PEMANFAATAN JETSON NANO NVIDIA UNTUK MENDETEKSI PENGGUNAAN MASKER SECARA REAL-TIME MENGGUNAKAN OPENCV PYTHON

LAPORAN TUGAS AKHIR

Laporan ini Disusun Guna Memenuhi Salah Satu Syarat Memperoleh Gelar Sarjana Strata (S1) pada Program Studi Teknik Informatika

Fakultas Teknologi Industri

Universitas Islam Sultan Agung Semarang

Disusun Oleh:

Nama : Sofianisa Fitriyati Prisunia

Nim 32602000123

Program Studi : Teknik Informatika

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM SULTAN AGUNG

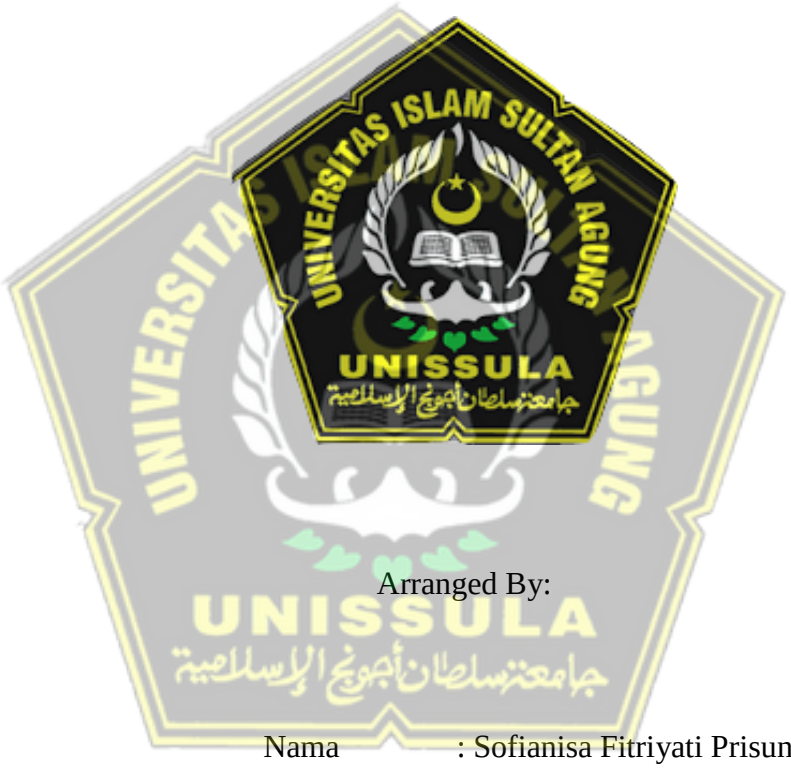
SEMARANG

2023

FINAL PROJECT

UTILIZATION OF JETSON NANO NVIDA TO DETECT MASK IN REAL-TIME USING OPENCV PYTHON

Proposed to complete the requirement to obtain a bachelor's degree (S1) At Informatics Engineering Departement of Industrial Technology Faculty Sultan Agung Islamic University



Arranged By:

Nama : Sofianisa Fitriyati Prisunia

Nim 32602000123

MAJORING OF INFORMATICS ENGINEERING

INDUSTRIAL TECHNOLOGY FACULTY

SULTAN AGUNG ISLAMIC UNIVERSITY

SEMARANG

2023

LEMBAR PENGESAHAN PEMBIMBING

Laporan Tugas Akhir dengan judul "Pemanfaatan Jetson Nano Nvidia untuk Mendeteksi Penggunaan Masker Secara Real-Time Menggunakan OpenCV Python" ini disusun oleh :

Nama : Sofianisa Fitriyati Prisunia

NIM : 32602000123

Program Studi : Teknik Informatika

Telah disahkan oleh dosen pembimbing pada :

Hari : .....

Tanggal : .....

Mengesahkan,

Pembimbing I	Pembimbing II
Andi Riansyah, ST, M.Kom	Ir. Sri Mulyono, M.Eng
NIDN. 0609108802	NIDN. 0626066601

Mengetahui,
Ketua Program Studi Teknik Informatika
Fakultas Teknologi Industri
Universitas Islam Sultan Agung

Ir. Sri Mulyono, M.Eng
NIDN. 0626066601

LEMBAR PENGESAHAN PENGUJI

Laporan tugas akhir dengan judul "Pemanfaatan Jetson Nano Nvidia untuk Mendeteksi Penggunaan Masker Secara Real-Time Menggunakan OpenCV Python" ini telah dipertahankan di depan dosen penguji Tugas Akhir pada :

Hari : .....

Tanggal : .....

TIM PENGUJI

Penguji I	Penguji II
Sam Farisa Chaerul Haviana, ST, M.Kom	Asih Widi Harini, Ssi. MT
NIDN. 0628028602	NIDN. 0617087002

SURAT PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan dibawah ini :

Nama : Sofianisa Fitriyati Prisunia

NIM : 32602000123

Judul Tugas Akhir : Pemanfaatan Jetson Nano Nvidia untuk Mendeteksi Penggunaan Masker Secara Real-Time Menggunakan OpenCV Python

Dengan bahwa ini saya menyatakan bahwa judul dan isi Tugas Akhir yang saya buat dalam rangka menyelesaikan Pendidikan Strata Satu (S1) Teknik Informatika tersebut adalah asli dan belum pernah diangkat, ditulis ataupun dipublikasikan oleh siapapun baik keseluruhan maupun sebagian, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka, dan apbila di kemudian hari ternyata terbukti bahwa judul Tugas Akhir tersebut pernah diangkat, ditulis ataupun dipublikasikan, maka saya bersedia dikenakan sanksi akademis. Demikian surat pernyataan ini saya buat dengan sadar dan penuh tanggung jawab.

Semarang, 01 Maret 2023

Yang Menyatakan,

Sofianisa Fitriyati Prisunia

PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH

Saya yang bertanda tangan dibawah ini :

Nama : Sofianisa Fitriyati Prisunia

NIM : 32602000123

Program Studi : Teknik Informatika

Fakultas : Teknologi industri

Alamat Asal : Jl. PDK RT.001 / RW.004, Lebak Bulus Cilandak, Jakarta Selatan. DKI Jakarta

Dengan ini menyatakan Karya Ilmiah berupa Tugas akhir dengan Judul : Pemanfaatan Jetson Nano Nvidia untuk Mendeteksi Penggunaan Masker Secara Real-Time Menggunakan OpenCV Python Menyetujui menjadi hak milik Universitas Islam Sultan Agung serta memberikan Hak bebas Royalti Non-Eksklusif untuk disimpan, dialihmediakan, dikelola dan pangkalan data dan dipublikasikan diinternet dan media lain untuk kepentingan akademis selama tetap menyantumkan nama penulis sebagai pemilik hak cipta. Pernyataan ini saya buat dengan sungguh-sungguh. Apabila dikemudian hari terbukti ada pelanggaran Hak Cipta/Plagiarisme dalam karya ilmiah ini, maka segala bentuk tuntutan hukum yang timbul akan saya tanggung secara pribadi tanpa melibatkan Universitas Islam Sultan agung.

Semarang, 01 Maret 2023

Yang menyatakan,

Sofianisa Fitriyati Prisunia

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Virus Corona (COVID-19) telah menyebar ke seluruh dunia menyebabkan lebih dari 178 juta kasus yang dikonfirmasi dan 3,9 juta kematian di seluruh dunia menurut data dari BBC News Indonesia pada Juni 2021. Oleh karena itu perlunya perhatian masyarakat dalam menaati protokol kesehatan yaitu dengan menggunakan masker. Mendeteksi penggunaan masker adalah sistem kecerdasan buatan secaraotomatis untuk mengetahui apakah seseorang memakai masker, tidak memakai masker, masker yang digunakan salah. Pengolahan citra OpenCV digunakan sebagai deteksi obyek masker. Sistem ini menggunakan algoritma You Only Look Once (YOLOv4) serta menggunakan *Darknet* sebagai sistem klasifikasi objek yang terdeteksi dan *data training*. sistem dapat mengklasifikasi wajah with mask, without mask, dan worn mask incorrectly serta mendapatkan nilai akurasi dari penggunaan metode ini, sistem ini berbasis Jetson Nano Nvidia.

Kata Kunci : *Face Mask Detection, Jetson Nano Nvidia, , Image Processing.*

ABSTRACT

The Corona Virus (COVID-19) has spread throughout the world causing more than 178 million confirmed cases and 3.9 million deaths worldwide according to data from BBC News Indonesia in June 2021. Therefore the need for public attention in adhering to health protocols, namely by using a mask. Detecting the use of a mask is an artificial intelligence system that automatically detects whether someone is wearing a mask, not wearing a mask, or the wrong mask is being used. OpenCV image processing is used as mask object detection. This system uses the You Only Look Once (YOLOv4) algorithm and uses Darknet as a detected object classification system and training data. the system can classify faces with mask, without mask, and worn mask incorrectly and obtain accuracy values from using this method, this system is based on Jetson Nano Nvidia.

Keywords : *Face Mask Detection, Jetson Nano Nvidia, , Image Processing.*

BAB I
PENDAHULUAN

1.1 Latar Belakang

Awal mula dimana terjadi nya virus SARS-CoV-2 pertama kali terdeteksi di China yaitu Wuhan pada tahun 2019. Menurut data dari BBC News Indonesia, Pada bulan Juni 2021 virus tersebut telah menyebar ke seluruh dunia dan telah menyebabkan lebih dari 178 juta kasus yang dikonfirmasi dan 3,9 juta kematian di seluruh dunia (Britt Yip and Valeria Perasso, 2021). Penyakit ini disebabkan oleh coronavirus sindrom pernapasan akut berat 2 (SARS-CoV-2). Kasus positif Covid-19 di Indonesia pertama kali terdeteksi pada tanggal 2 Maret 2020. Upaya pemerintah dalam mencegah serta penguatan sistem kesehatan dalam pengendalian Covid -19 dengan menjalankan protokol kesehatan salah satunya, yakni memakai masker (Indra Jaya, 2021).

Saat ini, pendeteksian penggunaan masker di tempat umum masih dilakukan secara manual melalui mengamatan baik satuan petugas keamaan maupun petugas yang berwenang dan kurang efektif karena masih memungkinkan banyak terjadi kesalahan (human error). Untuk mengurangi kesalahan dalam pendeteksian masker maka perlu sistem kecerdasan buatan secara otomatis dengan pengolahan citra sebagai sistem deteksi masker yaitu *Face mask detection.* Sistem ini menggunakan You Only Look Once (YOLOv4) sebagai sistem klasifikasi objek yang terdeteksi.

Berdasarkan penjelasan tersebut, penulis melakukan observasi melalui penelitian sebelumnya untuk mengembangkan sistem yang dapat mendeteksi wajah bermasker. Pada sistem ini memanfaatkan Jetson Nano Nvidia yang tersambung dengan kamera. Kamera berfungsi untuk menangkap gambar objek serta mendeteksi kemudian diolah dengan computer vision menjadi pengolahan citra, dan menghasilkan 3 kelas data *real-time* menggunakan masker (mask), masker yang dipakai salah (bad mask) dan tidak menggunakan masker (no mask).

1.2 Rumusan Masalah

Dari pernyataan latar belakang permasalahan yang telah diuraikan diatas, dapat diidentifikasi beberapa permasalahan dalam penelitian ini, yaitu:

1. Bagaimana membuat sistem serta memanfaatkan Jetson Nano Nvidia untuk mendeteksi penggunaan masker secara *real-time*?
2. Bagaimana mengimplementasikan *Face Mask Detection* menggunakan *YOLO*v4 sebagai sistem pendeteksi menggunakan masker?
3. Bagaimana hasil nilai akurasi *Face Mask Detection* menggunakan *YOLO*v4 sebagai sistem pendeteksi menggunakan masker?

1.3 Pembatasan Masalah

Sesuai dengan rumusan diatas, berikut ini merupakan batasan masalah yaitu :

1. Pendeteksian menggunakan masker memiliki tiga hasil klasifikasi *Mask, No mask,* dan *Bad Mask*.
2. *Image processing* yang digunakan berupa dataset gambar orang menggunakan masker dan tanpa masker.
3. Metode yang digunakan Yolov4 serta pengolahan *image processing* citra digital.
4. Luaran dari sistem ini adalah mendeteksi wajah dengan sensor kamera untuk data diolah kedalam sistem menghasilkan informasi yaitu *Mask, No mask, Bad Mask*.

1.4 Tujuan

Tujuan tugas akhir dari penelitian ini yaitu menghasilkan sistem yang dapat mendeteksi penggunaan masker wajah secara real-time dengan metode You Only Look Once (YOLOv4), sistem dapat mengklasifikasi wajah *Mask, No mask, Bad Mask*. serta mendapatkan nilai akurasi dari penggunaan metode ini, sistem ini berbasis Jetson Nano Nvidia.

1.5 Manfaat

Adapun maanfaat tugas akhir ini adalah dengan pemanfaatan media Jetson Nano Nvidia sebagai perangkat komputer kecil yang mudah digunakan di mana saja untuk mendeteksi penggunaan masker wajah secara real time.

Adanya sistem kecerdasan buatan secara otomatis ini untuk mengurangi terjadi nya human error dan efektif dalam menunjang program pemerintah yaitu protokol kesehatan penanganan covid-19 dalam menggunakan masker. Sistem dapat mengklasifikasi dengan 3 kelas yaitu *mask*, *no mask* dan *bad mask* serta menampilkan nilai akurasi nya.

1.6 Sistematika Penulisan

Sistematika penulisan yang digunakan penulis dalam menyusun laporan tugas akhir ini antara lain sebagai berikut:

BAB 1: PENDAHULUAN

Pada Bab 1, penulis mengutarakan latar belakang dari penelitian yang diambil, membuat rumusan masalah, batasan masalah, serta tujuan dan manfaat yang diperoleh, dan sistematika penulisan.

BAB 2: TINJAUAN PUSTAKA DAN DASAR TEORI

Pada Bab 2, penulis akan memaparkan dasar-dasar teori yang digunakan, serta rujukan dari penelitian terdahulu, untuk membantu penulis dalam memahami bagaimana teori yang berhubungan dengan perangkat komputer kecil *Jetson Nano Nvidia*, dan metode *YOLOv4*.

BAB 3: METODE PENELITIAN

Dalam Bab 3, menjelaskan proses tahapan-tahapan penelitian dalam mengembangkan sistem, dimulai dari mendapatkan data hingga proses klasifikasi data yang ada.

BAB 4: HASIL DAN ANALISIS PENELITIAN

Pada Bab 4, penulis memaparkan apa saja hasil dari penelitian ini, implementasi sistem pada komputer kecil menggunakan Jetson Nano Nvidia, dilanjutkan dengan hasil akhir *system*, klasifikasi data, dan akurasi dari *system*.

BAB 5: KESIMPULAN DAN SARAN

Pada Bab 5, penulis menjelaskan kesimpulan keseluruhan dari hasil proses penelitian mulai dari awal hingga akhir.

BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Tinjauan pustaka ini akan membahas tentang penelitian terdahulu yang berkaitan dengan penelitian yang akan dilakukan oleh penulis. Literatur ini nantinya akan dijadikan acuan (referensi) dalam melakukan penelitian ini.

Pada refrensi jurnal yang berjudul "Face Mask Detection using Machine Learning" yang ditulis oleh Pornphat Sroison (2022) penelitian ini membahas mengenai *computer vision*, membangun sistem yang dapat mengenali serta menghitung objek yang ditangkap oleh kamera berdasarkan warna. Kesimpulan hasil dari penelitian menunjukkan bahwa warna objek dapat digunakan untuk mendeteksi dan menghitung warna objek yang ditentukan oleh kamera (Nalinipriya dkk., 2022).

Tri Septiana Nadia Puspita Putri (2021) dari Universitas Muhammadiyah Malang telah membuat penelitian dengan berjudul "Face Mask Detection Covid-19 Using Convolutional Neural Network (CNN)". Perancangan dan implementasi program deteksi menggunakan masker, menggunakan bahasa pemrograman Python dan metode CNN (*Convolutional Neural Network*). Hasil pengujian sistem dengan pengumpulan data sebanyak 36x untuk pengujian deteksi masker wajah dengan *video streaming* mendapatkan nilai akurasi sebesa 94,44% (Septiana dkk., 2020).

Dalam penelitan yang dilakukan oleh Raden B. Hadiprakoso dan Nurul Qomariasih, mereka membuat jurnal berjudul "Deteksi masker wajah menggunakan deep transfer learning dan augmentasi gambar" menggunakan bahasa pemrograman Python, Jupyter Notebook IDE, dan *library* Mathlib plot, TensorFlow, dan Keras. Pengolahan data menggunakan dataset CelebA untuk wajah tidak memakai masker dan dataset untuk maskedface net yaitu wajah yang bermasker, proses penambahan citra dan *deep transfer learning*, hasil akurasinya adalah 98,3% sedangkan skor F1 pada *dataset* validasi adalah 98,7%. Implementasi menggunakan *mobile* dan Raspberry pi untuk mendeteksi video secara real-time (Budiarto Hadiprakoso & Qomariasih,

2022).

Kemudian pada penelitian yang dilakukan oleh Galang Aprilian Anarki dan kawan-kawan, mereka melakukan penelitian berjudul “Penerapan Metode Haar Cascade Pada Aplikasi Deteksi Masker” (Aprilian Anarki dkk., 2021). Penelitian ini menggunakan Python sebagai bahasa pemrogramannya serta menggunakan *library* OpenCV dan input gambar citra berupa gambar dan video. Hasil penelitian menunjukkan bahwa aplikasi dapat dengan benar mendeteksi masker dari citra yang berasal dari video *webcam* dari internal dan eksternal atau berupa gambar, dan didapatkan nilai akurasi tertinggi secara keseluruhan sebesar 88,7% dan terendah sebesa 44,9%. Serta memiliki fitur peringatan (alert) berupa *audio* dan memotret.

2.2 Dasar Teori

2.2.1 Jetson Nano Nvidia

Jetson Nano Nvidia adalah sebuah komputer kecil yang memiliki Prosesor ARM Cortex-A57 CPU Quadcore 64 Bit dengan 128 GPU (CUDA) core yang mempunyai kemampuan komputasi hingga 472 Gflops, yang mampu mengimplementasikan klasifikasi deteksi gambar, objek, segmentasi, dan lainnya. kompatibel untuk digunakan dengan kartu microSD maupun SSD sebagai penyimpanan nya, keunggulan jetson terdapat pada salah satunya yaitu JetPack sangat *compatible* dengan *platform* kecerdasan buatan (artificial intelligence).

Gambar 2. 1 *Jetson Nano Nvidia*

Pada *Jetson Nano Nvidia* terdapat beberapa spesifikasi ditujukan pada table 2.1 sebagai berikut ini :

Tabel 2. 1 Spesifikasi *Jetson Nano Nvidia*

Sumber Daya Utama	5V 2A (via micro USB) / 5V 4A (via jack DC)
RAM	2 GB 64-bit LPDDR4 25,6 GB
CPU	Quad-core ARM A57 @ 1,43 GHz
GPU	128-core Maxwell
Video Encoder	4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264/H.265)
Video Dekoder	4K @ 60 | 2x 4K @ 30 | 8x 1080p
Penyimpanan	microSD dan SSD
Konektifitas	Gigabit Ethernet, M.2 Key E
Tampilan Output	HDMI 2.0 dan eDP 1.4
Slot Kamera	1x MIPI CSI-2 DPHY lanes
USB	4x USB 3.0, USB 2.0 Micro -B
Pin I/O	GPIO, I2C, I2S, SPI, UART

2.2.2 *Convolutional Neural Network* (CNN)

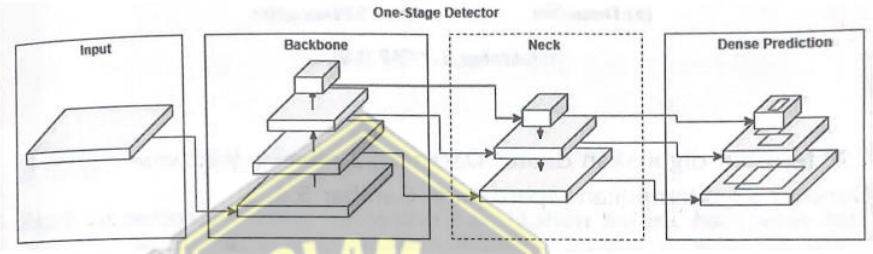
Convolutional Neural Network merupakan arsitektur dasar yang digunakan pada bidang *deep learning* dapat digunakan untuk melakukan klasifikasi, *object detection*, dan *image segmentation* pada citra. Nilai pixel citra pada *CNN* direpresentasikan dalam matriks berdimensi sama dengan resolusi citra, dikonvolusikan dengan matriks filter berukuran *fxf*. Nilai dari filter inilah yang akan menentukan hasil ekstraksi fitur setelah proses konvolusi. *Convolutional Neural Network* memiliki tiga jenis *layer* utama yaitu *convolution layer, pooling layer, dan fully connected layer.*

2.2.3 YOLOv4 (You Only Look Once)

Menurut Nur Arkhamia Batubara dan Rolly Maulana Awanggas (2020) buku tutorial *object detection plate number with convolution neural network* (CNN), *YOLO* merupakan algoritma yang dirancang untuk mendeteksi objek secara *real-time*. Sistem deteksi yang digunakan adalah dengan menggunakan *localizer* untuk mendeteksi atau *repurpose classifier*. Model diterapkan pada sebuah citra di beberapa lokasi dan skala. Citra dengan nilai tertinggi dianggap sebagai pendeteksian .

Arsitektur *single-stage* YOLOv4 dapat menghasilkan waktu inferensi yang cepat mencapai hasil yang canggih (43,5% AP) untuk deteksi objek *real*

time dan mampu berjalan dengan kecepatan mencapai 55 FPS pada GPU V100 jauh lebih unggul dibanding YOLOv3. Pada penelitian tugas akhir ini YOLOv4 memilikitahapan-tahapan dalam mengklasifikasi objek deteksi. YOLOv4 menggunakan *library darknet* untuk memprediksi lokasi dan kelas objek. Pada penelitian ini nantinya terdapat tiga kelas yaitu *mask*, *no mask*, dan *bad mask.*



Gambar 2. 2 Layer cara kerja *YOLOv4*

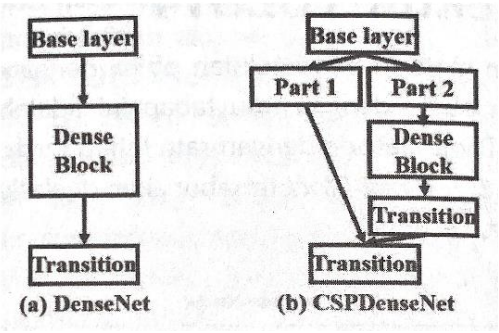
Layer block-block pada gambar diatas meliputi penjelasan sebagai berikut:

1. *Input*

   Citra akan di-*resize* sesuai resolusi lapisan input. Resolusi tersebut dapat diubah dengan syarat dapat dibagi dengan 32. Secara *default*, citra input yang diterima adalah citra berwarna atau citra dengan 3 kanal.

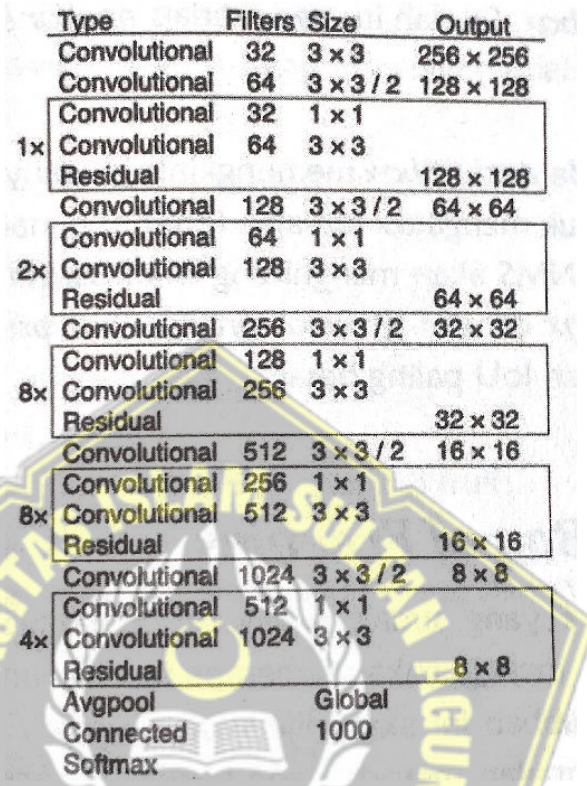
2. *Backbone*

   Arsitektur yang melakukan ekstrasksi fitur. *Backbone* yang dipilih yaitu CSPRNext50, CSPDark-net53, dan Efficient-B3. Namun setelah melakukan beberapa percobaan CSPDarknet-53 (cross stage partial connections) menjadi model yang paling optimal. CSP membagi *input feature maps* ke dua bagian.



Gambar 2. 3 CSP

CSP digunakan di atas Darknet-53 sebagai backbone YOLOv4. Darknet-53 ditampilkan seperti pada gambar 2.4.



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

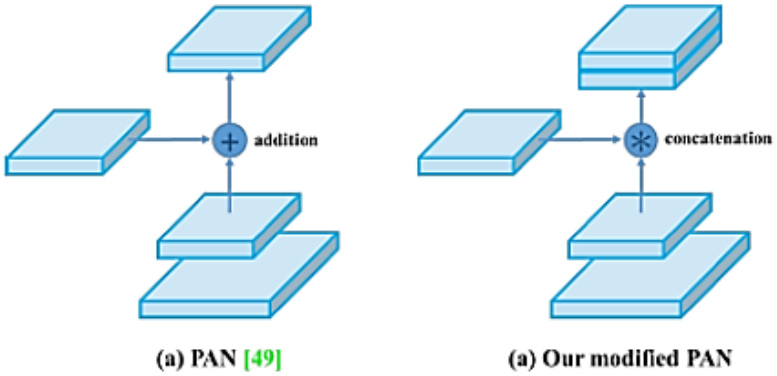
Gambar 2. 4 Darknet-53

3. *Neck*

Menambahkan lapisan antara *backbone* dan *dense prediction*. YOLOv4 menggunakan *Path Aggregation Network* (PAN) dan *Spatial Pyramid Pooling* (SPP). Dapat mendeteksi objek berbagai ukuran.

a. PAN (Path Aggregation Network)

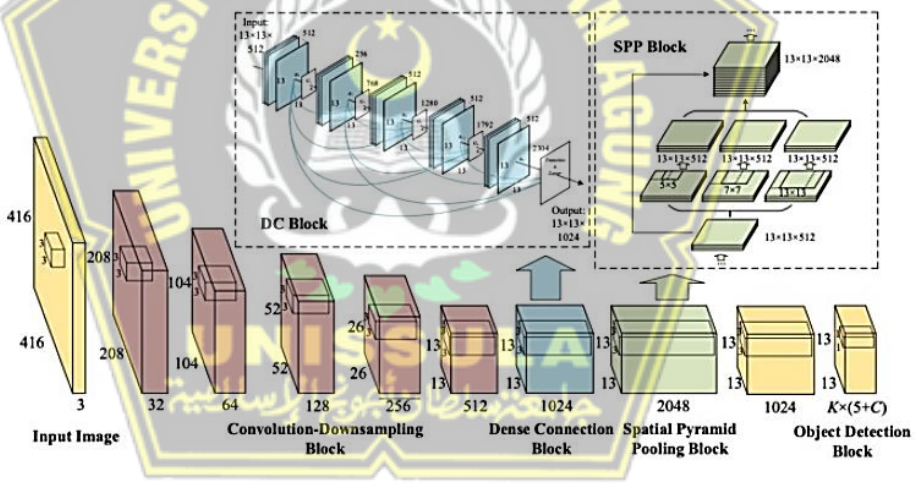
Kegunaan PAN adalah untuk meningkatkan proses segmentasi *instace* dengan menjaga informasi spasial, membantu lokalisasi *pixel* yang tepat untuk prediksi mask.



Gambar 2. 5 PAN Layer

b. SPP (Spatial Pyramid Pooling)

Dengan adanya *block* ini untuk meningkatkan bidang reseptif dan memisahkan fitur konteks signifikan dan terhubung ke lapisan terakhir dari lapisan konvolusional CSPDarknet yang saling berhubungan erat.



Gambar 2. 6 Neck Spatial Pyramid Pooling

Berikut penjelasan performa dari SPP (Spatial Pyramid Pooling) terdapat pada gambar 2. 7 yaitu :

Layers	Parameters		Output	Layers	Parameters		Output
	Filters	Size / Stride			Filters	Size / Stride	
Conv 1	32	3×3/ 1	416×416×32	**DC Block**	**1024**	**3×3 / 1 ×4**	**13×13×2304**
Maxpool 1		2×2 / 2	208×208×32	**Conv 14-21**	**256 or 512**	**1×1 / 1**	
Conv 2	64	3×3 / 1	208×208×64	**Conv 22**	**1024**	**3×3 / 1**	**13×13×1024**
Maxpool 2		2×2 / 2	104×104×64	**Conv 23**	**512**	**1×1 / 1**	**13×13×512**
Conv 3	128	3×3 / 1	104×104×128	**SPP Block Maxpool 6-8**		**5×5 / 1**	**13×13×2048**
Conv 4	64	1×1 / 1	104×104×64			**7×7 / 1 Concat**	
Conv 5	128	3×3 / 1	104×104×128			**13×13 / 1**	
Maxpool 3		2×2 / 2	52×52×128	**Conv 26**	**512**	**1×1 / 1**	**13×13×512**
Conv 6	256	3×3 / 1	52×52×256	Conv 27	1024	3×3 / 1	13×13×1024
Conv 7	128	1×1 / 1	52×52×128	Reorg Conv13		/ 2	13×13×256
Conv 8	256	3×3 / 1	52×52×256	Concat -1, -2			13×13×1280
Maxpool 4		2×2 / 2	26×26×256	Conv 30	1024	3×3 / 1	13×13×1024
Conv 9-12	512	3×3 / 1 1×1 / 1 ×2	Conv 31	Conv31	*K*5+C*	1×1 / 1	13×13×(*K*5+C*)
Conv 13	512	3×3 / 1	26×26×512	Detection			
Maxpool 5		2×2 / 2	13×13×512				

Gambar 2. 7 Performa SPP

4. *Dense Prediction / Head*

   Tahap ditentukan *bounding box* dan klasifikasi terhadap apa yang ada di dalam *bounding box*. YOLOv4 membagi citra input menjadi *grid cell* pada setiap *cell* terdapat *anchor box*. Setelah itu pada setiap *anchor box* dilakukan prediksi.

   Fungsi utama yaitu menemukan kotak pembatas dan melakukan klasifikasi. Koordinat kotak pembatas (x,y, tinggi, dan lebar) serta skor terdeteksi.

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_h e^{t_h}$$

Gambar 2. 8 Head

5. *Bag of Freebies* (BoF)

   Sekumpulan metode yang bertujuan meningkatkan akurasi dari hasil deteksi tanpa meningkatkan *inference cost* yaitu metode augmentasi data dan regularisasi.

   a. BoF for Backbone :

CutMix, dan Mosaic data augmentation, DropBlock regularization, Class label smoothing.

b. BoF for Detector :

CloU-loss, CmBN, DropBlock regularization, Mosaic data augmentation, Self-Adversarial Training, Eliminate grid sensitivity, Using multiple anchors for a single ground truth, Cosine anneling scheduler, Optimal hyperparameters, Random training shapes.

6. Bag of Specials (BoS)

Sekumpulan metode yang digunakan untuk meningkatkan *inference cost* dengan jumlah yang kecil tetapi secara signifikan meningkatkan akurasi dari hasil deteksi.

a. BoS for backbone :

Mish activation, Cross-stage partial connections (CSP), Multiinput weighted residual connections (MiWRC)

b. BoS for detector :

Mish activation, SPP-block, SAM-Block, PAN path-aggregation block, dan DIoU-NMS.

Dibawah ini merupakan tabel perbandingan YOLOv3 dengan YOLOv4 sebagai berikut :

Tabel 2. 2 Tabel Perbandingan

	YOLOv3	**YOLOv4**
Neural Network Type	Full Convolution	Full Convolution
Backbone	Darknet-53	CSPDarknet53
Loss Function	Binary cross entropy	Binary cross entropy
Neck	FPN	SSP dan PANet
Head	YOLO layer	YOLO layer

Dibanding dengan Yolov3, YOLOv4 mengalami peningkatan yang sangat signifikan yaitu peningkatan *Average Precision* (AP) sebesar 10% dan *Frame Per Second* (FPS) sebesar 12%. Terdapat beberapa kelebihan dari YOLOv4 yaitu sebagai berikut :

a. YOLOv4 adalah *state of the art* (terbaik) dari model deteksi objek

berbasis *deep learning* yang dapat mendeteksi objek dengan akurat secara *real time*, sehingga cocok untuk dijadikan pembanding pada penelitian atau pun pada saat mengerjakan proyek.

b. YOLOv4 relatif lebih mudah diinstal dibandingkan model berbasis tensorflow (contohnya Mask-RCNN) dan Caffe (contohnya SSD).
c. YOLOv4 bersifat *open source* sehingga boleh dimodifikasi sesuai kebutuhan pengguna.
d. Dokumentasinya cukup lengkap dan forumnya sangat aktif.
e. Biasanya model *deep learning* membutuhkan data yang sangat banyak untuk menghasilkan akurasi yang tinggi. Namun model *deep learning* berbasis YOLO memiliki kemampuan generalisasi yang sangat tinggi sehingga dengan sedikit data saja model tersebut dapat mendeteksi dengan baik.

Model yang akan dibandingkan adalah beberapa model yang popular yaitu *Single Shot MultiBox Detector* (SSD) dan *Mask Region Based Convolutional Neural Network* (Mask-RCNN). Perbandingan YOLOv4 dengan *Model Deep Learning* lain yaitu :

Tabel 2. 3 Perbandingan dan Performa Model

	Metode	**Ukuran Weights**	**Akurasi**	**Kecepatan**
YOLOv4	Satu tahap	256 MB	mAP = 43	83 FPS
SSD	Satu tahap	138 MB	mAP = 35,1	39 FPS
Mask-RCNN	Dua tahap	256 MB	mAP = 40,3	3 FPS

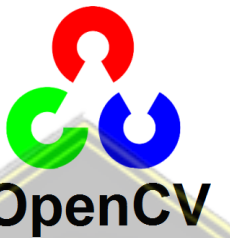
Kesimpulannya adalah YOLOv4 lebih akurat dan lebih cepat dibandingkan model lain.

2.2.4 OpenCV

Menurut buku yang berjudul “Python untuk Membuat Game hingga Face Detector” ditulis oleh Jubilee Enterprise *Open Source Computer Vision* merupakan library dengan fungsi untuk memproses terhadap citra, baik video, gambar, dan foto. Serta dapat membaca file gambar, melakukan *editing* warna, hingga mendeteksi wajah secara efektif serta memiliki

keunggulan yaitu mampu bekerja secara *real-time*.

OpenCV dirancang oleh intel dan bersifat gratis (open source), dapat digunakan dalam beberapa bahasa pemrograman yaitu bahasa C, C++, Pyhton, Java, dan sebagainya. OpenCV dapat dijalankan pada sistem operasi yaitu Windows, Android, iOS, Linux, dan Mac OS. Serta sangat mudah dan ringan untuk pengolahan citra digital.



Gambar 2. 9 Logo *OpenCV*

Salah satu tugas *computer vision* adalah *image classification*, *object detection*, dan *image segmentation*.

a. *Image classification*, menentukan kategori suatu citra berdasarkan objek yang di dalam area nya, terdapat *input image classification* yaitu citra dengan objek sedangkan *output* nya adalah hasil klasifikasi objek dari citra *input* dengan nilai probalitas. Algoritma yang digunakan dalam penelitian ini adalah CNN (*Convolutional Neural Network*).

b. *Object Detection*, menggabungkan *classification* dan *localization*. Tujuannya yaitu memprediksi posisi objek dengan *bounding box* serta mengklasifikasikan objek di setiap *bounding box*. Model *object detection* yang digunakan untuk penelitian ini ialah You Only Look Once (YOLO).

c. *Image Segmentation*, berfungsi untuk memisahkan citra menjadi beberapa area dengan masing-masing setiap bentuk dan batas tertentu. *Image Segmentation* dapat mengetahui bentuk suatu objek.

2.2.5 Darknet

Gambar 2. 10 Logo Darknet

Saat ini *darknet* diambil alih oleh Alexey Bochkovskiy, library tersebuty mendukung untuk model *YOLO*v4, dan beberapa model deteksi objek. *YOLO*v4, *YOLO*v4-Tiny, dan *YOLO*v4-CSP adalah beberapa model deteksi objek yang memiliki beberapa fitur :

1. Kode mendukung pelatihan presisi campuran untuk GPU dengan inti Tensor dapat meningkatkan kecepatan pelatihan (training) sekitar 3x lipat pada GPU yang mendukungnya.
2. Augmentasi mosaic dapat ditambahkan selama pelatihan (training) membantu meningkatkan akurasi model karena model belajar mendeteksi objek dalam gambar yang lebih sulit.

Kode mendungkung pelatihan multi-resolusi gambar sekitar 50% dari resolusi dasar setiap 10 batch saat melatih (Sovit Rath, 2022).

Darknet adalah sebuah *framework neural network* yang bersifat terbuka (*open course*) disediakan dalam bahasa pemrograman C dan Cuda. *Library Darknet* memiliki fitur, salah satunya adalah algoritma deteksi yang dikembangkan, seperti:

1. ImageNet Classification

   Proses mengklasifikasikan *image* dengan model popular seperti ResNet dan ResNeXt.
2. Tiny Darknet

   Klasifikasi gambar dengan proses dan dataset yang lebih kecil.

3. YOLO (You Only Look Once).

   YOLO adalah pendeteksian objek secara *real-time*, semua objek di setiap *frame* dideteksi dengan menampilkan *bounding box* dan *label* secara *real-time*.

   Deteksi objek menggunakan YOLOv4 (darknet) yaitu *pre-trained weights* merupakan hasil dari *training* YOLOv4 terhadap dataset COCO. Terdapat 80 class objek pada dataset COCO seperti tabel 2.3

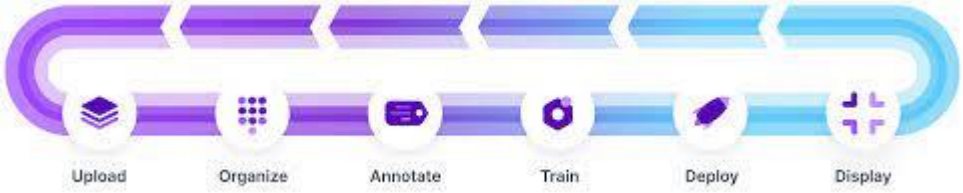
Tabel 2. 4 *class object dataset* COCO

No	Nama	No	Nama	No	Nama
1	*Person*	28	*Tie*	55	*Donut*
2	*Bicycle*	29	*Suitcase*	56	*Cake*
3	*Car*	30	*Frisbee*	57	*Chair*
4	*Motorbike*	31	*Skis*	58	*Sofa*
5	*Aeroplane*	32	*Snowboard*	59	*Potted plant*
6	*Bus*	33	*Sport ball*	60	*Bed*
7	*train*	34	*Kite*	61	*Dining table*
8	*Truck*	35	*Baseball bat*	62	*Toilet*
9	*Boat*	36	*Baseball glove*	63	*Tvmonitor*
10	*Traffic light*	37	*Skateboard*	64	*Laptop*
11	*Fire hydrant*	38	*Surfboard*	65	*Mouse*
12	*Stop sign*	39	*Tennis racket*	66	*Remote*
13	*Parking meter*	40	*Bottle*	67	*Keyboard*
14	*Bench*	41	*Wine glass*	68	*Cell phone*
15	*Bird*	42	*Cup*	69	*Microwave*
16	*Cat*	43	*Fork*	70	*Oven*
17	*Dog*	44	*Knife*	71	*Toaser*
18	*Horse*	45	*Spoon*	72	*Sink*
19	*Sheep*	46	*Bowl*	73	*Refrigerator*
20	*Cow*	47	*Banna*	74	*Book*
21	*Elephant*	48	*Apple*	75	*Clock*
22	*Bear*	49	*Sandwich*	76	*Vase*
23	*Zebra*	50	*Orange*	77	*Scissor*
24	*Giraffe*	51	*Broccoli*	78	*Teddy bear*
25	*Backpack*	52	*Carrot*	79	*Haird drier*
26	*Umbrella*	53	*Hot dog*	80	*Toothbrush*
27	*Handbag*	54	*Pizza*		

2.2.6 Roboflow

Roboflow adalah *platform* atau media yang dikhususkan untuk membantu AI engineer dalam memproses kumpulan *dataset raw* untuk

digunakan dalam *project computer vision* dengan alur kerja seperti pada gambar 2.4.



Gambar 2. 11 Alur Kerja Komputer *Vision* pada *Roboflow*

Platform *end-to-end* ini dapat menganotasikan dataset mentah menjadi dataset dengan format YOLOv4 seperti yang dibutuhkan pada penelitian yang saat ini sedang diteli. Diawali dengan mengunggah dataset *raw* kemudian platform roboflow mengolah data tersebut sehingga nantinya data dapat dianotasikan atau pemberian *bounding box*. Kemudian data tersebut dapat dijadikan bahan *input* untuk latih model. Model yang telah dilatih dapat digunakan dan hasil deteksi dari model nya dapat ditampilkan.

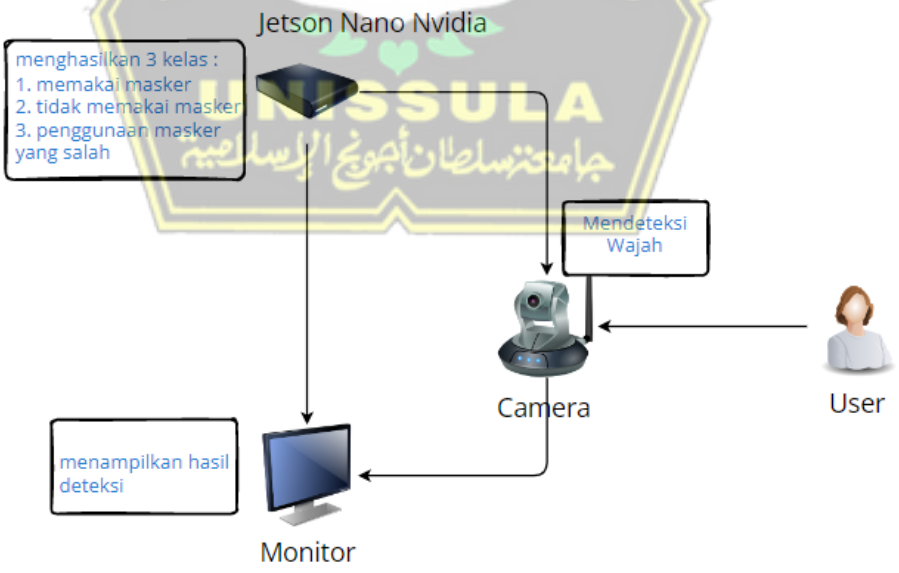
Pada Roboflow terdapat 20 library model pendukung yang dapat menjadi pilihan model saat akan mebuat projek *computer vision*, yaitu YOLOX, Vision Transformer, YOLO v3 PyTorch, YOLO v3 Keras, YOLOv4, YOLOv4-tiny, YOLOv5, YOLOR, OpenAI Clip, Scaled-YOLOv4, Resnet32, EfficientDet-D0-D7, YOLOv4 Darknet, EfficientNet, Detectron2, EfficientDet, Faster R-CNN, MobileNetSSDv2, Pytorch, MobileNetv2 Classification, dan ResNet-32.

BAB III
METODE PENELITIAN

3.1 Deskripsi Sistem

Pada penelitian ini, sistem yang akan dirancang adalah sistem mendeteksi masker wajah secara *real time* dengan metode *YOLO*v4, sistem ini termasuk salah satu penerapan *deep learning* yaitu *face recognition* metode untuk mengidentifikasi atau memverifikasi identitas seseorang menggunakan wajahnya. sistem ini berbasis Jetson Nano Nvidia tidak hanya dapat mengindentifikasi masker wajah namun mampu untuk mengklasifikasi dan memberikan nilai akurasi yang didukung dengan *library darknet*.

Kamera berperan untuk menangkap wajah *user*, kemudian sistem mendeteksi wajah mengolah informasi wajah user tersebut memvalidasi sebelum sistem melakukan klasifikasi apakah wajah *user* ini menggunakan masker, tidak menggunakan masker, atau penggunaan masker yang salah. menampilkan nilai akurasi dari hasil deteksi, sehingga sistem ini nantinya mengurangi *human error* dan mempercepat saat melewati *security gate* disuatu gedung atau tempat. Lebih jelasnya dapat dilihat sesuai dengan diagram konteks.

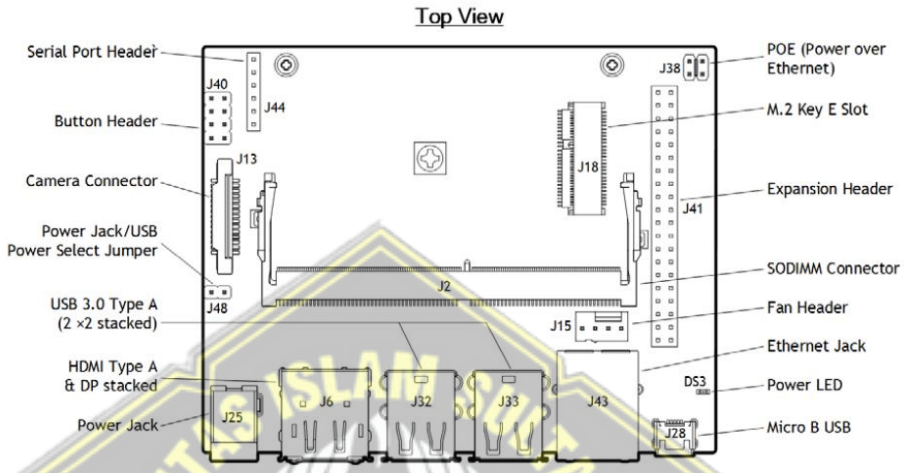


Gambar 3. 1 Perancangan *hardware face mask detection*

3.2 Kebutuhan Sistem

Pada penerapan sistem *face mask detection*, membutuhkan perangkat kerja dan perangkat lunak diantaranya sebagai berikut :

3.2.1 Jetson Nano Nvidia



Gambar 3. 2 *Design Jetson Nano Nvidia* 2GB

Sistem *face mask detection* menggunakan *Jetson Nano Nvidia* sebagai mikrokontroler atau bisa disebut juga komputer kecil. *Jetson Nano Nvidia* digunakan untuk menjalankan program. Kemudian *Jetson Nano Nvidia* bekerja yang nantinya mampu menjalankan sistem untuk mengklasifikasikan 3 kelas dari data ter*input* secara *real time* gambar berupa video *stream* menggunakan *webcam* Logitech C270. *Jetson Nano Nvidia* yang digunakan memiliki spesifikasi dijelaskan pada Tabel 3.1 berikut ini :

Tabel 3. 1 Spesifikasi *Jetson Nvidia Nano*

Sumber Daya Utama	5V 2A (via micro USB) / 5V 4A (via jack DC)
RAM (Random Access Memory)	4 GB 64-bit LPDDR4 25,6 GB
CPU (Central Processing Unit)	Quad-core ARM A57 @ 1,43 GHz
GPU (Graphics Processing Unit)	128-core Maxwell
Penyimpanan	microSD Card
Konektifitas	Gigabit Ethernet, M.2 Key E
Tampilan Output	HDMI 2.0 dan eDP 1.4
Slot Kamera	Terdapat 1x MIPI CSI-2 DPHY lanes
USB	4x USB 3.0, USB 2.0 Micro -B
Pin I/O	GPIO, I2C, I2S, SPI, UART

Video Encoder	4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264/H.265)
Dekoder Video	4K @ 60 | 2x 4K @ 30 | 8x 1080p

Tabel 3. 2 Spesifikasi *Jetson Nano Nvidia*

Komponen	**Port/Pin Jetson Nano**
Webcam Logitech C270	Port USB tipe A
Monitor	Port HDMI

Spesifikasi *Jetson Nano Nvidia* yang ditunjukkan pada tabel 3.2 yang merupakan tabel deskriptif *port webcam* Logitech C270 dan monitor untuk terhubung dengan *Jetson Nano Nvidia*. Konektor pada *port* kamera dihubungkan ke pin *cam* 0 yang berada di bagian bawah *Jetson Nano Nvidia* dan kabel HDMI untuk tampilan dihubungkan dari monitor ke *Jetson Nano Nvidia* melalui *port* HDMI yang tersedia.

3.2.2 Sistem Operasi Ubuntu

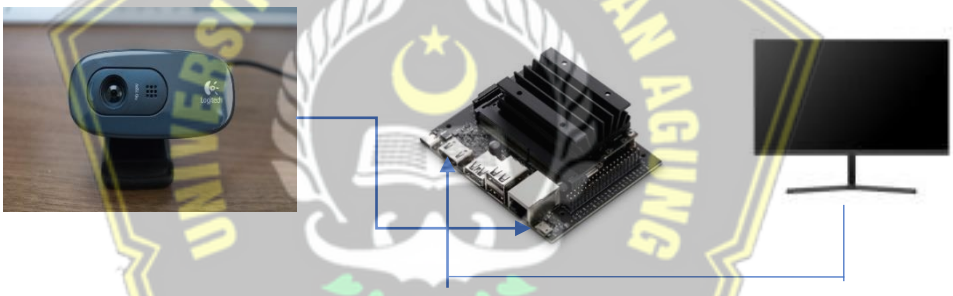
Gambar 3. 3 Sistem operasi Ubuntu

Jetson Nano Nvidia tidak berdiri sendiri *Jetson* memerlukan sistem operasi untuk menjalankan komputer kecil tersebut yaitu sistem operasi Ubuntu. Ubuntu adalah merupakan salah satu sistem operasi terbuka secara gratis (*open source*) dan salah satu perangkat lunak Linux gratis berbasis Debian. Sistem operasi Ubuntu ini sudah terdapat beberapa fitur bawaan seperti Libreoffice, Firefox, Nautilus, Rhythmbox, Totem, Thunderbird, Empathy, Evince, GNOME, dan lain nya.

3.2.3 Webcam Logitech C270

Gambar 3. 4 *WebCam* Logitech C270

Data yang tangkap oleh *webcam* Logitech C270 berupa wajah secara *real-time* yang dihasilkan dari sistem menangkap gambar wajah yang kemudian akan diolah dan diklasifikasikan oleh *Jetson Nano Nvidia*. Desain perancangan penggunaan *Webcam* Logitech C270 pada sistem *face mask detection*, keterangan rancangan penggunaan *webcam* terdapat pada gambar 3.5 sebagai berikut.



Gambar 3. 5 Perancangan penggunaan *WebCam* Logitech C270 pada *Jetson Nano*

Kamera terhubung dengan *Jetson Nano Nvidia* melalui *port* USB. Webcam Logitech C270 memiliki beberapa spesifikasi yang dapat dilihat pada Tabel 3.3 berikut ini :

Tabel 3. 3 Spesifikasi *Webcam* Logitech C270

Resolusi kamera	HD 720p
Kecepatan Bingkai	30 frame per second (FPS)
Bidang Pandang	Diagonal 55°
Kefokusan pada Lensa	Lensa Plastik
Mic Mono Omnidirectional	Terintegrasi
Fitur Teknologi	Teknologi *Noise Cancelling*
Pencahayaan	Otomatis
Klip Pemasangan	Universal
Port	USB 2.0 Tipe-A
Dapat digunakan Sistem Operasi	Windows, macOS, dan Chrome

3.2.4 Monitor

Gambar 3. 6 Monitor HP LV1561w

Pada pembuatan sistem ini penulis menggunakan Monitor merek HP LV1561w sebagai pemantau (*display*). Monitor digunakan untuk menampilkan informasi data yang telah diproses oleh *Jetson Nano Nvidia* kemudian data ditampilkan berupa video *real-time*. Hasil tampilan yang terdapat pada monitor berupa pendeksian klasifikasi serta akurasi dari menggunakan masker (mask), tidak menggunakan masker (no mask), dan penggunaan masker yang salah (bad mask).

Tabel 3. 4 Spesifikasi Monitor tipe Hp LV1561w

Tipe Monitor	LCD
Tegangan Rata-Rata	60 Hz
Waktu Respon	8 ms
Ukuran Layar	15.6 inch
Konektor Masukan	28 W
Rasio Kontras	450:1
Rasio Aspek	16:9
Bidang Pandang	90°
Resolusi	1366 x 768
Tersedia Warna	16.7 Million
Konsumsi *Power*	28 W
Brightness	200 cd/m2
Teknologi Panel	TN

3.2.5 You Only Look Once (YOLOv4)

YOLO adalah sebuah metode untuk mendeteksi objek. *YOLO* termasuk bagian dari metode *Convolutional neural networks* (CNN) yang banyak diapkikasikan pada data citra. Seperti penjelasan yang tertera pada bab 2, *YOLO* memproses gambar secara *real-time* 55 FPS pada GPU V100 jauh lebihunggul dibanding *YOLO*v3. Pada pratiknya *YOLO*v4 sudah terdapat pada *library Darknet*

sehingga *dataset* yang telah siap dapat langsung di *training* menggunakan *library* tersebut.

3.2.6 *Library Darknet*

Gambar 3. 7 Logo *library Darknet*

Pada *library* ini digunakan dalam sistem adalah fungsi *YOLO*v4 yang memiliki beberapa persiapaan sebelum dijalankan. Pertama, mengumpukan *raw dataset*. Kedua, Dataset tersebut di anotasikan pada *Roboflow*. Setelah itu hasil pengolahan disimpan dalam format *YOLOv4*. Ketiga, kemudian data yang telah di anotasikan berdasarkan kelas nya di training menggunakan *library darknet* dimana library tersebut dilengkapi algoritma otomatis sehingga ketika *training* akan menghasilkan *result* secara langsung. Jalankan inferensi *YOLO*v4 menggunakan modul OpenCV-DNN-CUDA dapat digunakan untuk pengenalan pola dalam penelitian ini berupa wajah.

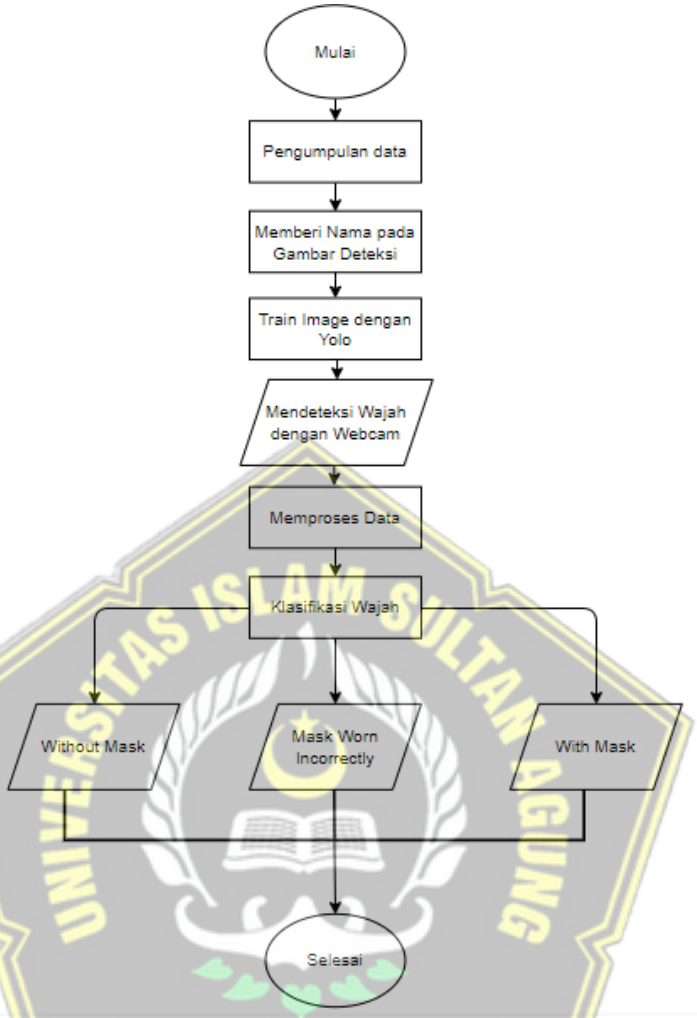
3.2.7 Objek Manusia (user)

Pada sistem ini objek yang digunakan untuk pengenalan wajah adalah manusia. Pengenalan wajah yang dilakukan untuk mengklasifikasikan objek wajah manusia dengan mengklasifikasikannya menggunakan masker (*mask*), tidak menggunakan masker (*no mask*), dan penggunaan masker yang salah (*bad mask*) skema deteksi wajah dapat dilihat pada gambar 3.3 berikut ini :

Gambar 3. 8 Pengklasifikasian Objek Manusia (*user*)

3.3 Metodologi Penelitian

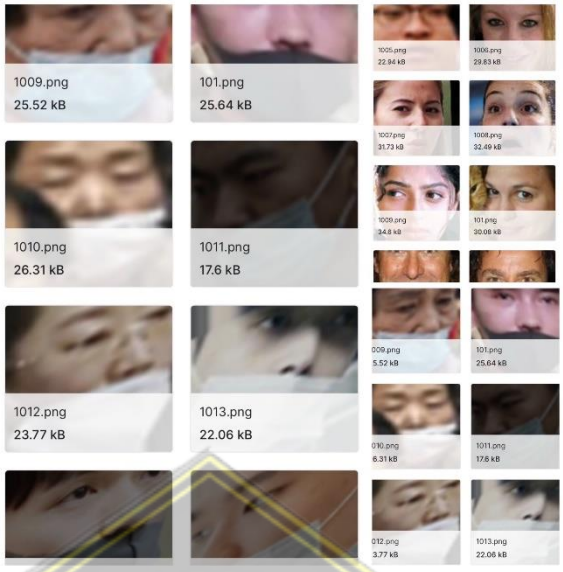
Metodologi pada sistem ini menerapkan algoritma *YOLO*v4 dari *library Darknet* untuk mendeteksi wajah dan pengenalan wajah menggunakan OpenCV. Pada pengklasifikasian wajah akan dibagi menjadi tiga kelas, yaitu pada tahap awal mengumpulan *dataset* pada Google *Dataset* Kaagle's, tahap berikutnya menganotasikan *dataset* kumpulan gambar wajah menggunakan *Roboflow*, dan *training dataset* gambar wajah tersebut menggunakan *library Darknet* :



Gambar 3. 9 *Flowchart* alur sistem

3.3.1 Pengumpulan data

Untuk membuat sistem ini diawali dengan tahap pengumpulan data, *Dataset* menggunakan data dari *Google Dataset Search* yaitu Kaggle's *Face Mask Detection* dengan sumber *dataset* lebih dari 1000 gambar yang memiliki 3 kelas yaitu *With mask, Without mask, Mask worn incorrectly* yang belum diberikan *label. Dataset* yang digunakan untuk data training yaitu 90%, dan digunakan sebagai testing yaitu 10%.



Gambar 3. 10 *dataset raw image*

3.3.2 Memberi nama pada gambar deteksi (labelling)

Pada tahap kedua dilakukan proses membuat *bounding box* yaitu pelebelan pada *dataset* untuk mendapatkan hasil pendeteksian objek dengan hasil akurasi yang baik, dan untuk memberikan label kelas pembeda diantara citra *mask, no mask,* dan *bad mask.* Penganotasian label gambar menggunakan platform Roboflow, dimana fungsi roboflow dapat mengolah data *raw* menjadi data siap *training* dengan format disesuaikan kebutuhan. Pada penelitian ini format gambar yaitu *YOLO*v4 Pytorch. Setelah membuat pelebelan pada dateset selanjutnya membuat model dan kelas yang digunakan oleh sistem.

3.3.3 *Train image* dengan YOLOv4

Pada tahap ketiga dilakukan proses *training* pada gambar yang telah diberikan label, untuk menggunakan *YOLO*v4 diperlukan *library darknet* yang di developer oleh AlexeyAB/darknet sebagai sistem kecerdasan buatan yang dapat mendeteksi suatu objek wajah. Tahapan ini memerlukan *training* untuk melatih *dataset* yang digunakan pada sistem *face mask detection*. selanjutnya kumpulan data 1000 *dataset* yang telah berformat *YOLO Darknet* terdiri dari 500 gambar mengenakan masker (with mask), 250 gambar tidak bermasker (without mask, dan 250 menggunakan masker yang salah (mask worn incorrectly) kemudian *dataset* gambar tersebut diberikan kelas dan label sesuai

masing-masing gambar. Sebelum data diolah harus melalui tahap *pre-processing*, dimana kita menyiapkan model kelas dan objek target untuk klasifikasi seperti “mask”, “no mask”, dan “bad mask” yang kemudian akan digunakan sebelum ke tahapan *training* secara keseluruhan dari sistem *face mask detection.*

1. Mengukur Kinerja

   Mean Average Precision (mAP) adalah metriks untuk mengevaluasi sebuah model deteksi objek. Nilai mAP berkisar dari 0 hingga 100, pada tahap ini menggunakan *Confusion Matrix*. Perlunya diketahui nilai prediksi dan aktual data dengan keterangan sebagai berikut:

   Nilai prediksi benar *True Positive* (TP), nilai data negative yang diprediksi benar *True Negative* (TN), nilai data negative namun diprediksi sebagai daa positif False Positive (FP), dan nilai data positif namun diprediksi data negative False Negatif (FN)

   Confusion Matrix Parameter yang dapat mengukur kinerja model klasifikasi, sebagai basis untuk mengukur performa pengujian model terhadap dataset yaitu akurasi, presisi dan *recall*.

   a. Akurasi

      Menggambarkan berapa persen kelas objek yang benar (positif) dan tidak benar prediksi (negatif).

   b. Presisi

      Rasio prediksi benar positif dibanding dengan keseluruhan hasil yang diprediksi benar (positif).

   c. Recall

      Rasio prediksi benar positif dibanding dengan keseluruhan data yang benar positif.

3.3.4 Mendeteksi wajah dengan webcam

Pada tahap ini proses dimana kamera (Logitech C270) sebagai pendeteksi menangkap wajah *user* yang terhubung dengan perangkat komputer kecil yaitu Jetson Nano Nvidia untuk tahap selanjutnya yaitu data deteksi wajah di proses.

3.3.5 Memproses data

Setelah wajah terdeteksi pada tahap ini sistem memproses data agar dapat melakukan pengklasifikasi wajah dari himpunan dataset yang sudah dikumpulkan, dengan melatih dataset dari gambar “with mask”, “without mask”, dan “mask worn incorrectly” menjadi “mask”, “no mask”, dan “bad mask”. Untuk training dataset menggunakan *library darknet*.

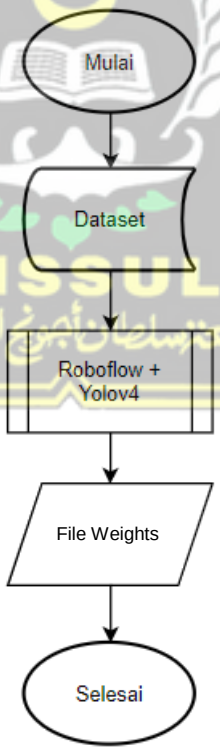
3.3.6 Klasifikasi wajah

Tahap terakhir yaitu sistem dapat pengklasifikasian wajah yang merupakan proses pengujian pada ratusan gambar agar menghasilkan keakurasian saat mendeteksi wajah secara *real-time*. Klasifikasi deteksi pada wajah terdiri dari 3 yaitu *With mask, No mask, Bad Mask.*

3.4 Desain Sistem

3.4.1 *Flowchart* Sistem

a. *Flowchart Training* Data

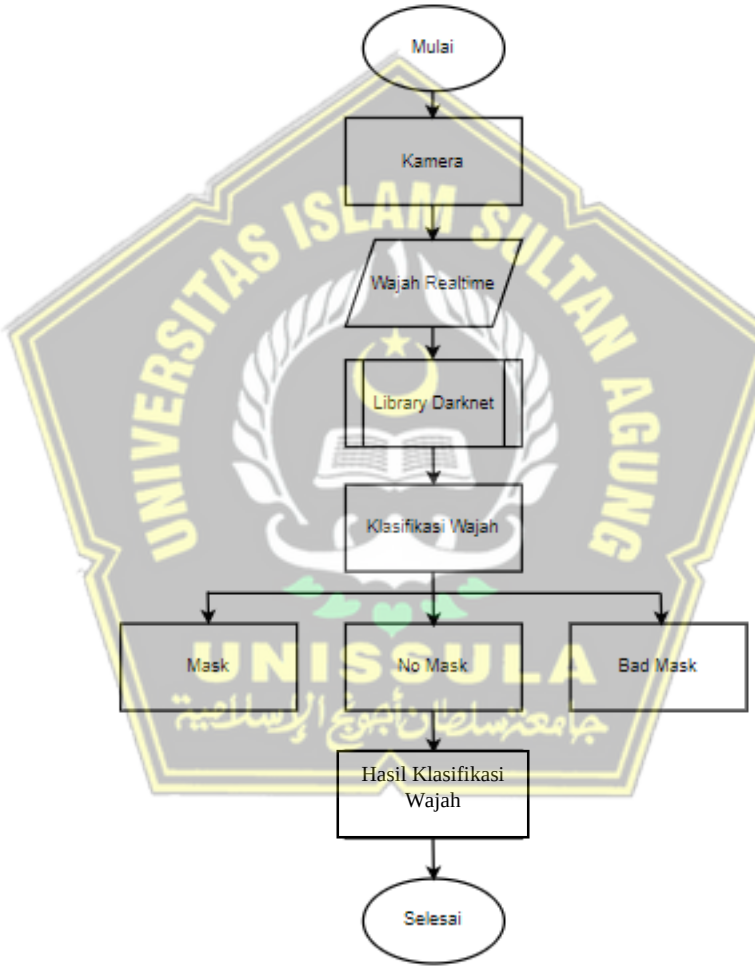


Gambar 3. 11 *Flowchart training* data

Pada gambar 3.11 menunjukan *Flowchart* dari proses data *training* pada sistem *face mask detection*. Proses ini membutuhkan *dataset* berupa kumpulan

gambar wajah yang pada tahap sebelumnya telah disimpan dari proses pengambilan data. Kemudian *dataset* tersebut diolah menggunakan *Roboflow* dimana peran *Roboflow* untuk memberi anotasi serta memberikan label penamaan, kemudian diolah dengan algoritma *YOLO*v4 dari *library Darknet*. Dari hasil Training.yml menghasilkan histogram berupa matriks, matriks tersebut mewakili gambar dari *dataset*.

b. *Flowchart* pengenalan wajah



Gambar 3. 12 *Flowchart* pengenalan wajah

Pada gambar 3.12 *Flowchart* pengenalan wajah menjelaskan proses pengenalan wajah yang dimulai ketika kamera terhubung ke *Jetson Nano Nvidia* untuk melakukan pendeteksi wajah secara *real-time*. Dengan *library Darknet* yang sudah terdapat algoritma *YOLO*v4, Peran *library Darknet* untuk mengklasifikasi

wajah kemudian didapatkan objek wajah yang kemudian diolah lagi dengan *library darknet.*

3.5 Konfigurasi Dan Instalasi Sistem

3.5.1 Konfigurasi Sistem

Pada tahap konfigurasi yang harus dilakukan adalah mengatur konfigurasi *YOLOv4 hyperparameter* nya, dan jumlah data *training* dari masing-masing klasifikasi wajah.

3.5.2 Instalasi Sistem

Perlunya *library* pendukung untuk penunjang sistem dapat berjalan dengan baik, berikut *library* yang dapat berjalan pada sistem operasi Ubuntu yang harus instalasi, yaitu :

1. YOLOv4 + Darknet
2. OpenCV versi 4.5.3
3. Python versi 3.7
4. CUDA versi 10.2.300
5. GPU dan CUDNN versi 8.2.1.32
6. Ms COCO
7. Numpy version 1.24.0

BAB IV
HASIL DAN ANALISIS PENELITIAN

4.1 Implementasi Sistem

4.1.1 Perakitan Perangkat Keras

Berikut beberapa tahap penjelasan dalam merakit perangkat keras yaitu :

1. Jetson Nano Nvidia

Gambar 4. 1 Perangkat *Jetson Nano Nvidia* 2GB

Jetson Nano Nvidia yang dipakai adalah versi 2GB dengan spesifikasi yang telah dijelaskan pada tabel 3.1 spesifikasi Jetson Nano Nvidia yaitu pada bab 3. Pada penelitian ini *Jetson Nano Nvidia* tidak menggunakan *hard case*.

2. SD Card

Gambar 4. 2 *SD Card*

Fungsi utama dari *SD Card* yaitu untuk menyimpan data, pada penelitian ini membutuhkan data operasi sistem dan kebutuhan perangkat lunak lain nya. Dengan menggunakan *SD Card* dengan kecepatan tinggi dan

size yang banyak dapat berpengaruh terhadap kinerja sistem operasi yang di pasang pada *Jetson Nano Nvidia*. *SD Card* dengan spesifikasi yang rendah memiliki kemungkinan untuk terjadinya kegagalan dalam proses instalasi sistem operasi *Ubuntu*.

3. Kamera

Gambar 4. 3 Perangkat Kamera

Implementasi penggunaan kamera dapat di letakkan dimana saja, untuk penelitian ini dipasang di depan monitor dan dihubungkan pada *Jetson Nano Nvidia*. Kualitas kamera dan pencahayaan kamera dapat berpengaruh pada hasil dari sistem *face mask detection* yang nantinya akan di jalankan. Perangkat kamera yang dipakai dalam penelitian ini dapat dilihat seperti pada gambar 4.3.

4. Converter HDMI to VGA

Gambar 4. 4 *Converter HDMI* ke *VGA*

Converter HDMI to VGA pada gambar 4.4 digunakan untuk menghubungkan kabel dari VGA ke *Jetson Nano Nvidia* karena Jetson hanya memiliki port HDMI. Dengan adanya *converter* tersebut tampilan (display) dapat di teruskan ke layar monitor.

5. Kabel VGA Monitor

Gambar 4. 5 Kabel VGA

Kabel VGA monitor digunakan untuk menampilkan hasil dari sistem (display) *Jetson Nano Nvidia* pada layar monitor maka dibutuhkannya kabel VGA seperti pada gambar 4.5.

6. Monitor

Gambar 4. 6 Monitor

Berfungsi untuk menampilkan sistem *on-screen Jetson Nano Nvidia* sebagaimana spesifikasi monitor yang digunakan dapat dilihat pada tabel 3.5 Spesifikasi Monitor tipe Hp LV1561w atau jika sedang tidak menggunakan monitor, dapat di *remote desktop connection* pada laptop secara langsung. Berfungsi untuk menampilkan hasil dari sistem (display) *Jetson Nano Nvidia* pada layar monitor dan untuk menghubungkan nya menggunakan kabel VGA seperti pada gambar 4.5.

7. AC/DC Adaptor *Power* (5V 3A tipe-C)

Gambar 4. 7 *Adaptor Power*

AC/DC Adaptor *Power* dengan spesifikasi *output* 5 *Volt* 3 *Ampere* digunakan untuk sumber daya utama dalam menjalankan *Jetson Nano Nvidia*. Gambar perangkat adaptor yang digunakan pada penelitian ini dapat dilihat pada gambar 4.7.

8. *Keyboard* dan *Mouse Wireless*

Gambar 4. 8 *Keyboard* dan *Mouse*

Keyboard dan *Mouse Wireless* Logitech dengan tipe MK235 digunakan sebagai perangkat keras pendukung untuk menjalankan *Jetson Nano Nvidia*. Gambar Keyboard dan Mouse yang digunakan dapat dilihat pada gambar diatas 4.8

4.2 Pengintegrasian Perangkat Lunak

Pada tahap pengintegrasian perangkat lunak terdapat beberapa proses, menjabarkan alur proses ini, yang mana akan penulis jabarkan dari A sampai

F dari sub bagian 4.2 sebagai berikut :

A. Instalasi kebutuhan sistem

Sebelum ketahap pengerjaan program dan *training* data, langkah awal untuk menggunakan sebuah aplikasi atau sistem.

1. Instalasi CMake, CMake adalah build tools yang bisa kita gunakan untuk build, testing, dan packaging.

```
cmake -D CMAKE_BUILD_TYPE=RELEASE
      -D CMAKE_INSTALL_PREFIX=/usr/local
      -D INSTALL_C_EXAMPLES=ON
      -D INSTALL_PYTHON_EXAMPLES=ON
      -D OPENCV_GENERATE_PKGCONFIG=ON
      -D OPENCV_EXTRA_MODULES_PATH=~/opencv_build/
```

Gambar 4. 9 Instalasi Cmake pada *Jetson Nano*

2. Instalasi Python, digunakan untuk membagi dataset yang akan dilakukan pada tahap *Training Custom Object*.

```
sudo apt-get install python3

sudo apt-get update
```

Gambar 4. 10 Instalasi Python pada *Jetson Nano*

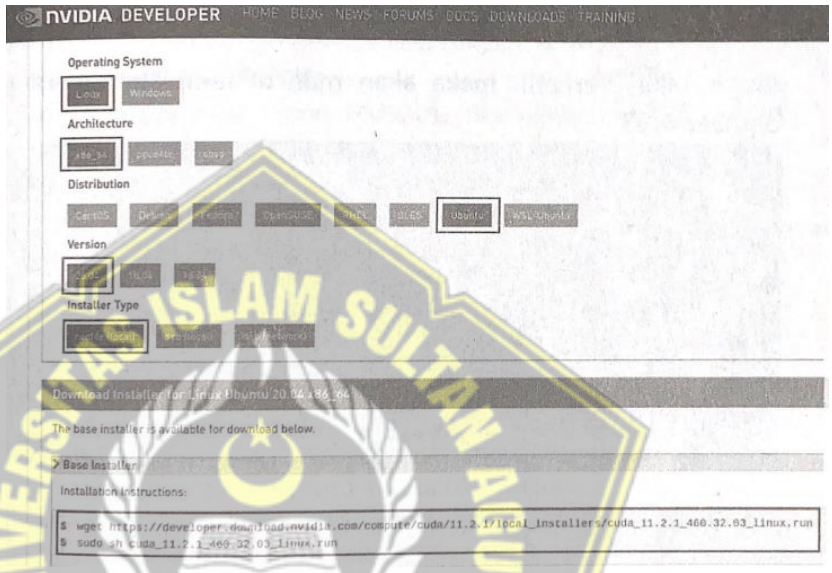
3. Instalasi OpenCV, berfungsi untuk mendeteksi wajah, pelacakan wajah, pengenalan wajah,, dan berbagai metode kecerdasan buatan atau disebut juga AI (Artificial Intellegence) dan tersedia beberapa algoritma terkait *Computer Vision* sederhana untuk low level API.

   “sudo apt install libopencv-dev python3-opencv”

```
stunningvisionai@stunningvisionai: ~/opencv_build
                                    $ git clone https://github.com/o
pencv/opencv.git
Cloning into 'opencv'...
remote: Enumerating objects: 291548, done.
remote: Counting objects: 100% (127/127), done.
remote: Compressing objects: 100% (103/103), done.
remote: Total 291548 (delta 37), reused 63 (delta 19), pack-reused 291421
Receiving objects: 100% (291548/291548), 483.28 MiB | 1.02 MiB/s, done.
Resolving deltas: 100% (202640/202640), done.
                                    $ git clone https://github.com/o
pencv/opencv_contrib.git
Cloning into 'opencv_contrib'...
remote: Enumerating objects: 35240, done.
remote: Counting objects: 100% (518/518), done.
remote: Compressing objects: 100% (365/365), done.
remote: Total 35240 (delta 215), reused 348 (delta 121), pack-reused 34722
Receiving objects: 100% (35240/35240), 131.07 MiB | 856.00 KiB/s, done.
Resolving deltas: 100% (21709/21709), done.
                                    $
```

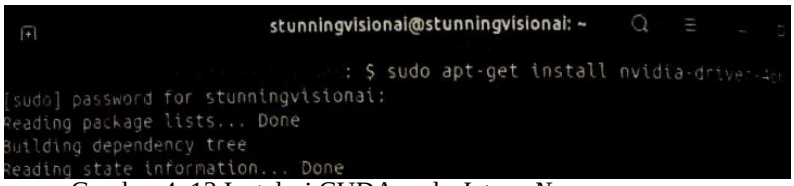
Gambar 4. 11 Instalasi *OpenCV* pada *Jetson Nano*

4. Instalasi driver GPU, mengaktifkan mode Graphics Processing Unit (GPU) yang digunakan harus berbasis Nvidia untuk meningkatkan kinerja YOLOv4 dibandingkan dengan hanya menggunakan CPU.



Gambar 4. 12 Instalasi GPU pada *Jetson Nano*

5. Instalasi CUDA, CUDA (Compute Unified Device Architecture) memiliki arsitektur dalam melakukan komputasi secara bersamaan yaitu CPU dan GPU. GPU yang dilengkapi CUDA mampu melakukan smulasi grafis dengan baik daripada hanya menggunakan CPU.

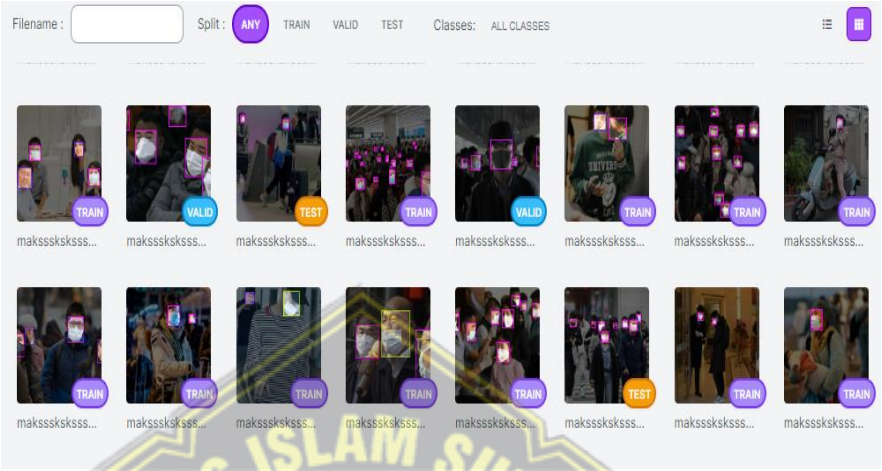


Gambar 4. 13 Instalasi CUDA pada *Jetson Nano*

6. Instalasi cuDNN, berfungsi untuk GPU yang mengatur penyetelan kinerja GPU secara otomatis. Mendukung untuk meningkatkan akselerasi pada kinerja CUDA.
   a) sudo cp cuda/include/cudnn*.h /usr/local/cuda/include
   b) sudo cp cuda/lib64/libcudnn* /usr/local/cuda/lib64

c) sudo chmod a+r /usr/local/cuda/include/cudnn*.h /usr/local/cuda/lib64/libcudnn*.

B. Menyiapkan Dataset (citra training)



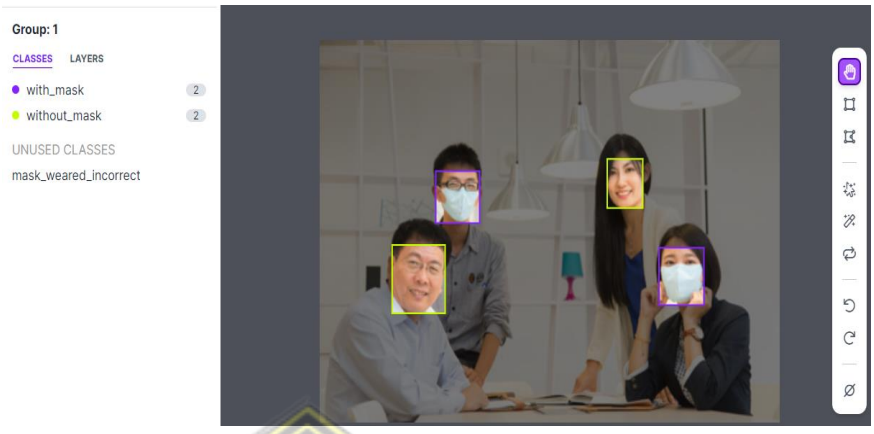
Gambar 4. 14 Dataset citra *training*

Pada tahap ini, untuk melatih model *YOLO*v4, diperlukan data berupa citra yang sudah dianotasikan. untuk penelitian ini database sudah di anotasikan melalui *platform* Roboflow yang terdiri dari 1000 dataset yaitu 5000 menggunakan masker, 250 tidak menggunakan masker, dan 250 penggunaan masker yang salah. Citra tersebut yang digunakan untuk mengajarkan apa yang perlu dideteksi oleh model. Citra dapat mewakili kondisi yang akan dihadapi model.

C. Anotasi Dataset

Setelah mengumpulkan data berupa citra, selanjutnya anotasi setiap citra tersebut berdasarkan format *YOLO*v4. Anotasi yang dilakukan yaitu dengan membuat label dimulai dari memberikan kotak batas (bounding box) dan nama kelas pada objek disetiap citra. Untuk penganotasian bisa menggunakan yolo_mark yang tersedia untuk *YOLO*v4 namun terdapat kekurangan nya yaitu penganotasian masih tergolong manual dan tidak bisa melakukan anotasi gambar dataset sekaligus harus satu persatu, maka dari itu pada penelitian ini menggunakan anotasi dari Roboflow karena sistem pada Roboflow mudah dan mendukung untuk penganotasian dengan banyak kelas.

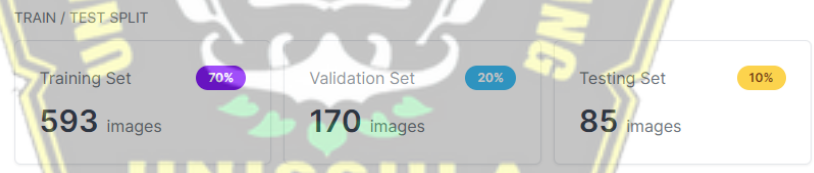
menganotasikan dapat dilihat pada gambar 4.14



Gambar 4. 15 Anotasi Dataset

D. Pembagian Dataset

Setelah dataset dianotasi, selanjutnya dataset tersebut menghasilkan menjadi 3 bagian yaitu *training*, *validation*, dan *test*. pada gambar 4.15. merupakan hasil pembagian *dataset* yang telah diolah *Roboflow* dan pada gambar 4.16 *validation dataset* menghasilkan *average precision* berdasarkan 3 kelas.



Gambar 4. 16 Pembagian *train*, *validation*, dan *test*

Komposisi Pembagian Dataset *(Default)* pada data di *Roboflow* f adalah 70% data traini, 20% data *validation*, dan 10% data *test*. Namun kita juga bisa menentukan sendiri komposisi saat pembagian dataset. Penentuan pembagian Dataset juga bisa dilakukan menggunakan split_dataset.py namun pada penelitian ini masih menggunakan *Roboflow*. Dari dataset 3 kelas tersebut yaitu *with_mask*, *without_mask*, dan *bad_mask* menghasilkan file data *train* (data/train.txt), *valid* (data/valid.txt), dan *test* (data/test.txt).

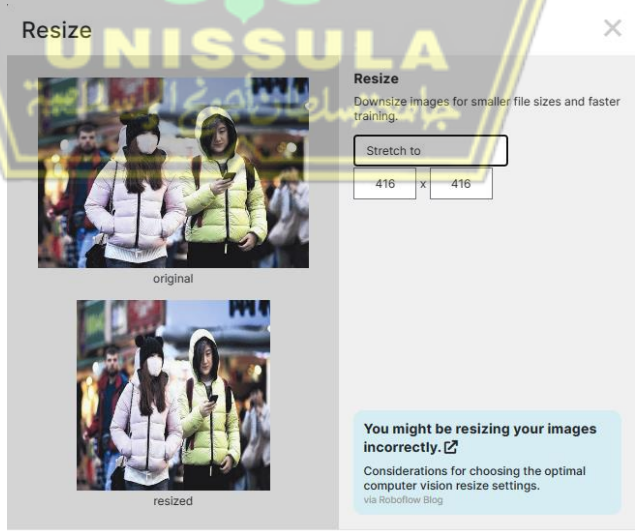
E. Preprocessing

Preprocessing pada *Roboflow* digunakan untuk mengurangi waktu

pelatihan (training) dan meningkatkan performa dengan menerapkan transformasi gambar ke semua gambar dalam kumpulan *dataset*.

Gambar 4. 17 Prepocessing

Pada Gambar 4.18 merupakan menu pilihan pada *Roboflow* dari tahap *preprocessing*, untuk penelitian ini menggunakan *filter null* karena *YOLO*v4 melakukan proses dengan 1 tahap (look once).



Gambar 4. 18 Resize

Pada Gambar 4.19 masih pada tahap *preprocessing* merupakan menu *resize* pada *Roboflow*, dilakukan *resize* menjadi 416 x 416 agar

memudahkan identifikasi pada wajah bermasker dan meningkatkan kecepatan untuk *training* gambar tersebut.

F. *Training Model*

Sebelum ke tahap *training model* dilakukan konfigurasi pada *YOLO*v4 bertujuan untuk menyesuaikan nilai *hyperparameter*. Pada file custom-yolov4-tiny-detector.cfg berisikan teks konfigurasi *default* namun perlunya beberapa konfigurasi yang perlu di edit sebagai berikut:

a. *Batch* = 64
b. *Subdivisions* = 64 (menyesuaikan GPU,, pada jetson nano saat ini yang digunakan RAM 2GB)
c. *Width* dan *height* = 416
d. Nilai *max_batches* = 6000
e. *Steps* = 4800,5400 (80%-90% dari 6000 nilai mx_batches)

```
%%writetemplate ./cfg/custom-yolov4-tiny-detector.cfg
[net]
# Testing
#batch=1
#subdivisions=1
# Training
batch=64
subdivisions=64
width=416
height=416
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.00261
burn_in=1000
max_batches = 6000
policy=steps
steps=4800,5400
scales=.1,.1
```

Gambar 4. 19 Konfigurasi YOLOv4

```
[yolo]
mask = 1,2,3
anchors = 10,14,  23,27,  37,58,  81,82,  135,169,  344,319
classes={num_classes}
num=6
jitter=.3
scale_x_y = 1.05
cls_normalizer=1.0
iou_normalizer=0.07
iou_loss=ciou
ignore_thresh = .7
truth_thresh = 1
random=0
nms_kind=greedynms
beta_nms=0.6
```

Gambar 4. 20 Konfigurasi *class* dan *filter*

Pada gambar 4. 20 menjelaskan teks yang perlu disesuaikan class pada setiap layer [yolo] dan filter pada setiap layer [convolutional].

Kemudian setelah konfigurasi dilakukan tahap *training model* ini "file-data" adalah obj.data yang dibuat pada *training model* Anotasi Dataset. Sedangkan "file-config" adalah yolov4-obj.cfg yang dibuat pada konfigurasi. File-data tersebut terdapat pada data/obj.data dan "file-config" terdapat pada cfg/yolov4-obj.cfg. untuk file utama yaitu "file-weights" yang digunakan adalah *pre-trained weights*.

```
!./darknet detector train data/obj.data cfg/custom-yolov4-detector.cfg yolov4.conv.137 -dont_show -map

 CUDA-version: 11010 (11020), cuDNN: 7.6.5, GPU count: 1
 OpenCV version: 3.2.0
 Prepare additional network for mAP calculation...
 compute_capability = 600, cudnn_half = 0
net.optimized_memory = 0
mini_batch = 1, batch = 24, time_steps = 1, train = 0
   layer   filters  size/strd(dil)      input                output
   0 conv     32       3 x 3/ 1    416 x 416 x   3 ->  416 x 416 x  32 0.299 BF
   1 conv     64       3 x 3/ 2    416 x 416 x  32 ->  208 x 208 x  64 1.595 BF
   2 conv     64       1 x 1/ 1    208 x 208 x  64 ->  208 x 208 x  64 0.354 BF
   3 route  1                                     ->  208 x 208 x  64
   4 conv     64       1 x 1/ 1    208 x 208 x  64 ->  208 x 208 x  64 0.354 BF
   5 conv     32       1 x 1/ 1    208 x 208 x  64 ->  208 x 208 x  32 0.177 BF
   6 conv     64       3 x 3/ 1    208 x 208 x  32 ->  208 x 208 x  64 1.595 BF
   7 Shortcut Layer: 4,  wt = 0, wn = 0, outputs: 208 x 208 x  64 0.003 BF
   8 conv     64       1 x 1/ 1    208 x 208 x  64 ->  208 x 208 x  64 0.354 BF
   9 route  8 2                                   ->  208 x 208 x 128
  10 conv     64       1 x 1/ 1    208 x 208 x 128 ->  208 x 208 x  64 0.709 BF
  11 conv    128       3 x 3/ 2    208 x 208 x  64 ->  104 x 104 x 128 1.595 BF
  12 conv     64       1 x 1/ 1    104 x 104 x 128 ->  104 x 104 x  64 0.177 BF
  13 route  11                                    ->  104 x 104 x 128
  14 conv     64       1 x 1/ 1    104 x 104 x 128 ->  104 x 104 x  64 0.177 BF
```

Gambar 4. 21 Proses *Training*

Pada gambar 4. 21 menjelaskan proses *training* menggunakan *Darknet*, dengan perintah "./darknet detector train data/obj.data cfg/custom-yolov4-detector.cfg yolov4.conv.137 –don't_show –map"

Tabel 4. 1 Penjelasan Perintah Deteksi Objek pada Citra

<executable>	***File executable* YOLOv4**
detector test	Perintah untuk melakukan deteksi objek pada citra
<file-data>	File yang berisi tentang jumlah *class*. Ekstensi file .data
<file-config>	File berisi konfigurasi YOLOv4 contohnya layer apa saja yang digunakan. Ekstensi file .cfg
<file-weights>	File *weights* yang digunakan sebagai *pretrained weights* sebagai input *transfer learning*. Ekstensi file .weights, File weights YOLOv4. Ekstensi file .weights
-thresh	*Flag* untuk menentukan batas *score* hasil deteksi. *Default threshold* adalah 0,25. Jika kurang dari *threshold*, maka hasil deteksi tidak akan ditampilkan.
<path-image>	Citra yang akan dideteksi.

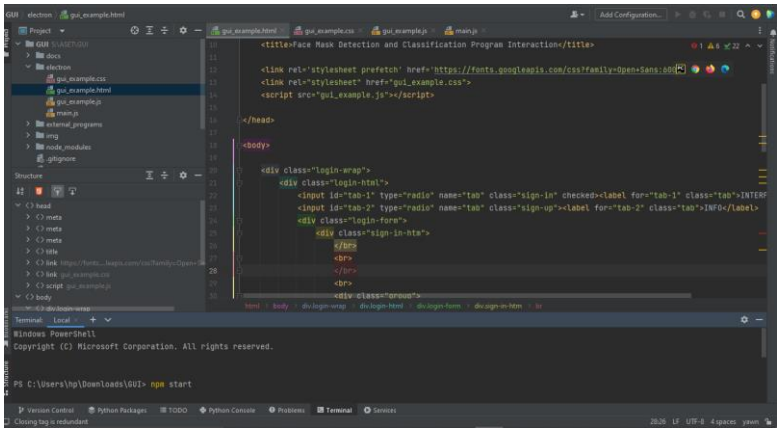
Setelah proses *training* selesai, *weights* hasil *training* akan tersimpan pada folder *YOLO*v4, kemudian dapat dilakukan *testing* terhadap *weights* dapat dilakukan dengan cara deteksi objek wajah user secara *real time*.

4.3 Pengujian Blackbox

Tahap ini dilakukan pengujian *Blackbox* untul menguji pengoperasian sistem untuk menentukan apakah sistem sudah sesuai dengan rancangan desain yang telah dibuat sebelumnya. Untuk memudahkan dalam pengujian dibagi menjadi tiga yaitu pengujian perangkat lunak, pengujian perangkat keras, dan pengujian tampilan sistem *on-screen* pada monitor. Berikut proses pengujiannya :

4.3.1 Pengujian perangkat lunak

Diawali dengan tahap pengujian perangkat lunak, pengujian dilakukan dengan tujuan untuk mengetahui apakah perangkat lunak dapat tampil dan berfungsi sesuai dengan rancangan sistem. Berikut hasil dari pengujian yaitu:



Gambar 4. 22 Code untuk GUI

Merupakan hasil dari *code* untuk membuat GUI (graphical user interface) dibuat menggunakan *library electron* berbasis dekstop, pada tampilan ini terdapat 2 menu yaitu:

Gambar 4. 23 GUI sistem *Face Mask Detection* (1)

a. *Interface*

Tampilan awal pada program dengan mengklik *launch program* berfungsi untuk menjalankan program serta menampilkan layar deteksi menggunakan webcam.

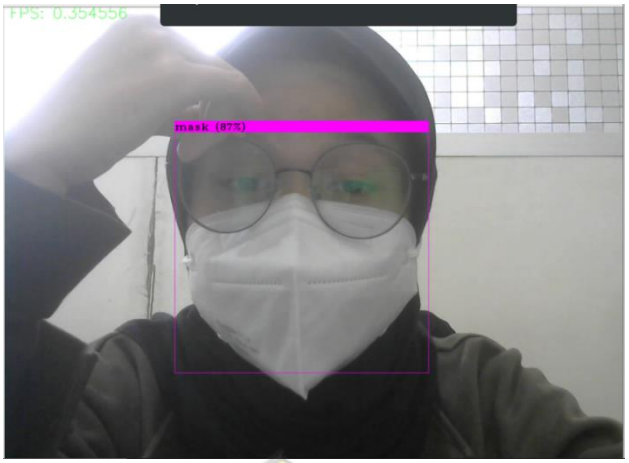
Gambar 4. 24 GUI sistem *Face Mask Detection* (1)

b. Info

Tampilan informasi sistem berupa penjelasan teks untuk memenuhi tugas akhir.

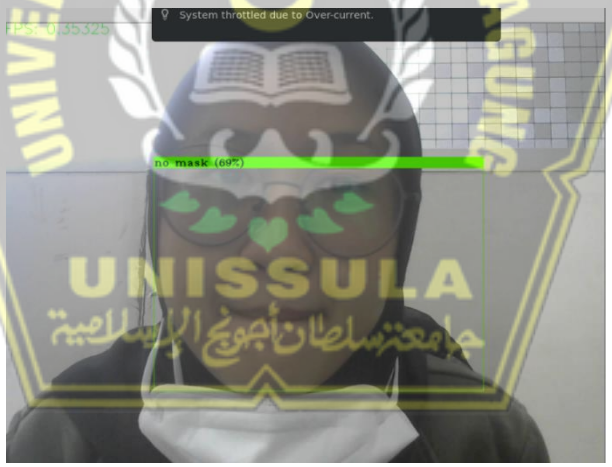
Gambar 4. 25 Grafik Kinerja GPU Saat Menjalankan Program

Gambar 4.25 merupakan grafik kinerja GPU saat menjalankan program, penjelasan bagaimana sistem bekerja dan CUDA memanfaatkan GPU secara maximal hampir 99% sehingga dikarenakan penggunaan kinerja *Jetson Nano Nvidia* sudah *maximal* maka untuk *frame rate* sudah tidak bisa di tingkatkan kembali.



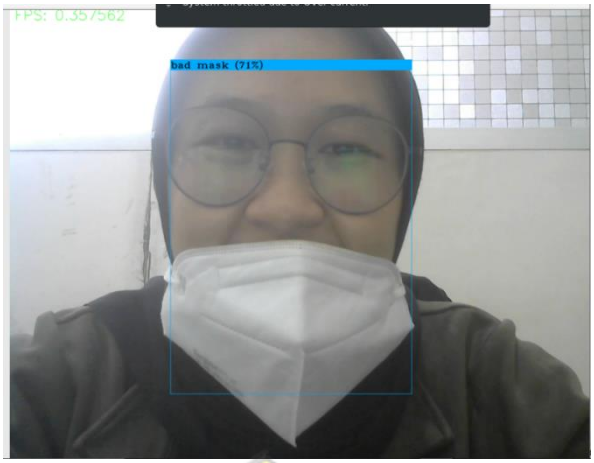
Gambar 4. 26 Hasil Menggunakan Masker

Pada tahap ini proses dalam pengambilan gambar wajah yang telah ditandai melalui kamera secara otomatis (realtime), untuk melihat hasil dari gambar wajah yang telah di terdeteksi akan tampil secara langsung dilengkapi keterangan klasifikasi dan akurasi nya. Pada gambar diatas 4.20 merupakan hasil klasifikasi menggunakan "masker".



Gambar 4. 27 Hasil Tidak Menggunakan Masker

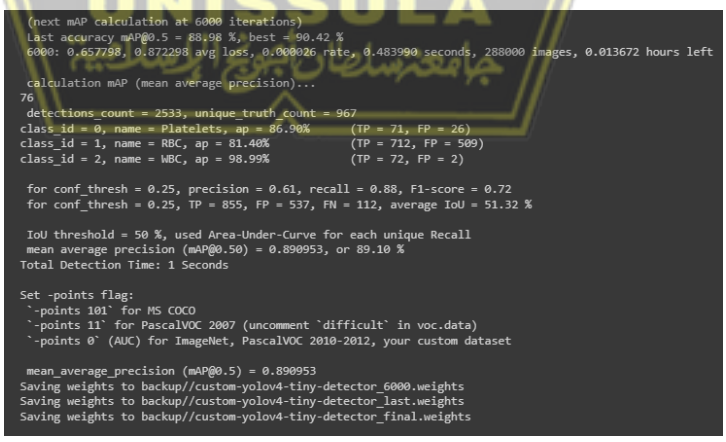
Selanjutnya pada gambar 4.21 merupakan hasil klasifikasi hasil tidak menggunakan masker "no mask" dan gambar di bawah yaitu 4.21 adalah hasil dari penggunaan masker yang salah "bad mask".



Gambar 4. 28 Hasil Penggunaan Masker Yang Salah

*Mean Average Precisio*n (mAP) adalah untuk mengevaluasi sebuah model deteksi objek. Nilai mAP berkisar dari 0 hingga 100. Semakin tinggi angkanya, semakin baik. mAP dapat dihitung dengan menghitung *average precision* (AP) secara terpisah untuk setiap kelas, kemudian nilai AP per kelas tersebut di rata-ratakan.

Menghitung *Average Precision* (AP) terdapat pada library *darknet* dengan menggunakan YOLOv4 dimana dapat menghitung *all point interpolated.* dan kecepatan untuk mengetahui sebuah model saat mendeteksi objek, hasil training menghasilkan nilai mAP dan F1-*score*. Hasil *training* dapat dilihat pada gambar 4.29



Gambar 4. 29 Hasil Training

Hasil perhitungan pada *YOLO*v4 melalui *library darknet* didapat dari hasil training *YOLO*v4-Tiny sehingga perhitungan akurasi didapat

secara otomatis. Penjelasan hasil percobaan *face mask detection* terdapat pada tabel 4.4 sebagai berikut.

Tabel 4. 2 Hasil percobaan *face mask detection*

Ket	Klasifikasi			Akurasi		
	Mask	No Mask	Bad Mask	Mask	No Mask	Bad Mask
Orang Ke-1	B	B	B	87 %	69%	71%
Orang Ke-2	B	B	B	80%	78%	70%
Orang Ke-3	B	B	B	81%	75%	69%
Orang Ke-4	B	B	X	73%	60%	60%
Orang Ke-5	B	B	X	70%	60%	60%
Total				78,2%	67,2%	66%

Keterangan :

B = Pendeteksian Benar

S = Pendeteksian Salah

X = Pendeteksian *double*

Dari table percobaan *face mask detection* di atas dapat diketahui bahwa :

Hasil dari percobaan keseluruhan pada tabel 6.4 diatas

1. Jumlah percobaan semua = 10 kali
2. Jumlah percobaan benar = 7 kali
3. Jumlah percobaan salah = 1 kali
4. Jumlah percobaan pendeteksian *double* = 2 kali

Dari Hasil perhitungan pada diatas hasil yang didapat belum sepenuhnya sempurna, namun cukup memuaskan untuk dapat digunakan di tempat umum. Maka dari itu perlu ditingkatkan kembali agar mendapat hasil yang lebih baik, diantaranya :

1. Meng *upgrade* perangkat keras diatas nya *Jetson Nano Nvidia* 2GB, seperti *Jetson Nano Nvidia* versi 4GB atau NX.
2. Penggunaan *library* selain *Darknet*.
3. Ditingkatkan nya versi *YOLO* yang terbaru dengan didukung oleh perangkat keras yang menunjang
4. *Dataset* yang lebih banyak dan lebih bersih dalam penganotasian nya.

4.3.2 Pengujian perangkat keras

Hardware testing (pengujian perangkat keras) yaitu merupakan tahap pengujian agar penulis dapat mengetahui apakah perangkat seperti *Jetson Nano Nvidia*, monitor, *SD Card* dan perangkat-perangkat pendukung lain nya dapat bekerja sesuai dengan sistem rancangan yang dibuat. Hasil pengujian terdapat pada Tabel 6.5 sebagai berikut.

Tabel 4. 3 Pengujian perangkat keras

Bahan Yang Diujikan	Perencanaan Uji	Output Yang Diharapkan	Hasil Pengujian
Perangkat Keras	Beberapa komponen perangkat keras utama yang harus dirakit yaitu Jetson Nano Nvidia, SD Card, Kamera, dan perangkat keras pendukung seperti Monitor, Keyboard, dan Mouse	Semua perangkat keras sudah terpasang dengan baik dan berfungsi	**Berhasil**, terpasang
Jetson Nano Nvidia (2 GB)	Menghubungkan perangkat Jetson Nano Nvidia dengan wifi	Jetson Nano Nvidia dapat terhubung ke wifi dengan baik	**Berhasil**, menghubungkan perangkat yang diperlukan Jetson Nano Nvidia
SD Card (256 GB)	Instalasi sistem operasi Ubuntu dan menjalankannya sistem operasi tersebut pada komputer Jetson Nano Nvidia berbasis Linux	Sistem operasi telah berhasil di pasang dan dijalankan pada komputer Jetson Nano Nvidia	**Berhasil**, instalasi sistem operasi
Kamera	Dapat menampilkan (memproyeksikan) gambar (yang dikemas dalam video real-time) pada layar monitor	Gambar dapat tampil pada layar monitor	**Berhasil**, menampilkan sistem

4.3.3 Pengujian Tampilan Sistem On-Screen pada Monitor

Pengujian tampilan sistem *on-screen* pada monitor, merupakan perangkat keras pendukung sistem yang digunakan untuk menguji apakah sistem dapat tampil di layar dan berfungsi sebagaimana dengan perencanaan yang telah dibuat sebelumnya. Tabel hasil pengujian terdapat pada Tabel 6.2 sebagai berikut.

Tabel 4. 4 Pengujian tampilan sistem pada monitor

Bahan Yang Diujikan	Perencanaan Uji	Output Yang Diharapkan	Hasil Pengujian
Tampilan dapat melihat hasil klasifikasi dan akurasi	Dapat menampilkan (memproyeksikan) hasil klasifikasi wajah dan akurasi nya	Sistem dapat menampilkan data user dengan baik dan dapat menjalankan fungsi nya	**Berhasil**, menampilkan sistem on-screen pada monitor

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari semua rangkaian tahapan penelitian ini menghasilkan kesimpulan bahwa implementasi sistem mendeteksi penggunaan masker secara *realtime* dengan metode YOLOv4 dan *library darknet* beserta OpenCV dapat bekerja pada perangkat *Jetson Nvidia Nano* dengan hasil yang cukup memuaskan untuk dapat digunakan di tempat umum yaitu dengan hasil akurasi dari semua percobaan mendapatkan nilai pada "mask" 78,2% , "no mask" 68,4% , dan "bad mask" 66%. Namun dari hasil penelitian tersebut masih cukup rendah jika di gunakan untuk skala yang lebih banyak dan luas seperti tempat umum dengan kerumunan orang atau jumlah pengunjung yang banyak.

Dari percobaan sistem setelah dijalankan memilki kelemahan dalam *frame rate* (FPS) yaitu terjadi selisih 6 detik dari *realtime* yang diharapkan, namun sistem berjalan dengan baik sebagaimana fungsinya mampu mendeteksi, mengklasifikasi, dan memberikan nilai akurasi.

5.2 Saran

A. Beberapa saran untuk perangkat sistem :

1. Jika memungkinkan untuk menunjang penggunaan secara *real-time* diperlukan peningkatan performa kecepatan sistem dapat di *upgrade* dari *Jetson Nano Nvidia* 2GB menjadi 4GB atau diatasnya NX dengan performa yang lebih baik.
2. Jarak peletakan kamera ke setiap wajah antara 50 – 100 cm.
3. Pencahayaan pada kamera ke wajah dan jenis kamera yang dapat mendukung bekerja nya sistem.

B. Beberapa saran yang dianjurkan untuk pengembangan sistem dan penelitian selanjutnya :

1. Perlu ditambahkan library dan algoritma lain nya untuk menunjang *alert*

suara ketika pengguna tidak menggunakan masker ataupun pengguna masker yang kurang tepat.

2. Karena dari hasil akurasi keseluruhan 3 kelas klasifikasi yaitu 70,86% , masih rendah dan perlu di tingkatkan kembali, sebaiknya pada penelitian berikutnya dapat digunakan algoritma pengenalan pola dan objek selain *YOLO*v4 dan *library* selain *darknet*.

DAFTAR PUSTAKA

Aprilian Anarki, G., Auliasari, K., & Orisa, M. (2021). Penerapan Metode Haar Cascade Pada Aplikasi Deteksi Masker. *JATI (Jurnal Mahasiswa Teknik Informatika)*, *5*(1), 179–186. https://doi.org/10.36040/jati.v5i1.3214

Britt Yip and Valeria Perasso. (2021). *Asal Covid-19: Apakah kita perlu tahu dari mana asal virus corona ini?* BBC News Indonesia. https://www.bbc.com/indonesia/dunia-57590872

Budiarto Hadiprakoso, R., & Qomariasih, N. (2022). Deteksi Masker Wajah Menggunakan Deep Transfer Learning Dan Augmentasi Gambar. *JIKO (Jurnal Informatika Dan Komputer)*, *5*(1), 12–18. https://doi.org/10.33387/jiko.v5i1.3591

Indra Jaya. (2021). *Penguatan Sistem Kesehatan dalam Pengendalian COVID-19*. Kementrian Kesehatan Republik Indonesia. http://p2p.kemkes.go.id/penguatan-sistem-kesehatan-dalam-pengendalian-covid-19/

Nalinipriya, G., Shobana, M., Siva, C., Kanisha, B., Monica, J. K., & Siva Vadivu Ragavi, V. (2022). Dynamic Face Mask Detection Using Machine Learning. *1st IEEE International Conference on Smart Technologies and Systems for Next Generation Computing, ICSTSN 2022*, 1–7. https://doi.org/10.1109/ICSTSN53084.2022.9761291

Septiana, T., Puspita, N., Fikih, M. Al, & Setyawan, N. (2020). Face Mask Detection Covid-19 Using Convolutional Neural Network (Cnn). *Seminar Nasional Teknologi Dan Rekayasa (SENTRA) 2020*, *3*, 27–32.

Sovit Rath. (2022). *Pothole Detection using YOLOv4 and Darknet*. Learnopencv. https://learnopencv.com/pothole-detection-using-yolov4-and-darknet/

Shin DJ, Kim JJ. A Deep Learning Framework Performance Evaluation to Use YOLO in Nvidia Jetson Platform. *Appl Sci*. 2022;12(8). doi:10.3390/app12083734.

Pandey VK, Gupta VK, Kumar S. Face Mask Detection Using Convolutional Neural Network. *Proc - 2021 3rd Int Conf Adv Comput Commun Control Networking, ICAC3N 2021*. Published online 2021:951-954.

doi:10.1109/ICAC3N53548.2021.9725689.

Arkhamia Batubara N., Maulana Awangga R.,. Tutorial Object Detection Plate Number With Convolution Neural Network (CNN). Kreatif. 2020:81-83.