() + os.sep + out)
if platform == 'darwin': # MacOS
    os.system('open ' + save_path)

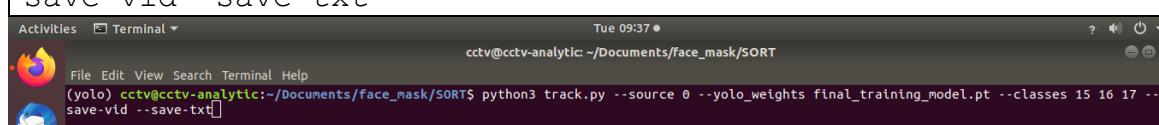
if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--yolo_weights', nargs='+', type=str,
default='yolov5l.pt', help='model.pt path(s)')
    parser.add_argument('--deep_sort_weights', type=str,
default='deep_sort_pytorch/deep_sort/deep/checkpoint/ckpt.t7',
help='ckpt.t7 path')
    # file/folder, 0 for webcam
    parser.add_argument('--source', type=str, default='0',
help='source')
    parser.add_argument('--output', type=str,
default='inference/output', help='output folder') # output folder
    parser.add_argument('--imgsz', '--img', '--img-size', nargs='+',
type=int, default=[640], help='inference size h,w')
    parser.add_argument('--conf-thres', type=float, default=0.4,
help='object confidence threshold')
    parser.add_argument('--iou-thres', type=float, default=0.5,
help='IOU threshold for NMS')
    parser.add_argument('--fourcc', type=str, default='mp4v',
help='output video codec (verify ffmpeg support)')
    parser.add_argument('--device', default='', help='cuda device,
i.e. 0 or 0,1,2,3 or cpu')
    parser.add_argument('--show-vid', action='store_true',
help='display tracking video results')
    parser.add_argument('--save-vid', action='store_true', help='save
video tracking results')
    parser.add_argument('--save-txt', action='store_true', help='save
MOT compliant results to *.txt')
    # class 0 is person, 1 is bycicle, 2 is car... 79 is oven
    parser.add_argument('--classes', nargs='+', type=int, help='filter
by class: --class 0, or --class 16 17')
    parser.add_argument('--agnostic-nms', action='store_true',
help='class-agnostic NMS')
    parser.add_argument('--augment', action='store_true',
help='augmented inference')
    parser.add_argument('--evaluate', action='store_true',
help='augmented inference')
    parser.add_argument("--config_deepsort", type=str,
default="deep_sort_pytorch/configs/deep_sort.yaml")
    parser.add_argument("--half", action="store_true", help="use FP16
half-precision inference")
    parser.add_argument('--visualize', action='store_true',
help='visualize features')
    parser.add_argument('--max-det', type=int, default=1000,
help='maximum detection per image')
    parser.add_argument('--dnn', action='store_true', help='use OpenCV
DNN for ONNX inference')
    opt = parser.parse_args()
    opt.imgsz *= 2 if len(opt.imgsz) == 1 else 1 # expand

    with torch.no_grad():
        detect(opt)

```

Penulis mulai mengimplementasi program pada input IP kamera. Digunakan juga pemodelan *machine learning* yang telah dilatih sebelumnya. Sehingga program diatas dapat dijalankan dengan menuliskan perintah pada terminal sebagai berikut.

```
Python3 track.py --source  
rtsp://admin:admin123@192.168.1.64:554/streaming/channels/1  
02 --yolo_weights final_training_model.pt --classes 0 1 2 --  
save-vid --save-txt
```



**Gambar 4.6-3 Perintah Running Program dan Menyimpan Hasil Deteksi ke Database**

Dapat dilihat pada kode perintah diatas akan dijalankan file Bernama track.py dengan memanfaatkan komunikasi RTSP dari IP kamera. Perintah diatas juga menginstruksikan untuk menggunakan file pemodelan *machine learning* yang telah dilatih yang penulis beri nama final\_training\_model.pt. Perintah --save-vid berguna untuk menyimpan hasil tangkapan video IP Kamera dan perintah --save-txt berguna untuk menyimpan hasil pendekstrian pada database dan suatu file eksternal .txt. Berikut adalah output-output yang dihasilkan Ketika program dijalankan.



**Gambar 4.6-4 Running Program Face-mask Detection**

Dapat dilihat pada Gambar 4.6-4 tampilan program ketika sedang berjalan pada IP Camera. Pewarnaan dari kamera juga sudah penulis atur agar berwarna *grayscale* dikarenakan dataset yang dipersiapkan berwarna *grayscale*. Seperti yang sudah dijelaskan sebelumnya, *grayscale* dipilih dikarenakan warna bukanlah suatu parameter penting pada proyek ini. Sehingga seluruh warna wajah dan warna masker dapat dideteksi dengan baik. Dapat juga dilihat program sudah memiliki identitas (id) pada setiap objek yang dideteksi. Ini menandakan bahwa proses *tracking* sudah berjalan dengan baik.

no	date	id	cls	bbox_left	bbox_top	bbox_w	bbox_h	label	
1	2022-01-28	14:36:24	1	0	250	295	149	179	Mask
2	2022-01-28	14:36:28	1	2	247	283	148	191	NoMask
3	2022-01-28	14:36:31	3	2	247	295	153	184	NoMask
4	2022-01-28	14:36:33	4	2	251	414	199	65	NoMask
5	2022-01-28	14:36:39	5	0	262	372	180	107	Mask
6	2022-01-28	14:36:43	6	0	277	350	170	129	Mask
7	2022-01-28	14:36:50	7	0	258	369	176	110	Mask
8	2022-01-28	14:36:54	7	2	254	285	161	194	NoMask
9	2022-01-28	14:37:02	10	0	255	400	184	79	Mask
10	2022-01-28	14:37:02	11	2	260	397	189	82	NoMask
11	2022-01-28	14:37:11	12	0	270	370	181	109	Mask
12	2022-01-28	14:37:24	12	2	257	304	162	175	NoMask
13	2022-01-28	14:37:28	13	2	245	425	257	54	NoMask
14	2022-01-28	14:37:37	14	0	270	353	169	126	Mask
15	2022-01-28	14:37:42	15	0	232	385	182	94	Mask
16	2022-01-28	14:37:47	16	2	241	375	191	104	NoMask
17	2022-01-28	14:37:56	17	0	258	385	193	94	Mask
18	2022-01-28	14:38:06	20	0	263	320	240	159	Mask
19	2022-01-28	14:38:21	21	2	261	299	161	180	NoMask
20	2022-01-28	14:38:24	23	2	244	393	186	86	NoMask
21	2022-01-28	14:38:30	24	0	263	405	192	74	NoMask

Gambar 4.6-5 Tampilan Database Program

Dapat dilihat pada gambar 3.2-17 yaitu tampilan database setelah menjalankan program *face-mask detection*. Terdapat informasi-informasi mengenai jumlah data (no), *timestamp* (date), id, nomor kelas (cls), koordinat *bounding box* (bbox\_left, bbox\_top, bbox\_w, bbox\_h), dan nama kelas. Program penyimpanan sudah dirancang sehingga data yang masuk ke database tidak mengalami pengulangan. Terlihat pada gambar tidak terdapat data yang sama untuk id dan label yang sama. Penyimpanan juga sudah dirancang untuk mengantisipasi perangkat yang ter-shutdown yang mengakibatkan program harus di-restart. Ini dilakukan dengan mencari id tertinggi dari data terakhir pada database kemudian melanjutkan *counting* dengan acuan id tertinggi tersebut. Sehingga tetap tidak terdapat pengulangan data untuk id dan label yang sama.

```

Fri 14:42 ●
count_result.txt
-/Documents/face_mask/SORT
Save   Plain Text  Tab Width: 8  Ln 1, Col 1  INS

1 1 0 250 295 149 179 Mask
2 1 2 247 283 148 191 NoMask
3 3 2 247 295 153 184 NoMask
4 4 2 251 414 199 65 NoMask
5 5 0 262 372 180 107 Mask
6 6 0 277 350 170 129 Mask
7 7 0 258 369 176 110 Mask
8 7 2 254 285 161 194 NoMask
9 9 0 255 404 184 79 Mask
10 11 2 260 397 189 82 NoMask
11 12 0 270 370 181 109 Mask
12 12 2 257 304 162 175 NoMask
13 13 2 245 425 257 54 NoMask
14 14 0 270 353 169 126 Mask
15 15 0 232 385 182 94 Mask
16 16 2 241 375 191 104 NoMask
17 17 0 258 385 193 94 Mask
18 18 0 263 320 240 159 Mask
19 21 2 261 299 161 180 NoMask
20 23 2 244 393 186 86 NoMask
21 24 0 263 405 192 74 Mask
22 25 2 482 405 157 74 NoMask
23 26 2 568 276 71 201 NoMask
24 28 2 469 426 170 53 NoMask
25 29 0 432 282 206 197 Mask
26 30 0 426 291 162 188 Mask
27 31 0 462 382 126 97 Mask
28 31 2 553 355 80 124 NoMask
29 36 0 499 439 134 40 Mask
30 37 2 604 385 35 94 NoMask
31 36 2 565 360 74 116 NoMask
32 38 0 593 414 46 65 Mask
33 39 0 547 372 92 107 Mask
34 39 2 558 367 81 112 NoMask
35 40 0 432 321 140 158 Mask
36 47 0 507 306 47 53 Mask

```

Gambar 4.6-6 Output File Eksternal .txt

File count\_result.txt memiliki informasi data yang sama persis dengan data yang tersimpan pada database melainkan output nya merupakan file dengan ekstensi .txt.

## 5 B500 PENGUJIAN

### 5.1 Pengujian Spesifikasi Akurasi Deteksi Objek

Table 5.1-1 Verifikasi Spesifikasi Akurasi Deteksi objek

Hal	1. Akurasi
Rincian	1. $\geq 80\%$ Data Real 2. $\geq 94.4\%$ Data <i>Dummy</i>
Metode Pengukuran	Pengujian dilakukan pada wajah manusia berupa data <i>real</i> (gambar wajah yang diambil di kantor) dan data <i>dummy</i> (gambar wajah dengan masker dan tanpa masker dari internet)
Prosedur Pengujian	<ul style="list-style-type: none"><li>- Pengujian data <i>real</i> dilakukan dengan mengambil frame gambar pada ketinggian CCTV pada 2.3 meter dan jarak objek bervariasi pada rentang 0 sampai 5 meter (sesuai spesifikasi jarak dan ketinggian). Kemudian dicuplik frame selama 20 detik sekali kemudian dilihat hasil pendektsian. Hanya wajah dengan sudut kemiringan wajah <math>\pm 60^\circ</math> saja yang diperhitungkan pada perhitungan akurasi (sesuai spesifikasi <i>angle</i> wajah). Sampel yang diambil lebih dari 100 frame gambar tangkapan CCTV pada berbagai variasi posisi kamera. Dihitung akurasi berdasarkan rumus perhitungan akurasi model <i>machine learning</i> dengan parameter TP (True Positive), TN (True Negative), FP (False Positive), FN (False Negative).</li><li>- Pengujian data <i>dummy</i> dilakukan dengan memanfaatkan gambar-gambar wajah menggunakan dan tanpa masker dari internet. Diujikan pada lebih dari 100 gambar pada <i>angle</i> wajah yang berbeda-beda. Hanya wajah dengan sudut kemiringan wajah <math>\pm 60^\circ</math> saja yang diperhitungkan pada perhitungan akurasi (sesuai spesifikasi <i>angle</i> wajah). Dihitung akurasi berdasarkan rumus perhitungan akurasi model <i>machine learning</i> dengan parameter TP (True Positive), TN (True Negative), FP (False Positive), FN (False Negative).</li></ul>

### 5.1.1 Pengujian Data Dummy

Pengujian akurasi pertama adalah pengukuran dengan menggunakan data *dummy*. Penulis mempersiapkan 35 foto untuk diuji. Dihitung semua hasil deteksi dan dijadikan 4 golongan dengan penjelasan sebagai berikut.

#### 1) True Positive (TP)

Merupakan hasil deteksi objek wajah menggunakan masker (Mask) dan wajah memang pada kondisi aktualnya menggunakan masker.

#### 2) True Negative (TN)

Merupakan hasil deteksi objek wajah tidak menggunakan masker (NoMask) dan wajah memang pada kondisi aktualnya tidak menggunakan masker.

#### 3) False Positive (FP)

Merupakan hasil deteksi objek wajah menggunakan masker (Mask) namun wajah pada kondisi aktualnya tidak menggunakan masker.

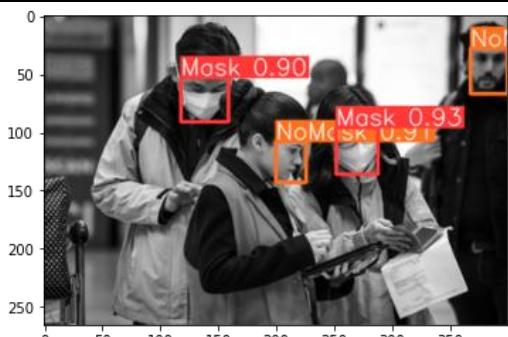
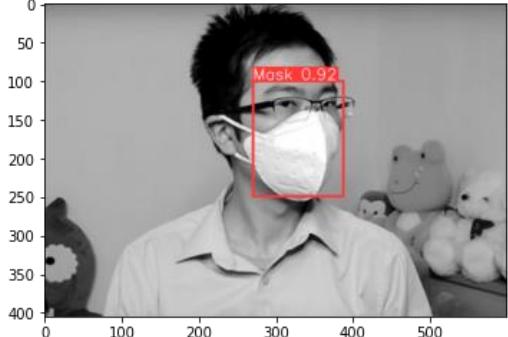
#### 4) False Negative (FN)

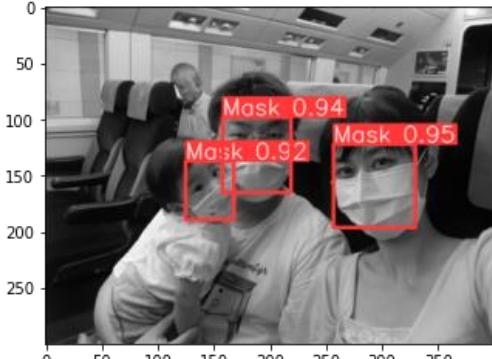
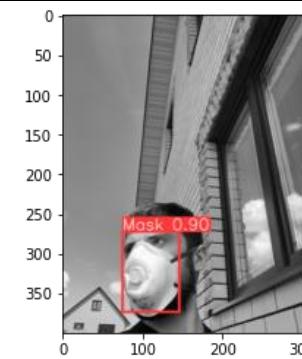
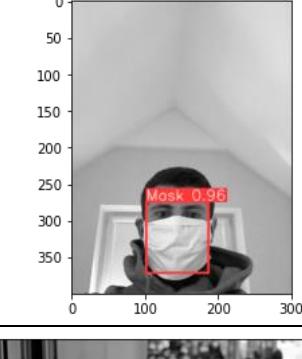
Merupakan hasil deteksi objek wajah tidak menggunakan masker (NoMask) namun wajah pada kondisi aktualnya menggunakan masker.

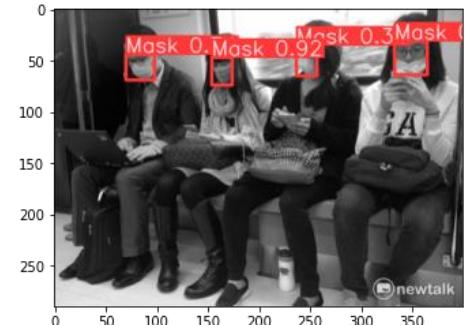
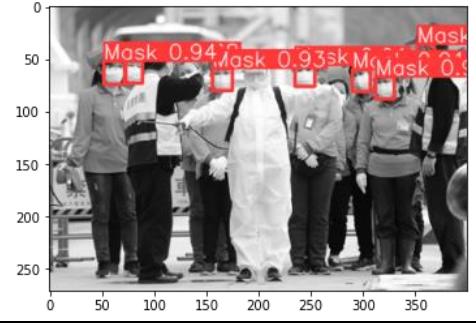
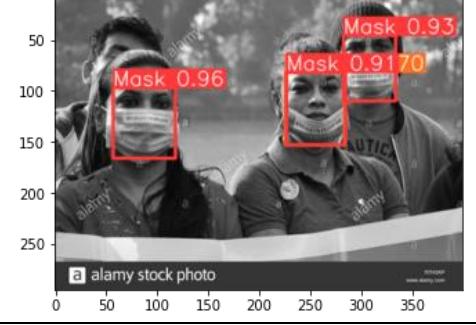
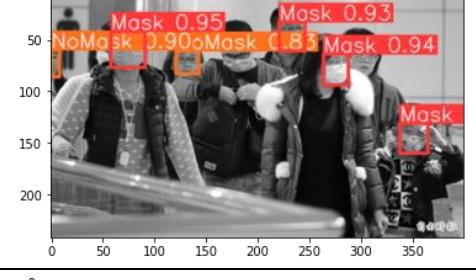
Berikut adalah hasil pengujinya.

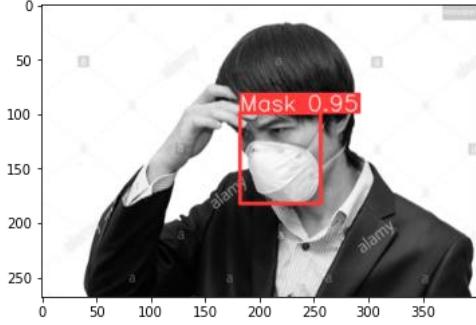
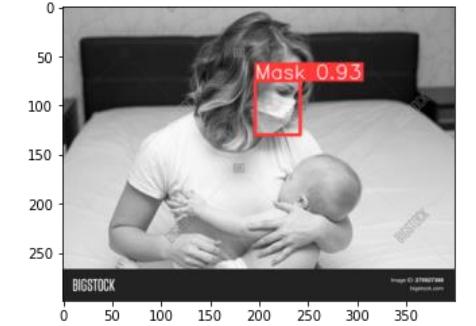
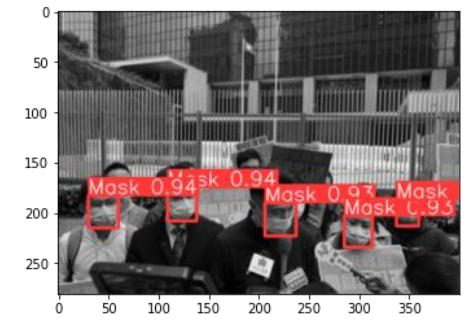
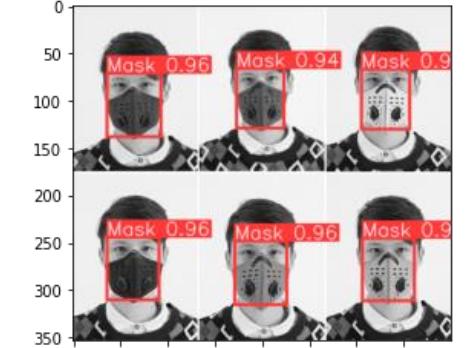
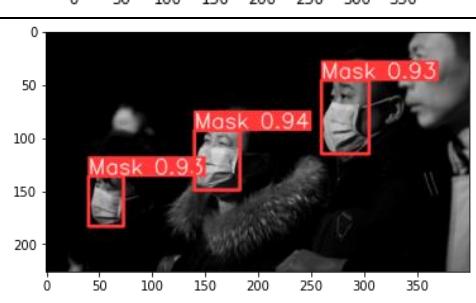
#### Proses pengujian :

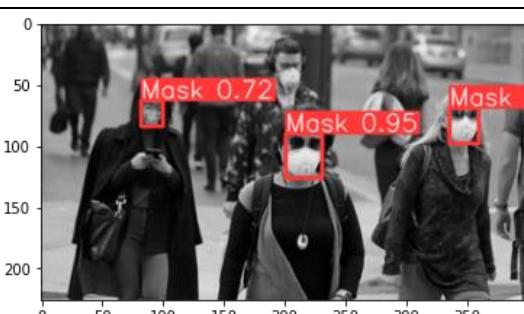
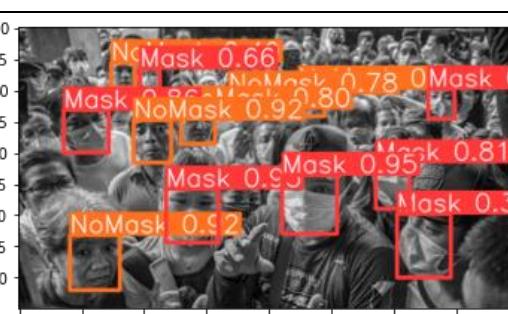
Table 5.1-2 Hasil Pengujian Akurasi Menggunakan Data Dummy

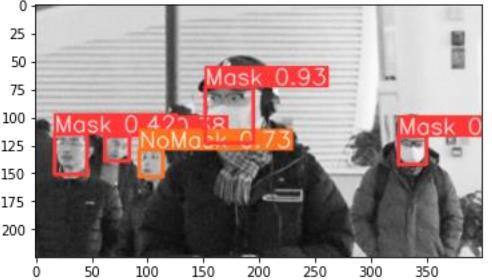
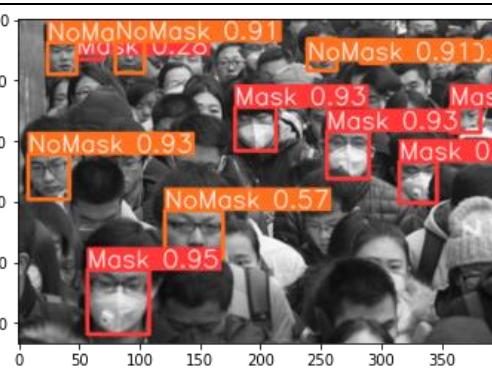
No	TP	TN	FP	FN	Foto
1	2	2	0	0	
2	1	0	0	0	

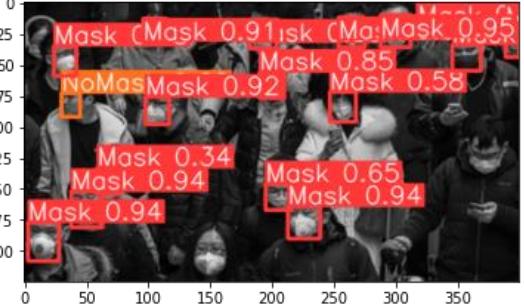
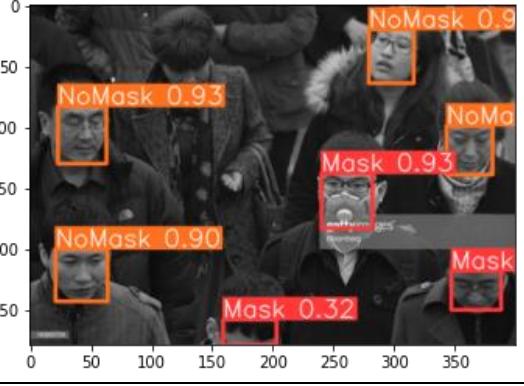
3	3	0	0	0	
4	1	0	0	0	
5	1	0	0	0	
6	6	0	1	0	
7	1	1	0	0	

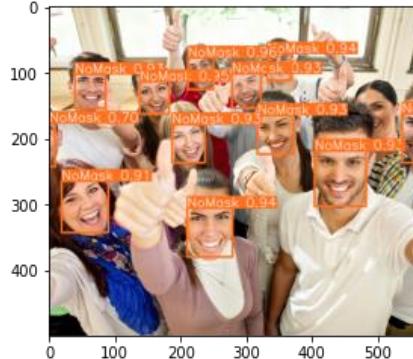
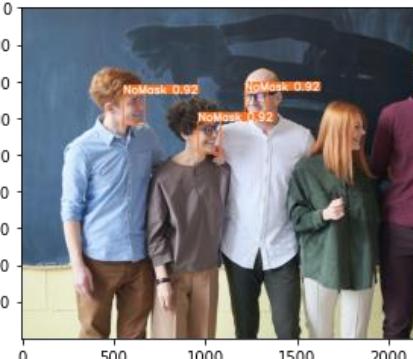
8	3	1	0	0	
9	4	0	0	0	
10	7	0	0	0	
11	3	0	0	1	
12	4	2	0	0	
13	3	1	0	0	

14	1	0	0	0	
15	1	0	0	0	
16	5	0	0	0	
17	6	0	0	0	
18	3	0	0	1	

19	7	0	0	0	
20	3	0	0	0	
21	3	0	0	0	
22	5	0	0	0	
23	6	7	0	0	

24	2	1	2	0	
25	7	0	0	0	
26	5	5	0	0	
27	3	3	0	0	
28	7	0	0	0	

29	14	1	0	0																																	
30	1	4	2	0																																	
31	0	14	0	0	<table border="1" data-bbox="743 931 1294 1245"> <thead> <tr> <th rowspan="2">Original</th> <th rowspan="2">Lighting changed</th> <th rowspan="2">Pose changed</th> <th colspan="4">Noise added</th> </tr> <tr> <th>15%</th> <th>30%</th> <th>45%</th> <th>60%</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>	Original	Lighting changed	Pose changed	Noise added				15%	30%	45%	60%																					
Original	Lighting changed	Pose changed	Noise added																																		
			15%	30%	45%	60%																															
32	0	48	0	0																																	

33	0	11	0	0	
34	0	4	0	0	
35	0	5	0	0	
Total	118	110	5	2	

Berikut adalah rekapan hasil pengujian akurasi menggunakan data *dummy* system yang dirancang.

**Table 5.1-3 Rekapan Hasil Pengujian Akurasi Menggunakan Data *Dummy***

<b>TP (True Positive)</b>	<b>118</b>
<b>FP (False Positive)</b>	<b>110</b>
<b>TN (True Negative)</b>	<b>5</b>
<b>FN (False Negative)</b>	<b>2</b>
<b>Total</b>	<b>235</b>
<b>Accuracy (%)</b>	<b>97.02</b>

Dapat dilihat pada table 5.1-3 yaitu table rekapan hasil pengujian akurasi data *dummy* dihasilkan akurasi yaitu 97%. Pada spesifikasi, diinginkan spesifikasi akurasi menggunakan data *dummy* yaitu diatas 95%. Sehingga Ini mengindikasikan bahwa akurasi system yang dirancang sudah memenuhi spesifikasi yang diinginkan.

### 5.1.2 Pengujian Data *Real*

Pengujian akurasi kedua adalah pengukuran dengan menggunakan data *real*. Penulis mengujikan langsung program secara *real-time* di ruangan kantor tempat penulis bekerja. Video hasil pendektsian disimpan kemudian dilakukan peninjauan setiap 20 detik sekali. Sehingga setiap 20 detik sekali dicuplik frame dari video hasil deteksi kemudian dilakukan perhitungan akurasi. Penulis menggunakan beberapa *angle* kamera dan kondisi waktu untuk memberikan variasi terhadap pengukuran. Diberikan warna berbeda pada baris table yang menandakan bahwa kondisi *angle* kamera atau waktu pengujian berubah. Terdapat total 144 *frame* yang dilakukan peninjauan untuk akurasi. Dihitung semua hasil deteksi dan dijadikan 5 golongan dengan penjelasan sebagai berikut.

#### 1) True Positive (TP)

Merupakan hasil deteksi objek wajah menggunakan masker (Mask) dan wajah memang pada kondisi aktualnya menggunakan masker.

#### 2) True Negative (TN)

Merupakan hasil deteksi objek wajah tidak menggunakan masker (NoMask) dan wajah memang pada kondisi aktualnya tidak menggunakan masker.

#### 3) False Positive (FP)

Merupakan hasil deteksi objek wajah menggunakan masker (Mask) namun wajah pada kondisi aktualnya tidak menggunakan masker.

#### 4) False Negative (FN)

Merupakan hasil deteksi objek wajah tidak menggunakan masker (NoMask) namun wajah pada kondisi aktualnya menggunakan masker.

#### 5) Not Calculated

Merupakan wajah yang terdeteksi maupun tidak terdeteksi baik itu benar atau salah yang berada diluar batas spesifikasi (jarak lebih dari 5 meter, angle sisi wajah tidak berada pada rentang  $\pm 60^\circ$ , dan terdapat *motion* sehingga wajah tidak tampak jelas)

Berikut adalah hasil pengujianya.

**Proses pengujian :**

**Table 5.1-4 Hasil Pengujian Akurasi Data Real**

No	TP	TN	FP	FN	Not Calcualted	Foto
1	1	2	0	0	0	<p>01-16-2022 Sun 13:16:16</p> <p>Face-Mask Detection</p>
2	1	0	0	0	1	<p>01-16-2022 Sun 13:16:36</p> <p>Face-Mask Detection</p>
3	1	0	0	0	0	<p>01-16-2022 Sun 13:16:56</p> <p>Face-Mask Detection</p>
4	1	1	0	0	0	<p>01-16-2022 Sun 13:17:16</p> <p>Face-Mask Detection</p>

5	1	0	0	0	0	
6	1	1	0	0	0	
7	2	0	0	0	0	
8	1	1	1	0	0	

9	0	1	2	0	0	
10	1	1	0	0	1	
11	1	0	0	0	0	
12	1	0	0	0	0	

13	1	0	1	0	0	
14	1	0	0	0	1	
15	1	0	0	0	2	
16	1	0	1	0	1	

17	1	0	0	0	2	 <p>01-16-2022 Sun 13:21:36</p> <p>150 Mask 0.71 152 Mask 0.63</p> <p>Face-Mask-Detection</p>
18	0	1	0	0	2	 <p>01-16-2022 Sun 13:21:56</p> <p>159 NoMask 0.77</p> <p>Face-Mask-Detection</p>
19	0	0	0	0	0	 <p>01-16-2022 Sun 13:22:16</p> <p>Face-Mask-Detection</p>
20	1	0	0	0	0	 <p>01-16-2022 Sun 13:22:36</p> <p>182 Mask 0.89</p> <p>Face-Mask-Detection</p>

21	1	0	0	0	1	 01-16-2022 Sun 16:36:20 Face-Mask Detection
22	0	0	0	0	2	 01-16-2022 Sun 16:36:40 Face-Mask Detection
23	0	0	0	0	4	 01-16-2022 Sun 16:37:00 Face-Mask Detection
24	1	0	0	0	1	 01-16-2022 Sun 16:37:20 Face-Mask Detection

25	0	0	0	1	1	 Face-Mask Detection
26	1	0	0	0	1	 Face-Mask Detection
27	1	0	0	0	0	 Face-Mask Detection
28	1	0	0	0	0	 Face-Mask Detection

29	1	0	0	0	2	<p>01-16-2022 Sun 16:39:00</p> <p>67 Mask 0.92</p> <p>82 NoMask 0.73</p> <p>Face-Mask Detection</p>
30	1	0	0	0	2	<p>01-16-2022 Sun 16:39:20</p> <p>67 Mask 0.93</p> <p>82 NoMask 0.66</p> <p>Face-Mask Detection</p>
31	1	0	0	0	2	<p>01-16-2022 Sun 16:39:40</p> <p>67 Mask 0.94</p> <p>82 NoMask 0.86</p> <p>Face-Mask Detection</p>
32	0	0	1	0	2	<p>01-16-2022 Sun 16:40:00</p> <p>82 NoMask 0.88</p> <p>Face-Mask Detection</p>

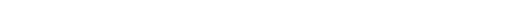
33	0	0	1	0	2	
34	1	0	0	0	1	
35	1	0	0	0	0	
36	0	0	0	1	1	

37	1	0	0	0	1	 01-16-2022 Sun 16:41:40 Face-Mask Detection
38	1	0	0	0	1	 01-16-2022 Sun 16:42:00 Face-Mask Detection
39	1	0	0	0	1	 01-16-2022 Sun 16:42:20 Face-Mask Detection
40	1	0	0	0	1	 01-16-2022 Sun 16:42:40 Face-Mask Detection

41	0	0	1	0	1	
42	0	0	1	0	1	
43	0	0	1	0	1	
44	0	0	1	0	1	

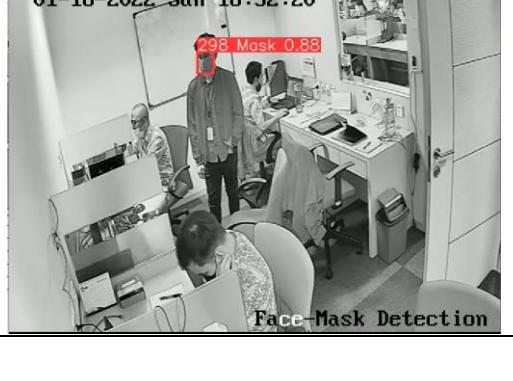
45	0	0	1	0	1	
46	0	0	1	0	1	
47	0	0	1	0	1	
48	1	1	0	0	0	

49	1	0	0	1	0	 Face-Mask Detection
50	1	1	0	0	1	 Face-Mask Detection
51	0	1	0	0	2	 Face-Mask Detection
52	0	1	0	0	2	 Face-Mask Detection

53	1	1	0	0	0	 <p>01-16-2022 Sun 16:47:40</p> <p>Face-Mask Detection</p>
54	2	0	0	0	0	 <p>01-16-2022 Sun 16:48:00</p> <p>Face-Mask Detection</p>
55	1	0	0	1	0	 <p>01-16-2022 Sun 16:48:20</p> <p>Face-Mask Detection</p>
56	0	0	0	1	1	 <p>01-16-2022 Sun 16:48:20</p> <p>Face-Mask Detection</p>

57	0	1	0	0	0	
58	0	0	0	0	2	
59	0	0	0	0	2	
60	1	0	0	0	1	

61	1	0	0	1	0	 <p>01-16-2022 Sun 16:50:00</p> <p>Face-Mask Detection</p>
62	0	0	0	0	1	 <p>01-16-2022 Sun 16:50:20</p> <p>Face-Mask Detection</p>
63	0	0	0	0	0	 <p>01-16-2022 Sun 16:50:40</p> <p>Face-Mask Detection</p>
64	0	0	0	1	1	 <p>01-16-2022 Sun 16:51:00</p> <p>Face-Mask Detection</p>

65	0	0	0	1	1	
66	2	0	0	0	0	
67	0	0	0	0	1	
68	1	0	0	0	1	

69	1	0	0	0	1	
70	1	0	0	0	1	
71	1	0	0	0	1	
72	1	0	0	0	1	

73	1	0	0	0	1	 Face-Mask Detection
74	0	0	0	1	1	 Face-Mask Detection
75	1	0	0	0	1	 Face-Mask Detection
76	1	0	0	0	1	 Face-Mask Detection

77	1	0	0	0	1	 01-16-2022 Sun 16:55:20 354 Mask 0.86 Face-Mask Detection
78	0	0	0	1	1	 01-16-2022 Sun 16:55:40 360 NoMask 0.51 Face-Mask Detection
79	1	0	0	0	1	 01-16-2022 Sun 16:56:00 354 Mask 0.91 Face-Mask Detection
80	1	0	0	0	1	 01-16-2022 Sun 16:56:20 368 Mask 0.86 Face-Mask Detection

81	1	0	0	0	1	 Face-Mask Detection
82	1	0	0	0	1	 Face-Mask Detection
83	1	0	0	0	1	 Face-Mask Detection
84	1	0	0	0	1	 Face-Mask Detection

85	1	0	0	0	1	
86	1	0	0	0	1	
87	0	1	0	0	1	
88	0	0	0	0	2	

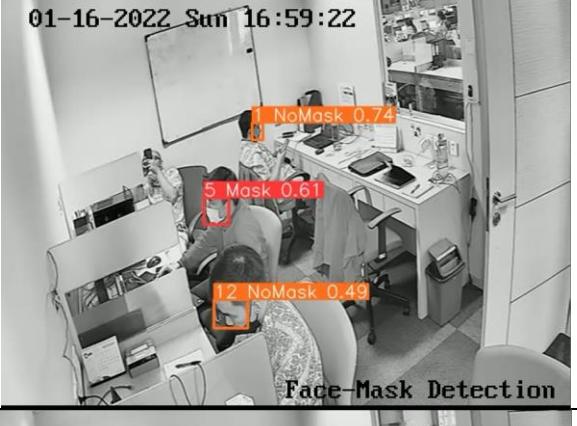
89	1	0	0	0	0	 Face-Mask Detection
90	1	0	0	0	0	 Face-Mask Detection
91	0	1	0	0	1	 Face-Mask Detection
92	0	1	0	0	1	 Face-Mask Detection

93	0	1	0	0	1	
94	0	1	0	0	1	
95	0	0	0	0	2	
96	1	0	0	0	0	

97	1	0	0	0	0	
98	1	0	0	0	0	
99	2	0	0	0	1	
100	1	0	0	0	1	

101	0	1	0	0	1	
102	0	1	0	0	1	
103	0	1	0	0	1	
104	0	1	0	0	1	

105	0	1	0	0	2	
106	0	1	0	0	2	
107	1	0	0	0	2	
108	1	0	0	0	2	

109	1	0	0	0	1	 <p>01-16-2022 Sun 16:02:00 139 Mask 0.90 Face-Mask Detection</p>
110	1	0	0	0	1	 <p>01-16-2022 Sun 16:03:00 139 Mask 0.87 Face-Mask Detection</p>
111	1	0	0	0	2	 <p>01-16-2022 Sun 16:59:22 1 NoMask 0.74 5 Mask 0.61 12 NoMask 0.49 Face-Mask Detection</p>
112	1	0	0	0	1	 <p>01-16-2022 Sun 16:59:42 5 Mask 0.88 12 NoMask 0.77 Face-Mask Detection</p>

113	1	0	0	0	1	 <p>01-16-2022 Sun 17:00:02</p> <p>Face-Mask Detection</p>
114	1	0	0	0	1	 <p>01-16-2022 Sun 17:00:22</p> <p>Face-Mask Detection</p>
115	1	0	0	0	1	 <p>01-16-2022 Sun 17:00:42</p> <p>Face-Mask Detection</p>
116	1	0	0	0	1	 <p>01-16-2022 Sun 17:01:02</p> <p>Face-Mask Detection</p>

117	1	0	0	0	1		 01-16-2022 Sun 17:01:22 Face-Mask Detection
118	1	0	0	0	1		 01-16-2022 Sun 17:01:42 Face-Mask Detection
119	1	0	0	0	1		 01-16-2022 Sun 17:02:02 Face-Mask Detection
120	1	0	0	0	1		 01-16-2022 Sun 17:02:22 Face-Mask Detection

121	1	0	0	0	1	
122	0	1	0	0	1	
123	0	1	0	0	1	
124	0	1	0	0	1	

125	1	0	0	0	1	 <p>Face-Mask Detection</p>
126	1	0	0	0	1	 <p>Face-Mask Detection</p>
127	1	0	0	0	1	 <p>Face-Mask Detection</p>
128	1	0	0	0	1	 <p>Face-Mask Detection</p>

129	1	0	0	0	1	 01-16-2022 Sun 17:05:22 Face-Mask Detection
130	1	0	0	0	1	 01-16-2022 Sun 17:05:42 Face-Mask Detection
131	1	0	0	0	1	 01-16-2022 Sun 17:06:02 Face-Mask Detection
132	1	0	0	0	1	 01-16-2022 Sun 17:06:22 Face-Mask Detection

133	1	0	0	0	1		 01-16-2022 Sun 17:06:42 Face-Mask Detection	
134	1	0	0	0	1		 01-16-2022 Sun 17:07:02 Face-Mask Detection	
135	0	2	0	0	1		 01-16-2022 Sun 17:15:32 Face-Mask Detection	
136	0	2	0	0	1		 01-16-2022 Sun 17:15:52 Face-Mask Detection	
137	0	1	0	0	1		 01-16-2022 Sun 17:16:12 Face-Mask Detection	

138	1	1	0	0	1			Face-Mask Detection
139	1	1	0	0	1			Face-Mask Detection
140	1	1	0	0	1			Face-Mask Detection
141	1	1	0	0	1			Face-Mask Detection
142	1	1	0	0	0			Face-Mask Detection

143	0	1	0	0	2			
144	0	1	0	0	1			
<b>Total</b>	<b>98</b>	<b>40</b>	<b>14</b>	<b>10</b>	<b>137</b>			

Berikut adalah rekapan hasil pengujian akurasi menggunakan data *real* system yang dirancang.

**Table 5.1-5 Rekapan Hasil Pengujian Akurasi Menggunakan Data Real**

<b>TP (True Positive)</b>	<b>98</b>
<b>FP (False Positive)</b>	<b>40</b>
<b>TN (True Negative)</b>	<b>14</b>
<b>FN (False Negative)</b>	<b>10</b>
<b>Not Calculated</b>	<b>137</b>
<b>Total</b>	<b>299</b>
<b>Accuracy (%)</b>	<b>85.18</b>

Dapat dilihat pada table 5.1-5 yaitu table rekapan hasil pengujian akurasi data *real* dihasilkan akurasi yaitu 85%. Pada spesifikasi, diinginkan spesifikasi akurasi menggunakan data *real* yaitu diatas 80%. Sehingga Ini mengindikasikan bahwa akurasi system yang dirancang sudah memenuhi spesifikasi yang diinginkan.

## 5.2 Pengujian Spesifikasi Jarak Deteksi Objek

Table 5.2-1 Verifikasi Spesifikasi Jarak Deteksi Objek

Hal	1. Jarak
Rincian	1. < 5 Meter
Metode Pengukuran	Pengujian dilakukan dengan menggunakan meteran digital (meteran laser)
Prosedur Pengujian	<ul style="list-style-type: none"> <li>- Pengujian dilakukan dengan menembakkan laser meteran ke arah lensa IP Camera pada ketinggian 2.3 meter (sesuai spesifikasi ketinggian) dari ujung sisi terjauh dari kamera pada ruangan kerja. Nilai yang terbaca pada meteran harus lebih dari atau sama dengan 5 meter agar spesifikasi jarak ini terpenuhi.</li> </ul>

Proses pengujian :



Gambar 5.2-1 Hasil Pembacaan Meteran Pengukuran Jarak

Dapat dilihat pada Gambar 5.2-1 yaitu hasil pembacaan meteran untuk mengukur jarak terjauh sisi ruangan dari lensa kamera. Pada spesifikasi, diinginkan spesifikasi jarak objek yaitu kurang dari 5 meter. Diperoleh hasil pengukuran yaitu 5.145 meter. Sehingga ini mengindikasikan bahwa ruangan yang dijadikan tempat pengujian akurasi ini merupakan sarana yang valid untuk digunakan.

## 5.3 Pengujian Spesifikasi Ketinggian Kamera

Table 5.3-1 Verifikasi Spesifikasi Ketinggian Kamera

Hal	1. Ketinggian
Rincian	1. < 2.3 Meter
Metode Pengukuran	Pengujian dilakukan dengan menggunakan meteran digital (meteran laser)

Prosedur Pengujian	<ul style="list-style-type: none"> <li>- Pengujian dilakukan dengan menembakkan laser meteran ke arah lensa IP Camera dari tanah dengan sudut kemiringan <math>0^\circ</math>. Nilai yang terbaca pada meteran harus lebih dari atau sama dengan 2.3 meter agar spesifikasi ketinggian ini terpenuhi.</li> </ul>
--------------------	--

### Proses pengujian :



**Gambar 5.3-1 Pengukuran Ketinggian Kamera**

Dapat dilihat pada Gambar 5.3-1 yaitu proses pengujian spesifikasi ketinggian, penulis menghitung ketinggian tripod dan ketinggian rak tempat tripod tersebut berdiri. Diperoleh hasil pengukuran ketinggian tripod yaitu 1.5 meter dan ketinggian rak yaitu 0.8 meter. Sehingga diperoleh ketinggian total dari kamera adalah 2.3 meter. Pada spesifikasi, diinginkan spesifikasi ketinggian kamera yaitu kurang dari 2.3 meter. Sehingga ini mengindikasikan bahwa pelektakkan kamera untuk menggunakan system yang dirancang sudah memenuhi spesifikasi yang diinginkan.

### 5.4 Pengujian Spesifikasi Angle Deteksi Objek

**Table 5.4-1 Verifikasi Spesifikasi Angle Deteksi Objek**

Hal	1. Angle Sisi Wajah (a)
Rincian	1. $-60^\circ < a < 60^\circ$
Metode Pengukuran	Pengujian dilakukan dengan data <i>real</i> yaitu wajah penulis dari berbagai variasi sisi wajah.
Prosedur Pengujian	Pengujian dilakukan dengan menghadapkan wajah pada berbagai sisi. Pada pengujian data <i>real</i> , ini dilakukan pada kondisi ketinggian kamera 2.3 meter dan jarak objek maksimum adalah 5 meter (sesuai spesifikasi jarak dan

	ketinggian). Patokan 60 derajat ialah selama kedua landmark mata pada wajah masih terlihat.
--	---

### Proses pengujian :

Pada pengujian spesifikasi ini hanya melakukan peninjauan terhadap wajah dengan sisi  $\pm 60^\circ$  untuk menghitung akurasi. Penulis memperoleh sudut  $\pm 60^\circ$  itu merupakan representasi Ketika kedua landmark mata masih dapat terlihat.

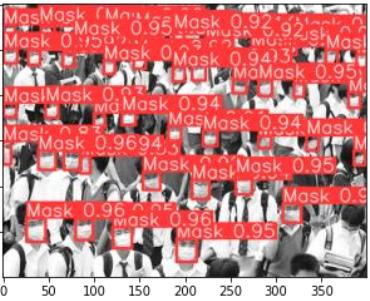
### 5.5 Pengujian Spesifikasi Jumlah Objek dalam 1 Frame

Table 5.5-1 Verifikasi Spesifikasi Jumlah Objek dalam 1 Frame

Hal	1. Jumlah Objek dalam 1 frame
Rincian	1. > 10 wajah
Metode Pengukuran	Pengujian dilakukan dengan data <i>dummy</i> (gambar wajah dengan masker dan tanpa masker dari internet)
Prosedur Pengujian	Dipersiapkan terlebih dahulu gambar-gambar yang dalam gambar tersebut lebih dari 10 wajah. Diujikan program pada input gambar tersebut dan dilihat apakah program dapat mendeteksi lebih dari 10 wajah atau tidak. Parameter keberhasilannya ketika lebih dari 10 wajah dapat dideteksi dengan baik. Diujikan pada 5 gambar.

### Proses pengujian :

Table 5.5-2 Proses Pengujian Spesifikasi Jumlah Objek dalam 1 Frame

No	Raw Data	Processed Data	Status
1			>10 Objects
2			>10 Objects

3			>10 Objects
4			>10 Objects
5			>10 Objects

Dapat dilihat pada table 5.5-2 yaitu table pengujian jumlah objek yang dapat terdeteksi dalam 1 frame. Terlihat dari 5 pengujian pada gambar, seluruhnya dapat mendeteksi frame lebih dari 10 objek. Ini mengindikasikan bahwa jumlah objek dalam 1 frame sudah sesuai dengan spesifikasi yang diinginkan. Pada program memang tidak diberikan Batasan jumlah objek yang dapat dideteksi. Sehingga jumlah objek yang masih dapat dideteksi dalam 1 frame dapat berjumlah berapapun bergantung pada kualitas gambar yang dimasukkan. Semakin kualitas gambar, maka maka akan semakin besar potensi system dapat melakukan deteksi.

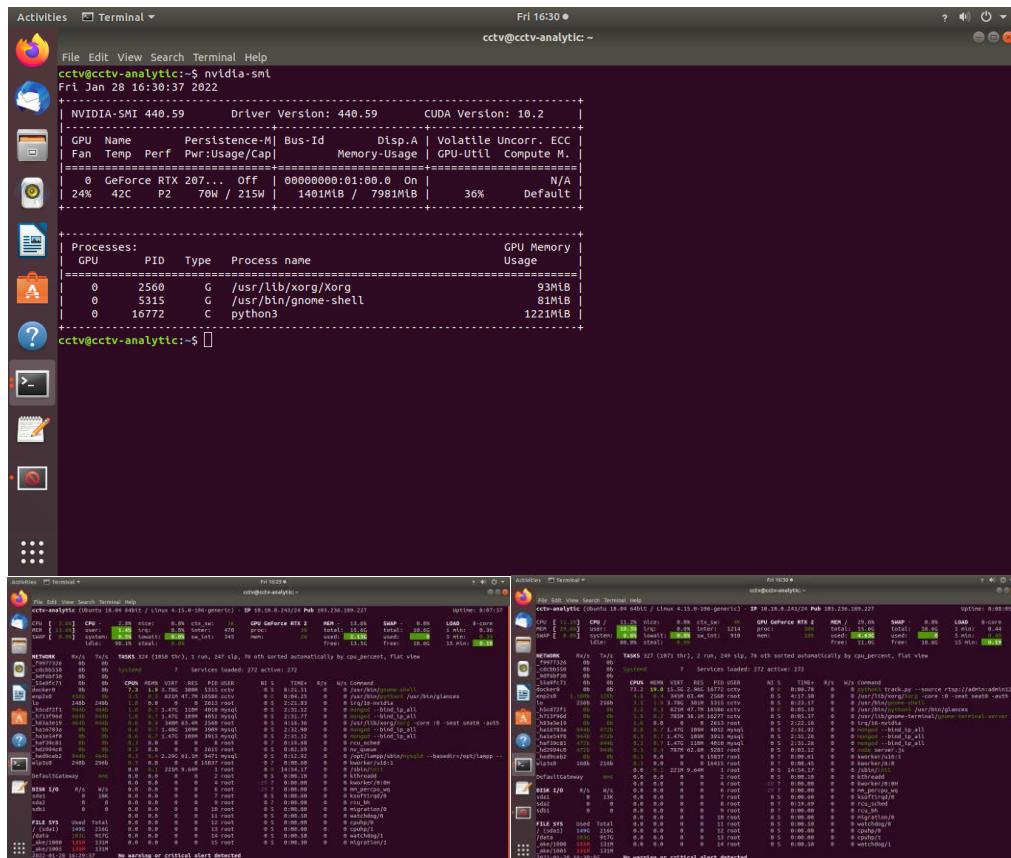
## 5.6 Pengujian Spesifikasi Jumlah Video Stream

Table 5.6-1 Verifikasi Spesifikasi Jumlah Video Stream

Hal	1. Jumlah Video Stream
Rincian	1. > 5 Video Stream
Metode Pengukuran	Pengujian dilakukan dengan meninjau pemakaian GPU, CPU, dan RAM computer terhadap 1 video stream
Prosedur Pengujian	Dijalankan program <i>face-mask detection</i> untuk input 1 IP Camera. Kemudian dilihat pada persentase pemakaian GPU, CPU, dan RAM. Untuk masing-masing komponen tersebut dikalikan dengan 5 maka hasilnya tidak boleh

	lebih dari 80% pemakaian total komponen (80% GPU, 80% CPU, 80% RAM). Dengan kata lain diberikan <i>spare</i> sebesar 20% agar computer tidak mengalami <i>overload</i> .
--	--

## Proses pengujian :



Gambar 5.6-1 Snapshot Pengukuran Konsumsi CPU, GPU, dan RAM

Table 5.6-2 Rekapan Pengukuran Konsumsi CPU, GPU, dan RAM

	Capacity	Measurement	5× Measurement	Percent of Usage (%)
CPU	100%	9.2%	46%	46%
GPU	8 GB	1.221 GB	6.1 GB	76%
RAM	16 GB	2.5 GB	12.5 GB	78%

Dapat dilihat pada table 5.6-2 yaitu hasil pengukuran CPU, GPU, dan RAM. Dapat untuk 1 *stream* program, menggunakan 9.2% CPU, 1.221 GB GPU, dan 2.5 GB RAM. Penulis hanya mengujikan pada 1 IP Camera karena keterbatasan jumlah perangkat kamera yang ada. Sehingga untuk pengujian spesifikasi jumlah *stream* video, dilakukan pendeketan dengan meninjau penggunaan CPU, GPU, dan RAM untuk 1 IP Camera. Dapat dilihat pada kolom Percent of Usage, konsumsi CPU, GPU, dan RAM untuk 5 video *stream* masih berada dibawah 80%. Ini mengindikasikan bahwa jumlah *video stream* program yang masih

dapat berjalan dengan baik sudah memenuhi spesifikasi yang diinginkan yaitu berjumlah 5 buah *stream*.

## 5.7 Pengujian Spesifikasi Penyimpanan Data

Table 5.7-1 Verifikasi Spesifikasi Penyimpanan Data

Hal	1. Penyimpanan Data
Rincian	1. Data dengan id dan label yang sama hanya dapat 1 kali tersimpan pada database
Metode Pengukuran	Pengujian dilakukan dengan menjalankan program <i>face-mask detection</i> kemudian melihat data hasil deteksi pada database
Prosedur Pengujian	Dijalankan program <i>face-mask detection</i> . Data hasil deteksi program kemudian tersimpan pada database. Ditinjau lebih lanjut apakah terdapat 2 data dengan id dan label yang sama tersimpan pada database. Diujikan minimal pada 20 data yang tersimpan.

### Proses pengujian :

no	date	id	cls	bbox_left	bbox_top	bbox_w	bbox_h	label
1	2022-01-28 14:36:24	1	0	250	295	149	179	Mask
2	2022-01-28 14:36:28	1	2	247	283	148	191	NoMask
3	2022-01-28 14:36:31	3	2	247	295	153	184	NoMask
4	2022-01-28 14:36:34	4	2	251	414	199	65	NoMask
5	2022-01-28 14:36:39	5	0	262	372	180	107	Mask
6	2022-01-28 14:36:43	6	0	277	350	170	129	Mask
7	2022-01-28 14:36:50	7	0	258	369	176	110	Mask
8	2022-01-28 14:36:54	7	2	254	285	161	194	NoMask
9	2022-01-28 14:37:02	10	0	255	400	184	79	Mask
10	2022-01-28 14:37:02	11	2	260	397	189	82	NoMask
11	2022-01-28 14:37:11	12	0	270	370	181	109	Mask
12	2022-01-28 14:37:24	12	2	257	304	162	175	NoMask
13	2022-01-28 14:37:28	13	2	245	425	257	54	NoMask
14	2022-01-28 14:37:37	14	0	270	353	169	126	Mask
15	2022-01-28 14:37:42	15	0	232	385	182	94	Mask
16	2022-01-28 14:37:47	16	2	241	375	191	104	NoMask
17	2022-01-28 14:37:56	17	0	258	385	193	94	Mask
18	2022-01-28 14:38:06	20	0	263	320	240	159	Mask
19	2022-01-28 14:38:21	21	2	261	299	161	180	NoMask
20	2022-01-28 14:38:24	23	2	244	393	186	86	NoMask
	2022-01-28 14:38:30	24	0	263	405	192	74	Mask

Gambar 5.7-1 Hasil Pengujian Spesifikasi Penyimpanan Data

Dapat dilihat pada gambar 5.7-1 yaitu tampilan database setelah menjalankan program *face-mask detection*. Terdapat informasi-informasi mengenai jumlah data (no), *timestamp* (date), id, nomor kelas (cls), koordinat *bounding box* (bbox\_left, bbox\_top, bbox\_w, bbox\_h), dan nama kelas. Program penyimpanan sudah dirancang sehingga data yang masuk ke database tidak mengalami pengulangan. Terlihat pada gambar tidak terdapat data yang sama untuk id dan label yang sama. Ini mengindikasikan bahwa penyimpanan data sudah sesuai dengan spesifikasi yang diinginkan.

Penyimpanan juga sudah dirancang untuk mengantisipasi perangkat yang ter-*shutdown* yang mengakibatkan program harus di-*restart*. Ini dilakukan dengan mencari id tertinggi dari data terakhir pada database kemudian melanjutkan *counting* dengan acuan id tertinggi tersebut. Sehingga tetap tidak terdapat pengulangan data untuk id dan label yang sama.

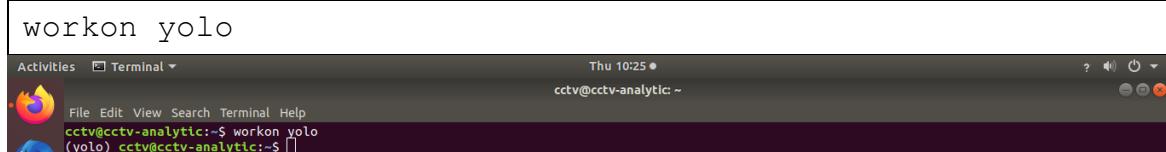
## 6 Guideline Penggunaan Program

### 6.1 Prosedur Aktivasi Virtual Environment

1. Membuka Terminal Linux
2. Aktivasi Virtual Environment

Selanjutnya adalah melakukan aktivasi *Virtual Environment* (VE) yang telah dibuat untuk proyek *face-mask detection*. VE yang dibuat diberi nama “yolo”. Perintah yang dapat digunakan pada terminal untuk mengaktifkan VE adalah seperti berikut.

```
workon yolo
```



Gambar 6.1-1 Perintah Aktivasi *Virtual Environment*

Dapat dilihat pada Gambar 6.1-1 diatas, indikasi keberhasilan mengaktivasi *Virtual Environment* yolo adalah tertulis pada bagian kiri yaitu nama dari VE tersebut. Dengan mengaktifkan VE Bernama “yolo”, komputer yang berjalan akan dibatasi untuk bekerja hanya pada environment “yolo” saja. Sehingga prosesnya tidak akan tercampur dengan program-program lain yang ada pada komputer.

### 6.2 Prosedur Labelling Dataset

1. Membuka Terminal Linux
2. Aktivasi *Virtual Environment* (Prosedur 6.1)
3. Memindahkan Direktori Kerja

Untuk dapat melakukan Labelling Dataset, setelah VE diaktifkan, dilakukan pemindahan direktori kerja pada terminal. Direktori kerja labelImg terletak pada folder **/Documents/face\_mask/labelImg**. Perintah yang dapat digunakan pada terminal untuk memindahkan direktori kerja program adalah seperti berikut.

```
cd Documents/face_mask/labelImg
```



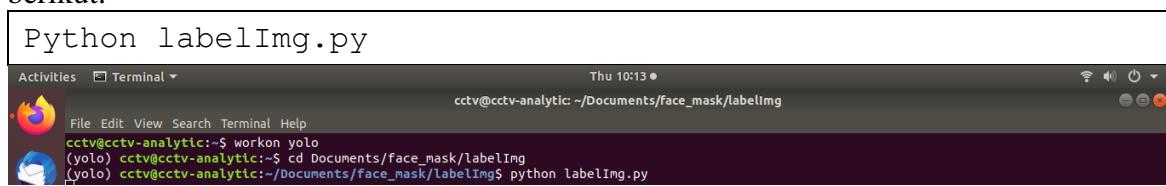
Gambar 6.2-1 Perintah Mengubah Direktori Kerja labelImg

Dapat dilihat pada Gambar 6.2-1 diatas, terlihat direktori kerja sudah berubah yaitu pada folder **Documents/face\_mask/labelImg**.

4. *Running* labelImg

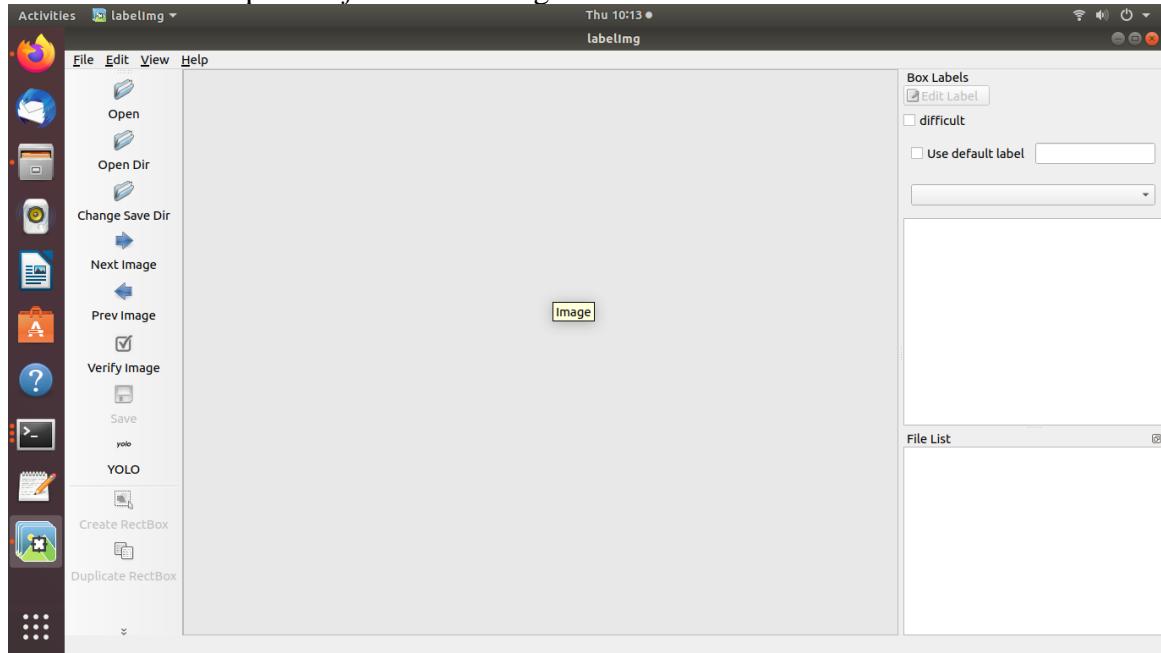
Setelah itu dapat dijalankan aplikasi labelImg dengan menuliskan kode perintah seperti berikut.

```
Python labelImg.py
```



Gambar 6.2-2 Perintah *Running* Software LabelImg

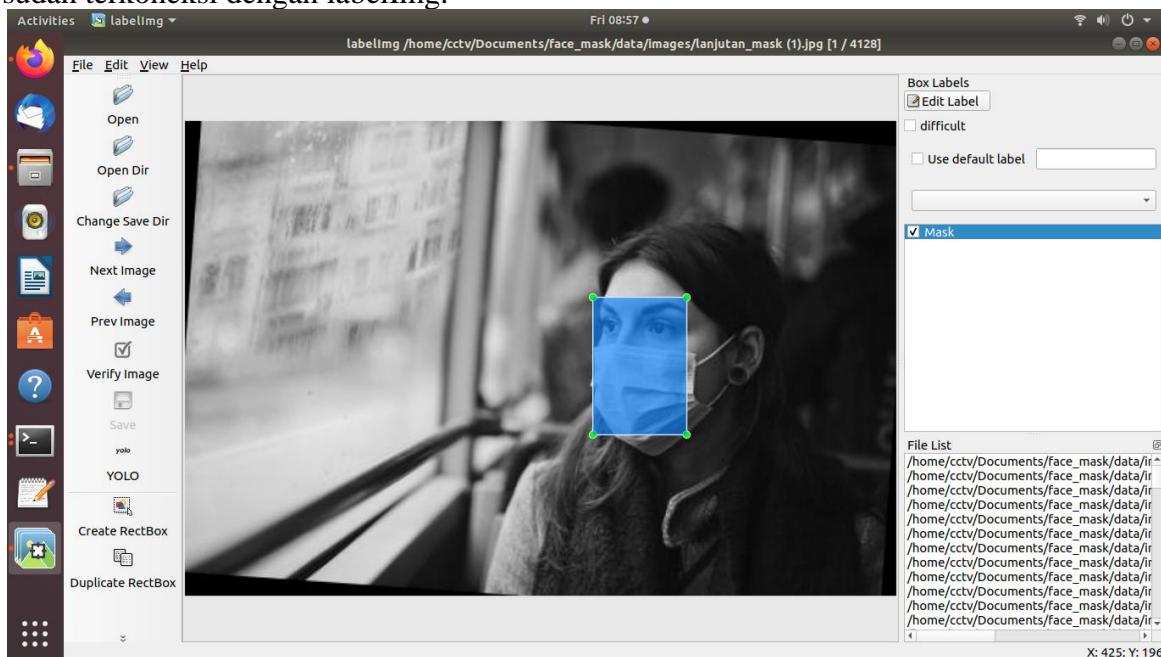
Berikut adalah tampilan *software* labelImg.



Gambar 6.2-3 Tampilan *Software* labelImg

### 5. Labelling Dataset

Untuk melihat hasil dataset yang telah dilabel, pilih gambar yang ingin pengecekan. Pada proyek ini, dataset disimpan pada alamat **/Documents/face\_mask/data/images**. Sehingga pada *navigation bar* “Open Dir” (Gambar 6.2-3) pilih alamat dataset tersebut. Selain itu, file hasil pelabelan dataset disimpan pada alamat **/Documents/face\_mask/data/label**. Sehingga pada *navigation bar* “Change Save Dir” (Gambar 6.2-3) ke alamat file hasil pelabelan tersebut. Berikut adalah tampilan Ketika folder dataset wajah dan hasil pelabelan sudah terkoneksi dengan labelImg.



Gambar 6.2-4 Tampilan Dataset yang Telah Dilabel dengan labelImg

Pada proyek *face-mask detection* ini, terdapat 4128 dataset gambar yang terdiri dari 1531 data wajah dengan masker dan 2597 data wajah tanpa masker.

Apabila diinginkan penambahan dataset, data dapat disimpan pada folder dataset (**/Documents/face\_mask/data/images**). Kemudian dilakukan pelabelan pada labelImg dengan mengakses gambar tersebut dan membuat *bounding box* sebagai *Region of Interest* dari gambar.

## 6. Output Pelabelan Dataset

Proses pelabelan data ini dilakukan pada semua data yang ada pada dataset. Label mengidentifikasi vektor data yang sesuai untuk ditarik sebagai acuan pelatihan model, di mana model, kemudian, belajar membuat prediksi terbaik. Dikarenakan algoritma pemrograman ML menggunakan YOLO, maka output dari hasil labelling data ini adalah berupa file anotasi dengan ekstensi .txt. Output file hasil pelabelan dataset disimpan pada alamat **/Documents/face\_mask/data/label**. Berikut cuplikan isi dari file anotasi hasil labelling data yang dibuat.

```
with_mask--1.jpg.rf.bff0b7f683681fc1e6d3e4500616a29e.txt
1 0 0.63375 0.3298076923076923 0.13 0.2673076923076923
2 |
```

Gambar 6.2-5 Cuplikan Isi File Anotasi Data Wajah Menggunakan Masker

```
without_mask--217.jpg.rf.637ad748550d9f20db44ce704bad3a09.txt
1 2 0.540625 0.5572429906542056 0.53125 0.39953271028037385
2 |
```

Gambar 6.2-6 Cuplikan Isi File Anotasi Data Wajah Tanpa Masker

```
classes.txt
1 Mask
2 MaskIncorrect
3 NoMask|
```

Gambar 6.2-7 Isi File classes.txt

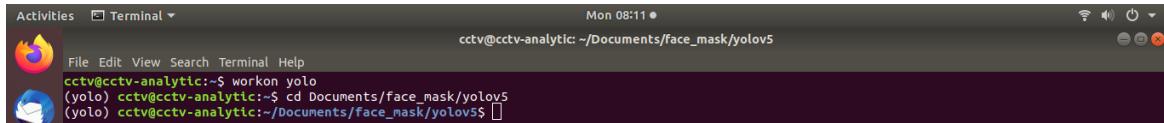
Gambar diatas merupakan gambar file anotasi dengan format YOLO. Akan secara otomatis terbuat file-file anotasi dari tiap data gambar (Gambar 6.2-5 dan Gambar 6.2-6) dan 1 file yang berisikan rekapan dari kelas-kelas yang ada pada proses pelabelan keseluruhan (Gambar 6.2-7). Format YOLO memiliki spesifikasi yaitu [class] [x\_center] [y\_center] [width] [height]. Perhitungan kelas dimulai dari indeks 0 sehingga indeks 0 merupakan representasi dari kelas Mask, indeks 1 merupakan representasi dari kelas MaskIncorrect, dan indeks 2 merupakan representasi dari kelas NoMask. Namun pada kasus ini penulis tidak menyiapkan dataset wajah pada kelas MaskIncorrect (wajah dengan penggunaan masker yang tidak tepat) dikarenakan dataset yang kurang banyak. Apabila dipaksa untuk digunakan akan memperlambat proses *training* dan akurasi yan menurun dikarenakan data yang kurang seimbang jumlahnya (*Data Imbalance*).

## 6.3 Prosedur Training Machine Learning Model Program Face-mask Detection

1. Membuka Terminal Linux
2. Aktivasi *Virtual Environment* (Prosedur 6.1)
3. Memindahkan Direktori Kerja

Untuk dapat melakukan *training*, setelah VE diaktifkan, dilakukan pemindahan direktori kerja pada terminal. Direktori kerja untuk *training* terletak pada folder **/Documents/face\_mask/yolov5**. Perintah yang dapat digunakan pada terminal untuk memindahkan direktori kerja program adalah seperti berikut.

```
cd Documents/face_mask/yolov5
```



```
Activities Terminal Mon 08:11 ●
cctv@cctv-analytic: ~/Documents/face_mask/yolov5
File Edit View Search Terminal Help
(yolo) cctv@cctv-analytic: ~$ cd Documents/face_mask/yolov5
(yolo) cctv@cctv-analytic: ~/Documents/face_mask/yolov5$
```

Gambar 6.3-1 Perintah Mengubah Direktori Kerja *Training*

Dapat dilihat pada Gambar 6.3-1 diatas, terlihat direktori kerja sudah berubah yaitu pada folder **Documents/face\_mask/yolov5**.

#### 4. Perbaharui *pipeline* dataset.yaml

File dataset.yaml digunakan sebagai *pipeline training* yang disimpan pada alamat **Documents/face\_mask/yolov5**. Isi dari file ini kode untuk mengakses data-data gambar dan pelabelan, serta *list* dari kelas yang ada pada program. *List* kelas yang ada dapat diambil dari output file hasil pelabelan (Gambar 6.2-7). Berikut isi file dataset.yaml yang telah disesuaikan dengan program *face-mask detection*.

```
! dataset.yaml
1 # Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt, or 3) list: [path/to/imgs1, path/to/imgs2, ...]
2 path: ../data
3 train: images
4 val: images
5
6 |
7 # Classes
8 nc: 3 # number of classes
9 names: ['Mask', 'MaskIncorrect', 'NoMask'] # class names
```

Gambar 6.3-2 Isi File dataset.yaml

#### 5. *Training* Pemodelan

Setelah direktori kerja sudah sesuai dan file dataset.yaml sudah disesuaikan, selanjutnya adalah proses *training* model *machine learning*. Proses *training* ini menggunakan dataset wajah dan pelabelan yang sudah dilakukan, dan file *pre-trained* model YOLO. Pada proyek ini, penulis menggunakan yolov5s sebagai *pre-trained* model program. Berikut adalah kode perintah untuk melakukan *training* pemodelan *machine learning* program.

```
python train.py --img 320 --batch 16 --epochs 450 --data
dataset.yaml --weights yolov5s.pt --workers 2
Activities Terminal Mon 09:15 ●
cctv@cctv-analytic: ~/Documents/face_mask/yolov5
File Edit View Search Terminal Help
(yolo) cctv@cctv-analytic: ~$ workon yolo
(yolo) cctv@cctv-analytic: ~$ cd Documents/face_mask/yolov5
(yolo) cctv@cctv-analytic: ~/Documents/face_mask/yolov5$ python train.py --img 320 --batch 16 --epochs 450 --data dataset.yaml --weights yolov5
.s.pt --workers 2
```

Gambar 6.3-3 Perintah Melakukan *Training Machine Learning Program Face-mask Detection*

Dapat dilihat pada Gambar 6.3-3 diatas yaitu kode perintah untuk melakukan *training*. Penulis menggunakan *image size* 320, *batch size* 16, dan *epochs* 450. Parameter ini dapat diubah sesuai dengan nilai yang diinginkan agar diperoleh hasil pemodelan yang terbaik. Berikut adalah tampilan Ketika proses *training* sedang berlangsung.

```

Activities Terminal Mon 09:32 •
cctv@cctv-analytic: ~/Documents/face_mask/yolov5

File Edit View Search Terminal Help

Epoch gpu_mem box obj cls labels img_size
3/449 1.53G 0.05447 0.01292 0.01313 27 320: 100% || 169/169 [01:16<00:00, 2.20it/s]
Class Images Labels P R mAP@.5 mAP@.5:95: 100% || 85/85 [00:25<00:00, 3.29it/s]
all 2703 2862 0.674 0.775 0.792 0.36

Epoch gpu_mem box obj cls labels img_size
4/449 1.53G 0.04883 0.0124 0.0109 24 320: 100% || 169/169 [01:16<00:00, 2.20it/s]
Class Images Labels P R mAP@.5 mAP@.5:95: 100% || 85/85 [00:25<00:00, 3.29it/s]
all 2703 2862 0.837 0.825 0.892 0.382

Epoch gpu_mem box obj cls labels img_size
5/449 1.53G 0.04863 0.0123 0.01111 22 320: 100% || 169/169 [01:16<00:00, 2.21it/s]
Class Images Labels P R mAP@.5 mAP@.5:95: 100% || 85/85 [00:25<00:00, 3.28it/s]
all 2703 2862 0.887 0.866 0.936 0.56

Epoch gpu_mem box obj cls labels img_size
6/449 1.53G 0.04327 0.01226 0.0105 25 320: 100% || 169/169 [01:15<00:00, 2.23it/s]
Class Images Labels P R mAP@.5 mAP@.5:95: 100% || 85/85 [00:25<00:00, 3.29it/s]
all 2703 2862 0.92 0.885 0.943 0.567

Epoch gpu_mem box obj cls labels img_size
7/449 1.53G 0.04155 0.01219 0.009889 36 320: 100% || 169/169 [01:14<00:00, 2.27it/s]
Class Images Labels P R mAP@.5 mAP@.5:95: 100% || 85/85 [00:25<00:00, 3.30it/s]
all 2703 2862 0.921 0.912 0.958 0.545

Epoch gpu_mem box obj cls labels img_size
8/449 1.53G 0.03885 0.01152 0.0088802 21 320: 100% || 169/169 [01:17<00:00, 2.18it/s]
Class Images Labels P R mAP@.5 mAP@.5:95: 100% || 85/85 [00:25<00:00, 3.30it/s]
all 2703 2862 0.854 0.853 0.912 0.554

Epoch gpu_mem box obj cls labels img_size
9/449 1.53G 0.03758 0.01165 0.0088524 24 320: 100% || 169/169 [01:14<00:00, 2.28it/s]
Class Images Labels P R mAP@.5 mAP@.5:95: 100% || 85/85 [00:25<00:00, 3.30it/s]
all 2703 2862 0.944 0.937 0.97 0.538

Epoch gpu_mem box obj cls labels img_size
10/449 1.53G 0.03712 0.01115 0.008293 37 320: 50% || 84/169 [00:45<00:35, 2.41it/s]

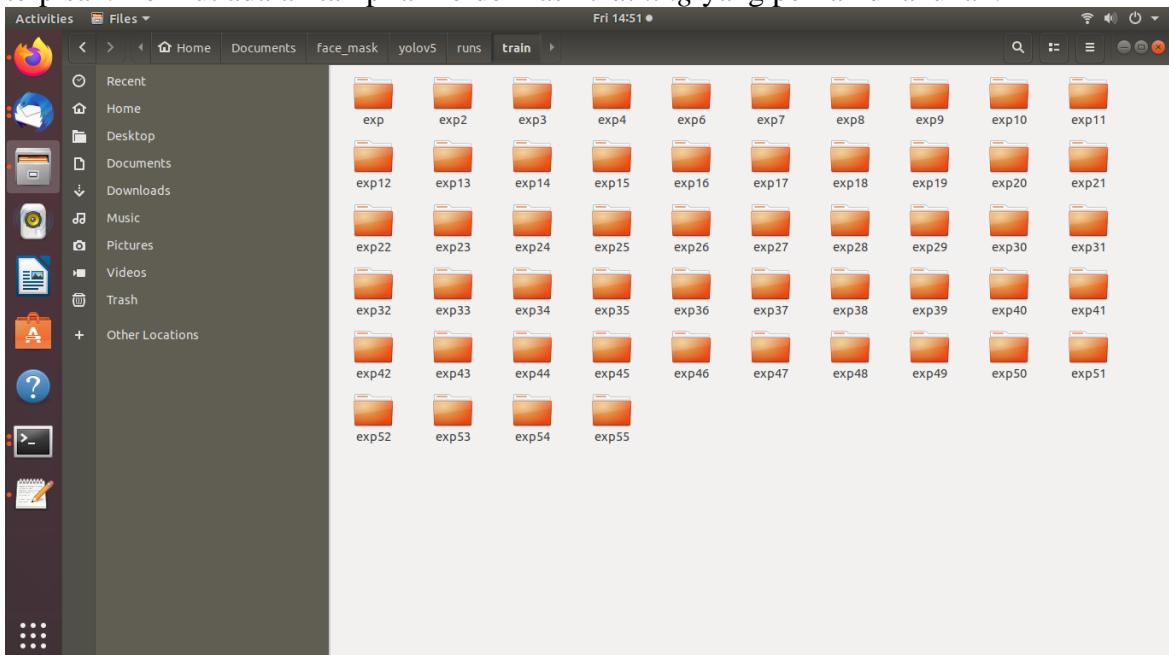
```

**Gambar 6.3-4 Proses Training Program**

Dapat dilihat pada Gambar 6.3-4 yaitu Ketika proses *training* sedang berlangsung. Akan dilakukan proses pelatihan sebanyak epochs yang dituliskan pada kode perintah terminal. Pada kasus ini, dilakukan iterasi sebanyak 450 kali. Semakin banyak iterasi yang diinginkan maka akan semakin lama waktu yang dibutuhkan untuk melakukan *training*.

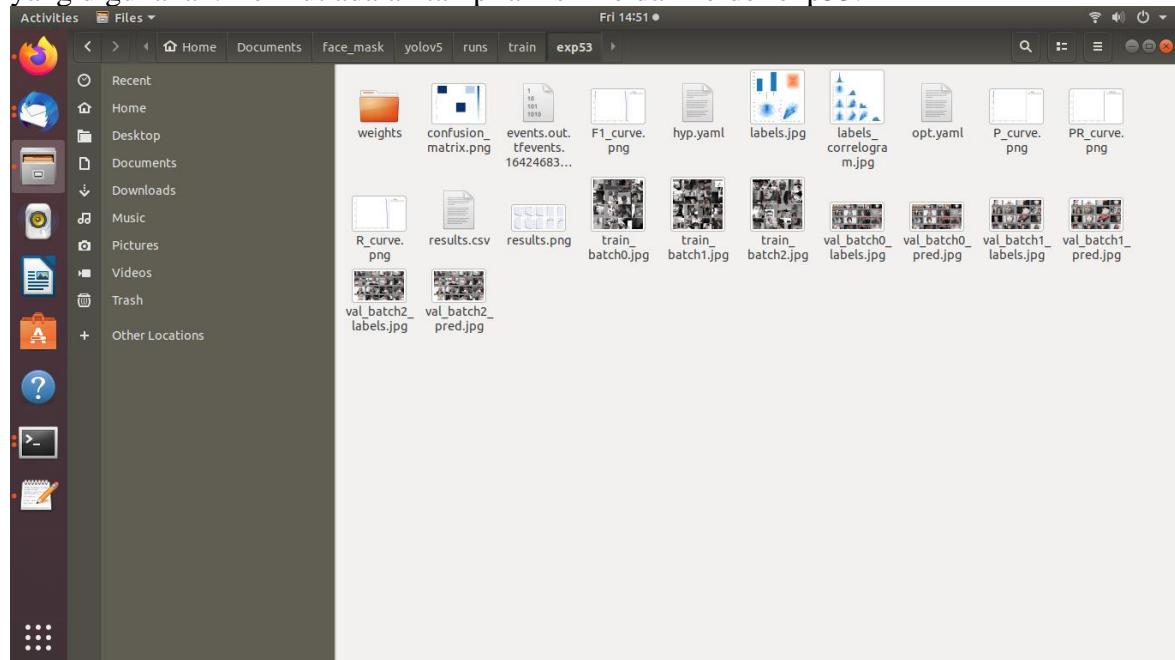
## 6. Output Training Program

Output hasil *training* secara otomatis disimpan kedalam folder dengan alamat **Documents/face\_mask/yolov5/runs/train**. Pada folder ini disimpan folder-folder hasil *training* untuk 1 kali pelatihan. Dengan kata lain setiap pelatihan disimpan pada folder yang terpisah. Berikut adalah tampilan folder hasil *training* yang pernah dilakukan.



**Gambar 6.3-5 Folder Hasil Training**

Dapat dilihat pada gambar 6.3-5 yaitu folder-folder hasil *training* yang pernah dilakukan penulis. Hasil *Training* itu merupakan tahapan-tahapan *training* yang pernah penulis lakukan dengan melakukan perubahan-perubahan terhadap jumlah dataset, jumlah iterasi, pewarnaan dataset, jenis *pre-trained* model, dan variasi-variasi lainnya. Penulis melakukan hal tersebut untuk mencapai hasil terbaik model *machine learning*. Folder hasil *training* akhir penulis yaitu pada folder **exp53** (menggunakan *pre-trained model* yolov5s), **exp54** (menggunakan *pre-trained model* yolov5m), dan **exp55** (menggunakan *pre-trained model* yolov5l). Ketiga folder tersebut berupa file akhir pemodelan yang hanya membedakan jenis *pre-trained model* nya saja. Jumlah dataset dan iterasi *training* menggunakan jumlah yang sama. Variasi ini bertujuan untuk dilakukan pengujian yang paling cocok dengan *hardware* yang digunakan. Berikut adalah tampilan isi file dari folder exp53.



**Gambar 6.3-6 Tampilan Isi Folder Hasil *Training***

Dapat dilihat pada Gambar 6.3-6 yaitu isi folder hasil 1 kali *training*. Terdapat beberapa file yang meliputi hasil mAP pemodelan, *confusion matrix*, *loss*, *prediction*, dan yang paling penting adalah file model itu sendiri yang terletak dalam folder weights. File pemodelan ini merupakan file dengan ekstensi .pt yang digunakan program untuk menghasilkan prediksi *machine learning*.

#### 7. Meng-copy file pemodelan ke direktori kerja *running* Program *Face-mask Detection*

Seperti yang dijelaskan sebelumnya, file pemodelan digunakan untuk menghasilkan prediksi *machine learning*. Agar mempermudah akses terhadap file pemodelan, di-copy file pemodelan yang ada pada alamat **Documents/face\_mask/yolov5/runs/train/exp53/weights/last.pt** ke alamat **Documents/face\_mask/SORT**. Kemudian me-rename file tersebut menjadi “final\_training\_model.pt”. Berikut adalah tampilan file pemodelan yang sudah berada di direktori kerja *running* program *Face-mask Detection*.



Gambar 6.3-7 File Pemodelan *Machine Learning* Akhir

#### 6.4 Prosedur Aktivasi Database Server

1. Membuka Terminal Linux
2. Izinkan Akses *Super-User*

Pada proyek yang dikembangkan ini, digunakan MySQL database server sebagai tempat penyimpanan hasil deteksi masker wajah dan digunakan XAMPP sebagai *framework* database. Diperlukan database server untuk menyimpan hasil pembacaan *machine learning* program *face-mask detection*. Untuk dapat menjalankan XAMPP, diperlukan akses sebagai *Super-User* terlebih dahulu. Hal ini bertujuan untuk meningkatkan keamanan database server pada komputer. Berikut adalah kode perintah pada terminal.

```
sudo su
```

Gambar 6.4-1 Perintah Untuk Akses Sebagai *Super-User*

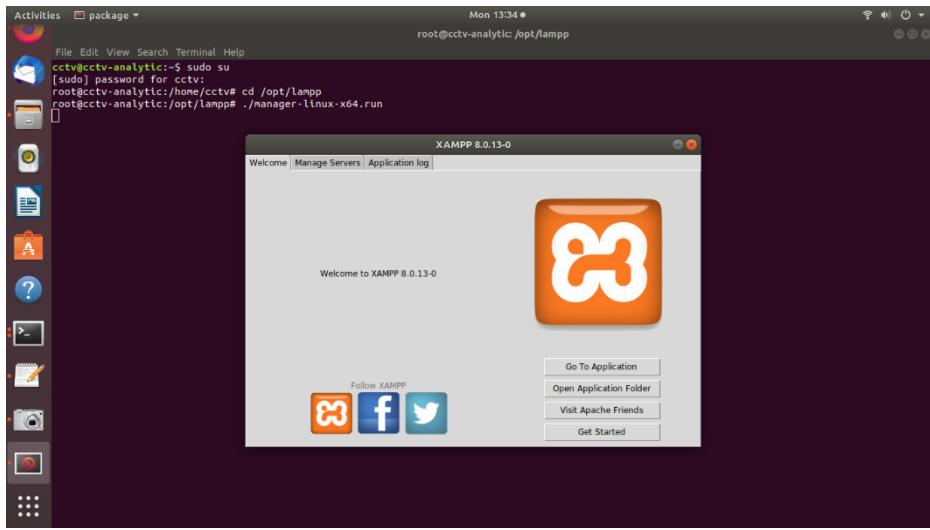
Dapat dilihat pada Gambar 6.4-1 diatas, akan diminta masukkan *password Super-User*. Pada komputer ini, *Password* yang harus di-inputkan adalah “cctv”. Berikut adalah tampilan apabila akses sebagai *Super-User* Diizinkan.

Gambar 6.4-2 Tampilan Akses *Super-User* Diizinkan

3. *Running XAMPP*

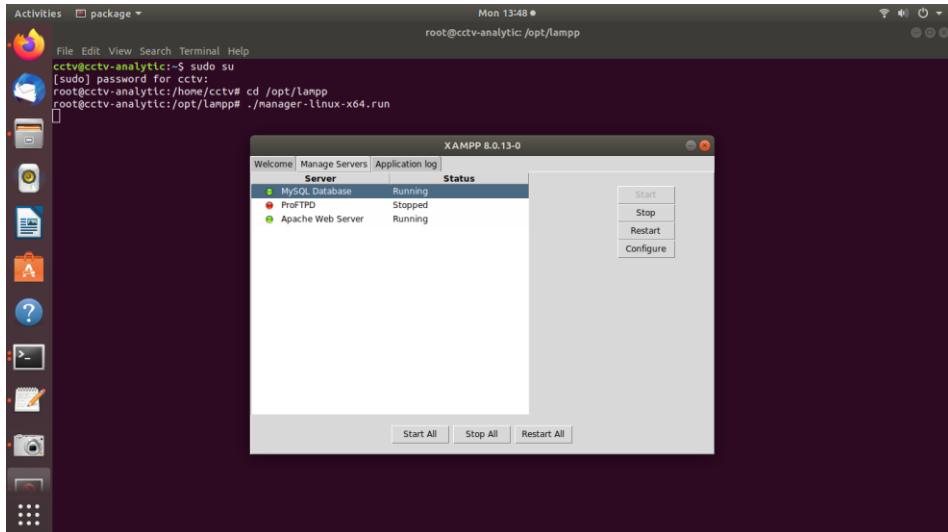
Tahapan selanjutnya adalah menjalankan *software* XAMPP. Berikut adalah kode perintah pada terminal untuk menjalankan XAMPP.

```
cd /opt/lamp
./manager-linux-x64.run
```



**Gambar 6.4-3 Tampilan *Software XAMPP* Berhasil Dijalankan**

Dapat dilihat pada Gambar 6.4-3 yaitu tampilan apabila XAMPP berhasil dijalankan. Setelah XAMPP berhasil dijalankan, selanjutnya adalah melakukan aktivasi Apache Web Server dan MySQL Database. Pada tab “Manage Servers”, Klik pada Apache Web Server kemudian tekan “Start”. Apabila status Apache Web Server sudah tertulis “Running”, ini menandakan bahwa Apache Web Server telah berhasil diaktifkan. Lakukan hal yang sama pada MySQL Database. Berikut adalah tampilan apabila Apache Web Server dan MySQL Database berhasil diaktifkan.



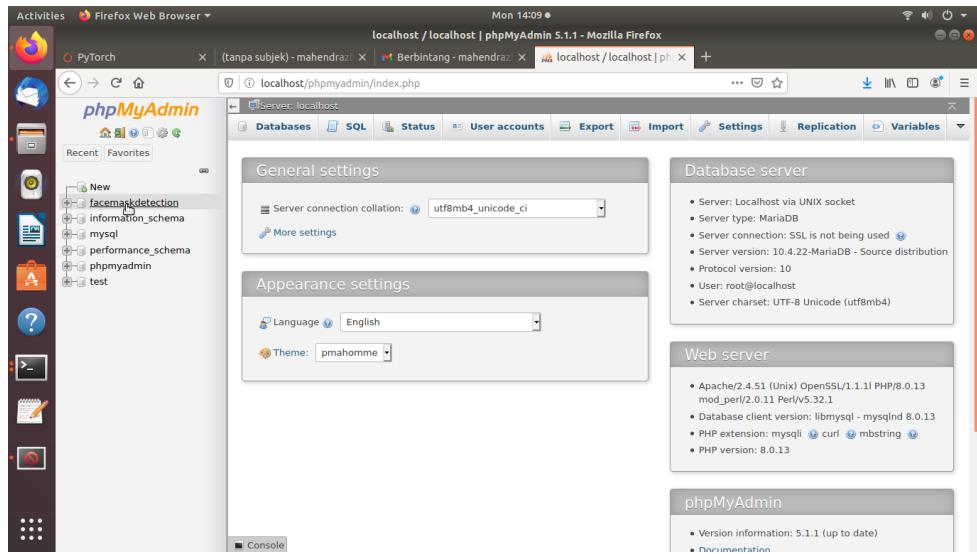
**Gambar 6.4-4 Tampilan MySQL Database dan Apache Web Server Berhasil Diaktifkan**

Dapat dilihat pada Gambar 6.4-4 yaitu tampilan Ketika MySQL Database dan Apache Web Server berhasil diaktifkan.

## 6.5 Prosedur Akses Database Server

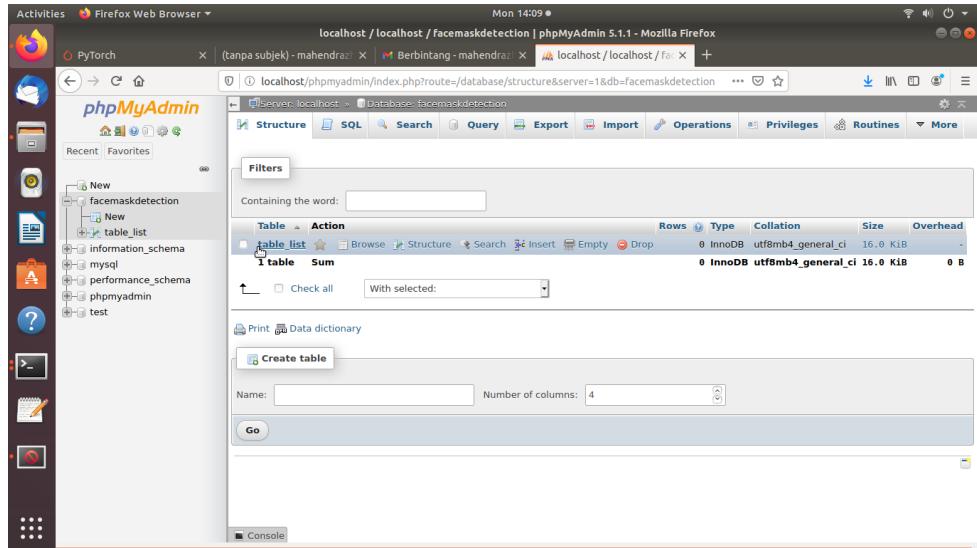
1. Pastikan database server telah aktif (Prosedur 6.4)
2. Membuka *browser*
3. Akses Database

Tahapan selanjutnya adalah melakukan akses terhadap database yang telah diaktifkan. Tuliskan alamat <http://localhost/phpmyadmin>. Berikut adalah tampilan gambar apabila database tersebut berhasil diakses.



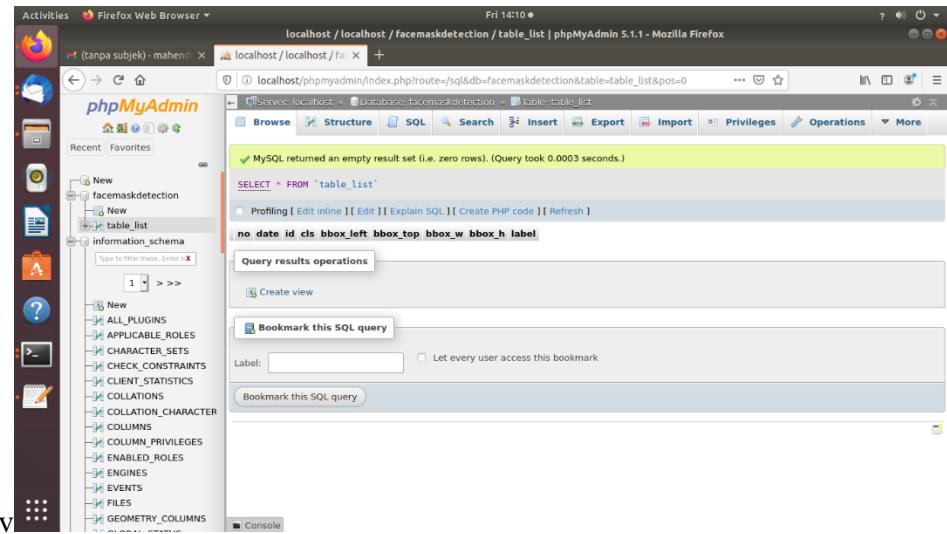
Gambar 6.5-1 Tampilan Database

Untuk melihat data yang ada pada database, klik database yang bernama “facemaskdetection” yang terletak pada sebelah kiri *navigation bar* seperti pada gambar 6.5-1 diatas.



Gambar 6.5-2 Daftar Tabel Pada Database

Dapat dilihat pada gambar 6.5-2 yaitu daftar table yang ada pada database “facemaskdetection”. Terlihat hanya terdapat 1 tabel yang bernama “table\_list”. Tabel ini adalah tabel yang berisikan data-data hasil pemrosesan *Machine Learning Face-mask Detection*. Selanjutnya klik pada tabel bernama “table\_list”.



**Gambar 6.5-3 Tampilan Data Pada Table\_list**

Dapat terlihat pada Gambar 6.5-3 tabel Bernama table\_list untuk menampung seluruh data hasil deteksi telah berhasil dibuat. Terdapat informasi-informasi mengenai jumlah data (no), timestamp (date), id, nomor kelas (cls), koordinat *bounding box* (bbox\_left, bbox\_top, bbox\_w, bbox\_h), dan nama kelas. Pada kondisi ini, data-data pada tabel ini masih kosong.

## 6.6 Prosedur Mengkoneksikan IP Cameran dengan Komputer

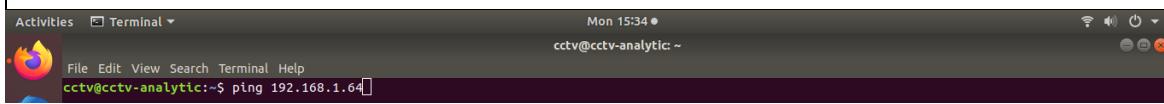


**Gambar 6.6-1 IP Camera**

1. Sambungkan Kabel LAN Komputer ke IP Camera (Gambar 6.6-1 )
2. Sambungkan Kabel Power ke IP Camera (Gambar 6.6-1 )
3. Membuka Terminal Linux
4. Cek Konektivitas IP Kamera

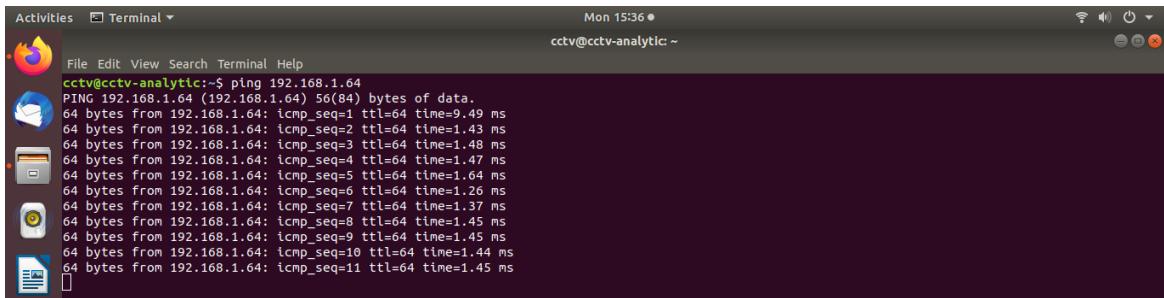
Setelah kabel LAN dan Power tersambung, cek pada computer apakah IP Camera sudah dapat terhubung dengan komputer. Perintah yang dapat digunakan untuk mengecek konektivitas IP Camera dengan computer adalah sebagai berikut.

```
ping 192.168.1.64
```



Gambar 6.6-2 Pengecekan Konektivitas IP Camera

Dapat dilihat pada Gambar 6.6-2 yaitu perintah untuk melakukan pengecekan konektivitas IP Camera. Kode 192.168.1.64 merupakan alamat IP yang dimiliki oleh IP Camera. Berikut adalah tampilan apabila IP Camera sudah berhasil terhubung dengan computer.



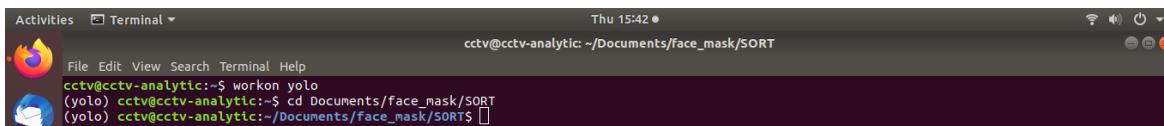
Gambar 6.6-3 Tampilan Apabila IP Camera yang Sudah Terhubung dengan Komputer

## 6.7 Prosedur Running Program Face-mask Detection

1. Pastikan sudah melakukan aktivasi database server (Prosedur 6.4)
2. Membuka Terminal Linux
3. Aktivasi *Virtual Environment* (Prosedur 6.1)
4. Memindahkan Direktori Kerja

Untuk dapat menjalankan program, mula-mula dilakukan terlebih dahulu pemindahan direktori kerja pada terminal. Direktori kerja program *face-mask detection* terletak pada folder **Documents/face\_mask/SORT** seperti yang terlihat pada Gambar 3.1-1. Perintah yang dapat digunakan pada terminal untuk memindahkan direktori kerja program adalah seperti berikut.

```
cd Documents/face_mask/SORT
```



Gambar 6.7-1 Perintah Mengubah Direktori Kerja

Dapat dilihat pada Gambar 6.7-1 diatas, terlihat direktori kerja sudah berubah yaitu pada folder *Documents/face\_mask/SORT*.

5. *Running Program*
  - Menjalankan Program Menggunakan *Webcam*

Untuk menjalankan program dengan menggunakan *webcam* sebagai input, hubungkan terlebih dahulu *webcam* dengan komputer melalui USB port. Setelah *webcam* tersambung, program siap dijalankan dengan menuliskan kode perintah pada terminal seperti berikut.

```
Python3 track.py -source 0 -yolo_weights final_training_model.pt -  
classes 15 16 17
```

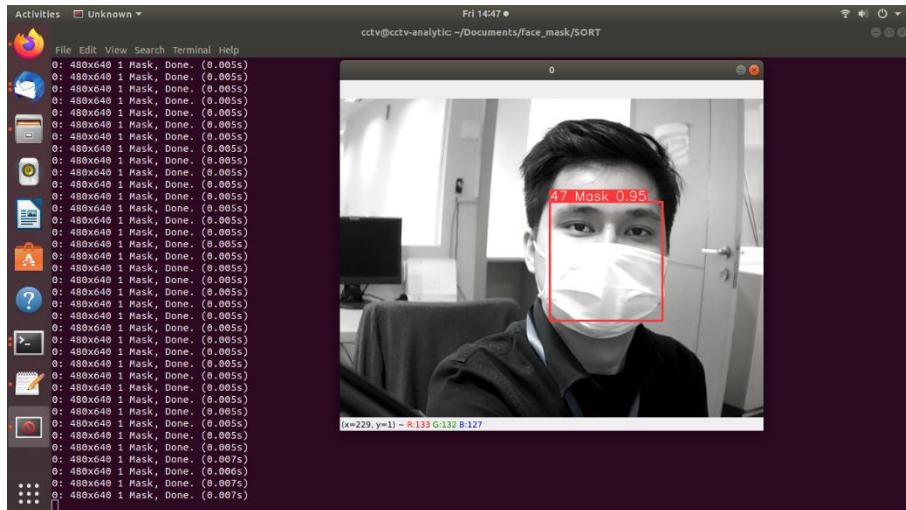
```

Activities Terminal Thu 15:43
cctv@cctv-analytic:~/Documents/face_mask/SORT
File Edit View Search Terminal Help
cctv@cctv-analytic:~$ workon yolo
(yolo) cctv@cctv-analytic:~$ cd Documents/face_mask/SORT
(yolo) cctv@cctv-analytic:~/Documents/face_mask/SORT$ python3 track.py -source 0 -yolo_weights final_training_model.pt -classes 15 16 17

```

**Gambar 6.7-2 Perintah *Running* Program Menggunakan *Webcam***

Kode “0” pada kode program 6.7-2 diatas merupakan bagian yang membuat program berjalan menggunakan *webcam*. Berikut adalah tampilan hasil *running* program menggunakan *webcam*.



**Gambar 6.7-3 Tampilan Hasil *Running* Program Menggunakan *Webcam***

- Menjalankan Program Menggunakan Video

Untuk menjalankan program menggunakan video sebagai input, siapkan terlebih dahulu video yang ingin digunakan. Pada kasus ini, video yang disiapkan diberi nama *video2.mp4* yang disimpan pada alamat **/Documents/face\_mask/video2.mp4** . Berikut adalah kode perintah pada terminal untuk menjalankan program menggunakan video.

```

python3 track.py -source ../video2.mp4 -yolo_weights final_training_model.pt -
classes 15 16 17

```

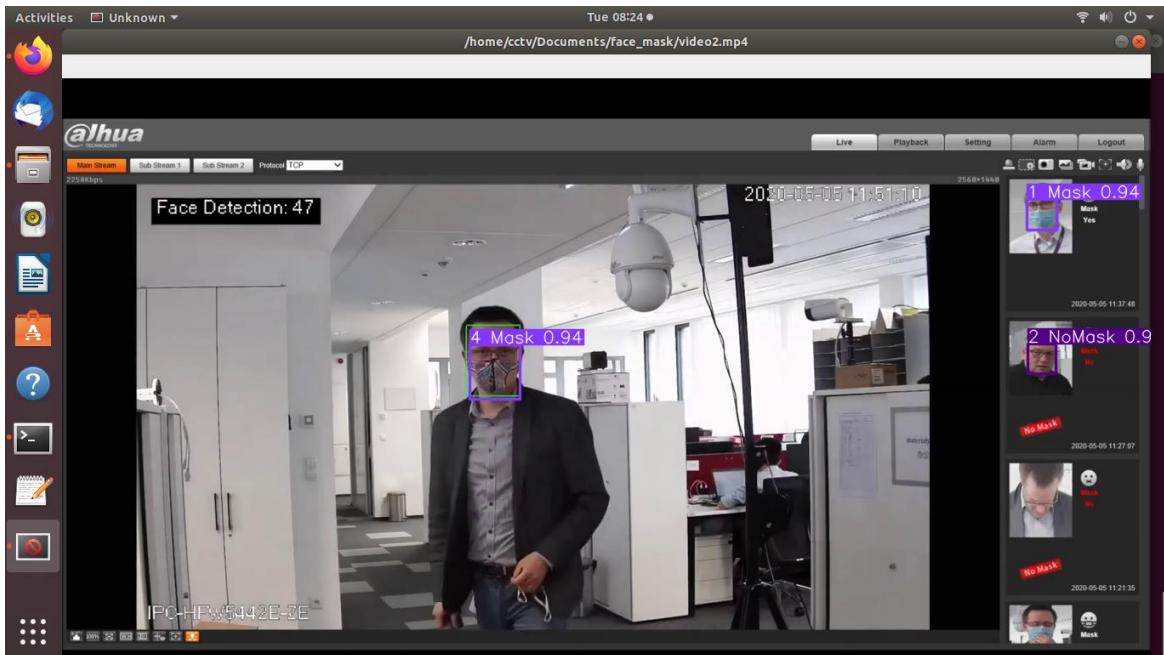
```

Activities Terminal Tue 08:24
cctv@cctv-analytic:~/Documents/face_mask/SORT
File Edit View Search Terminal Help
(yolo) cctv@cctv-analytic:~/Documents/face_mask/SORT$ python3 track.py --source ../video2.mp4 --yolo_weights final_training_model.pt --classes 15 16 17

```

**Gambar 6.7-4 Perintah *Running* Program Menggunakan Video**

Dapat dilihat pada gambar 6.7-4, dilakukan pengubahan pada kode bagian source untuk memberikan input yang berbeda terhadap program yaitu berupa video dengan ekstensi .mp4. File video disimpan di alamat yang lebih tinggi satu tingkat dibandingkan direktori kerja sehingga diperlukan kode “..” untuk mengubah direktori pembacaan file menjadi naik satu tingkat. Berikut adalah tampilan hasil *running* program menggunakan video.



**Gambar 6.7-5 Tampilan Hasil *Running* Program Menggunakan Video**

- Menjalankan Program Menggunakan IP Camera

Pastikan sudah mengkoneksikan IP camera dengan computer (Prosedur 6.6). Apabila sudah, berikut adalah kode perintah pada terminal untuk menjalankan program menggunakan IP Camera.

```
python3 track.py --source rtsp://admin:admin123@192.168.1.64:554/streaming/channels/102 --yolo_weights final_training model.pt --classes 0 1 2
```

**Gambar 6.7-6 Perintah *Running* Program Menggunakan IP Camera**

Kode

**rtsp://admin:admin123@192.168.1.64:554/streaming/channels/102**

pada kode program diatas merupakan perintah yang memberikan instruksi untuk menggunakan IP Camera sebagai input program. Berikut adalah tampilan hasil *running* program menggunakan IP Camera.



Gambar 6.7-7 *Running Program Face-mask Detection Menggunakan IP Camera*

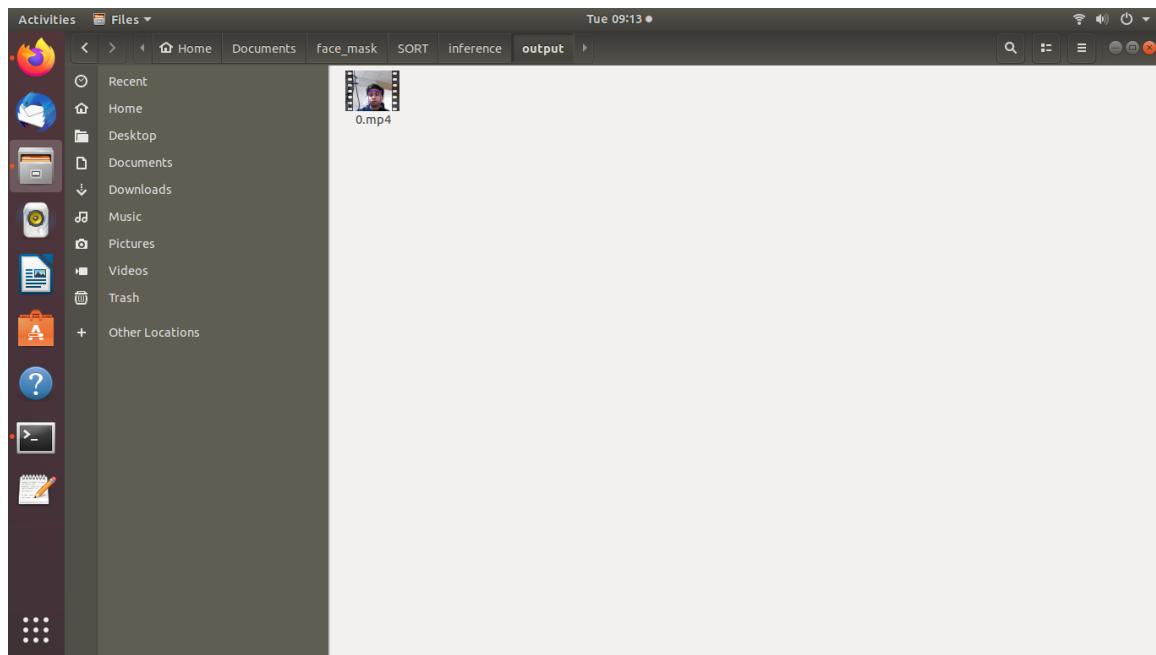
- Menjalankan Program dan Menyimpan Hasil Tangkapan Video

Untuk menyimpan hasil tangkapan video, diperlukan suatu perintah tambahan pada kode perintah terminal. Pada kasus ini, penulis menggunakan *webcam* sebagai input yang kemudian dilakukan pemrosesan *machine learning* dan menghasilkan output berupa video dan disimpan pada suatu folder. Berikut adalah kode perintah pada terminal untuk menjalankan program dan menyimpan hasil tangkapan video.

```
Python3 track.py --source 0 --yolo_weights final_training_model.pt --classes 0 1 2 --save-vid
```

Gambar 6.7-8 Perintah *Running Program dan Menyimpan Hasil Tangkapan Video*

Kode “*save-vid*” pada kode program diatas merupakan perintah yang memberikan instruksi untuk menyimpan video hasil tangkapan kamera. Video yang disimpan sudah berupa video yang telah dilakukan dijalankan program *face-mask detection*. Video tersebut disimpan pada alamat **/Documents/face\_mask/SORT/inference/output/0.mp4**. Berikut adalah tampilan video hasil program *face-mask detection* yang disimpan .



**Gambar 6.7-9 Video Hasil Program Face-mask Detection**

- Menjalankan Program dan Menyimpan Hasil Prediksi *Machine Learning* Pada Database

Pada tahap 2, sudah dilakukan proses aktivasi database server. Pada bagian ini, akan dimanfaatkan database tersebut untuk menyimpan hasil pendekripsi *machine learning* *face-mask detection* dalam bentuk teks. Sama seperti menyimpan video, diperlukan suatu perintah tambahan pada kode perintah terminal. Berikut adalah kode perintah pada terminal untuk menjalankan program dan menyimpan hasil pendekripsi *machine learning* *face-mask detection* dalam bentuk teks.

```
Python3 track.py --source 0 --yolo_weights final_training_model.pt --classes 0 1 2 -save-vid -save-txt
```

A screenshot of a terminal window. The title bar says "Activities Terminal". The command "python3 track.py --source 0 --yolo\_weights final\_training\_model.pt --classes 15 16 17 --save-vid --save-txt" is being typed into the terminal. The path "/Documents/face\_mask/SORT" is visible in the background.

**Gambar 6.7-10 Perintah Running Program dan Menyimpan Hasil Deteksi ke Database**

Kode “*save-txt*” pada kode program diatas merupakan perintah yang memberikan instruksi untuk menyimpan hasil pendekripsi kedalam database dan suatu file eksternal dengan ekstensi .txt. File eksternal tersebut bernama *count\_result.txt* yang tersimpan pada alamat **/Documents/face\_mask/SORT/count\_result.txt**. Berikut adalah tampilan database hasil program *face-mask detection* dan file eksternal *count\_result.txt*.

no	date	id	cls	bbox_left	bbox_top	bbox_w	bbox_h	label
1	2022-01-28 14:36:24	1	0	250	295	149	179	Mask
2	2022-01-28 14:36:28	1	2	247	283	148	191	NoMask
3	2022-01-28 14:36:31	3	2	247	295	153	184	NoMask
4	2022-01-28 14:36:33	4	2	251	414	199	65	NoMask
5	2022-01-28 14:36:39	5	0	262	372	180	107	Mask
6	2022-01-28 14:36:43	6	0	277	350	170	129	Mask
7	2022-01-28 14:36:50	7	0	258	369	176	110	Mask
8	2022-01-28 14:36:54	7	2	254	285	161	194	NoMask
9	2022-01-28 14:37:02	10	0	255	400	184	79	Mask
10	2022-01-28 14:37:02	11	2	260	397	189	82	NoMask
11	2022-01-28 14:37:11	12	0	270	370	181	109	Mask
12	2022-01-28 14:37:24	12	2	257	304	162	175	NoMask
13	2022-01-28 14:37:28	13	2	245	425	257	54	NoMask
14	2022-01-28 14:37:37	14	0	270	353	169	126	Mask
15	2022-01-28 14:37:42	15	0	232	385	182	94	Mask
16	2022-01-28 14:37:47	16	2	241	375	191	104	NoMask
17	2022-01-28 14:37:56	17	0	258	385	193	94	Mask
18	2022-01-28 14:38:06	20	0	263	320	240	159	Mask
19	2022-01-28 14:38:21	21	2	261	299	161	180	NoMask
20	2022-01-28 14:38:24	23	2	244	393	186	86	NoMask
21	2022-01-28 14:38:30	24	0	263	405	192	74	Mask

Gambar 6.7-11 Tampilan Database Program

Dapat dilihat pada gambar 6.7-11 yaitu tampilan database setelah menjalankan program *face-mask detection*. Terdapat informasi-informasi mengenai jumlah data (no), *timestamp* (date), id, nomor kelas (cls), koordinat *bounding box* (bbox\_left, bbox\_top, bbox\_w, bbox\_h), dan nama kelas. Program penyimpanan sudah dirancang sehingga data yang masuk ke database tidak mengalami pengulangan. Terlihat pada gambar tidak terdapat data yang sama untuk id dan label yang sama. Penyimpanan juga sudah dirancang untuk mengantisipasi perangkat yang ter-shutdown yang mengakibatkan program harus di-restart. Ini dilakukan dengan mencari id tertinggi dari data terakhir pada database kemudian melanjutkan *counting* dengan acuan id tertinggi tersebut. Sehingga tetap tidak terdapat pengulangan data untuk id dan label yang sama.

```

Fri 14:42 ●
count_result.txt
track.py
count_result.txt
count_result.txt

1 0 250 295 149 179 Mask
2 1 247 283 148 191 NoMask
3 3 2 247 295 153 184 NoMask
4 4 2 251 414 199 65 NoMask
5 5 0 262 372 180 167 Mask
6 6 0 277 350 170 129 Mask
7 7 0 258 369 176 110 Mask
8 7 2 254 285 161 194 NoMask
9 10 0 255 400 184 79 Mask
10 11 2 260 397 189 82 NoMask
11 12 0 270 370 181 109 Mask
12 12 2 257 304 162 175 NoMask
13 13 2 245 425 257 54 NoMask
14 14 0 270 355 169 126 Mask
15 15 0 232 385 182 94 Mask
16 16 2 241 375 191 104 NoMask
17 17 0 258 385 193 94 Mask
18 18 0 263 320 240 159 Mask
19 19 2 261 299 161 180 NoMask
20 20 2 244 393 186 86 NoMask
21 21 0 263 405 192 74 Mask
22 22 0 248 405 157 74 NoMask
23 23 2 256 276 71 201 NoMask
24 24 2 469 426 170 53 NoMask
25 25 0 432 282 206 197 Mask
26 26 0 426 291 162 188 Mask
27 27 0 462 382 126 97 Mask
28 28 1 553 355 86 124 NoMask
29 29 0 499 435 134 49 Mask
30 30 2 604 385 35 94 NoMask
31 31 2 565 360 74 116 NoMask
32 32 0 593 414 46 65 Mask
33 33 0 547 372 92 107 Mask
34 34 2 558 367 81 112 NoMask
35 35 0 432 321 140 158 Mask
36 36 0 407 306 17 92 Mask

```

Gambar 6.7-12 File count\_result.txt

File count\_result.txt memiliki informasi data yang sama persis dengan data yang tersimpan pada database melainkan output nya merupakan file dengan ekstensi .txt.

## **6.8 Spesifikasi Sistem Berdasarkan Pengujian**

**Table 6.8-1 Spesifikasi Sistem**

Parameter	Nilai
<b>Height of CCTV</b>	<b>2.3 meter</b>
<b>Object Distance</b>	<b>0.5 – 5 m</b>
<b>Classes</b>	<b>2 Classes (Mask and NoMask)</b>
<b>Accuracy (%)</b>	<b>85.18</b>
<b>Color</b>	<b>Grayscale</b>
<b>FPS</b>	<b>25</b>
<b>Amount of Video Stream</b>	<b>Up to 5 Streams</b>
<b>Database</b>	<b>MySQL</b>
<b>Angle of Face</b>	<b><math>\pm 60^\circ</math> with respect to vertical axis of the camera</b>
<b>Total Object Detected in 1 Frame</b>	<b>Not Limited</b>