DETEKSI KANTUK BERBASIS JETSON NANO BAGI PENGEMUDI RODA EMPAT MENGGUNAKAN YOLOV5

LAPORAN TUGAS AKHIR

Proposal ini disusun guna memenuhi salah satu syarat untuk menyelesaikan program studi Teknik Informatika S-1 pada Fakultas Teknologi Industri Universitas Islam Sultan Agung

DISUSUN OLEH :

NAMA : MUHAMMAD SUTRISNO

NIM : 32601800018

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM SULTAN AGUNG

SEMARANG

2022

FINAL PROJECT

JETSON NANO BASED DROWSINESS DETECTION FOR CAR DRIVER USING YOLOV5

Proposed to complete the requirement to obtain a bachelor's degree (S-1) at Informatics Engineering Departement of Industrial Technology Faculty Sultan Agung Islamic University

ARRANGED BY:

NAMA : MUHAMMAD SUTRISNO

NIM : 32601800018

MAJORING OF INFORMATICS ENGINEERING

INDUSTRIAL TECHNOLOGY FACULTY

SULTAN AGUNG ISLAMIC UNIVERSITY

SEMARANG

2022

LEMBAR PENGESAHAN PEMBIMBING

Laporan Akhir dengan judul **"DETEKSI KANTUK BERBASIS JETSON NANO BAGI PENGEMUDI RODA EMPAT MENGGUNAKAN YOLOV5"** ini disusun oleh :

Nama : Muhammad Sutrisno

NIM : 32601800018

Program Studi : Teknik Informatika

Telah disahkan oleh Dosen Pembimbing pada :

Hari : ....Kamis.....

Tanggal : ..2 Maret 2023.....

Mengesahkan,

Pembimbing I	Pembimbing II
Muhammad Qomaruddin, ST, M.Sc, Ph.D	Ir. Sri Mulyono, M.Eng.
NIDN. 0631057101	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 **"DETEKSI KANTUK BERBASIS *JETSON NANO* BAGI PENGEMUDI RODA EMPAT MENGGUNAKAN *YOLOV5* "** ini telah dipertahankan di depan dosen penguji Tugas Akhir pada :

Hari : Kamis

Tanggal : 2 Maret 2023

TIM PENGUJI

Penguji I	Penguji II
Imam Much Ibnu S, ST.MSc, Ph.D	Andi Riansyah, ST, M.Kom
NIDN. 0613037301	NIDN. 0609108802

SURAT PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan dibawah ini :

Nama : Muhammad Sutrisno

NIM : 32601800018

Judul Tugas Akhir : Deteksi Kantuk Berbasis Jetson Nano Bagi Pengemudi Roda Empat Menggunakan Yolov5

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, 2 Maret 2023

Yang Menyatakan,

METERAI TEMPEL

Muhammad Sutrisno

PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH

Saya yang bertanda tangan dibawah ini :

Nama : Muhammad Sutrisno

NIM : 32601800018

Program Studi : Teknik Informatika

Fakultas : Teknologi Industri

Alamat Asal : Ds. Tambahagung Dk. Kluwung 05/02 , Tambakromo, Pati

Dengan ini menyatakan Karya Ilmiah berupa Tugas akhir dengan Judul : Deteksi Kantuk Berbasis Jetson Nano Bagi Pengemudi Roda Empat Menggunakan Yolov5 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, 2 Maret 2023

Yang menyatakan,

Muhammad Sutrisno

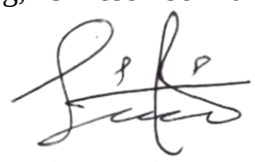
KATA PENGANTAR

Dengan mengucap syukur alhamdulillah atas kehadirat Allah SWT yang telah memberikan rahmat dan karunianya kepada penulis, sehingga dapat menyelesaikan Tugas Akhir dengan judul "Deteksi Kantuk Berbasis Jetson nano Bagi Pengemudi Roda Empat Menggunakan YOLOv5" ini untuk memenuhi salah satu syarat menyelesaikan studi serta dalam rangka memperoleh gelar sarjana (S-1) pada Program Studi Teknik Informatika Fakultas Teknologi Industri Universitas Islam Sultan Agung Semarang. Tugas Akhir ini disusun dan dibuat dengan adanya bantuan dari berbagai pihak, materi maupun teknis, oleh karena itu saya selaku penulis mengucapkan terima kasih kepada:

1. Rektor UNISSULA Bapak Prof. Dr. H. Gunarto, S.H., M.H yang mengizinkan penulis menimba ilmu di kampus ini.
2. Dekan Fakultas Teknologi Industri Ibu Dr. Novi Marlyana, S.T., M.T.
3. Dosen pembimbing I penulis M.Qomaruddin, S.T. M.Sc Ph.D. yang telah meluangkan waktu dan memberi ilmu.
4. Dosen pembimbing II penulis Ir.Sri Mulyono, M.Eng. yang memberikan banyak solusi dan saran.
5. Orang tua penulis yang telah mengizinkan untuk menyelesaikan laporan ini,
6. Dan kepada semua pihak yang tidak dapat saya sebutkan satu persatu.

Dengan segala kerendahan hati, penulis menyadari masih terdapat banyak kekurangan dari segi kualitas atau kuantitas maupun dari ilmu pengetahuan dalam penyusunan laporan, sehingga penulis mengharapkan adanya saran dan kritikan yang bersifat membangun demi kesempurnaan laporan ini dan masa mendatang.

Semarang, 29 Desember 2022



Muhammad Sutrisno

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Bekerja dalam keadaan mengatuk mengakibatkan pekerjaan yang dilakukan menjadi kurang efisien dan dampak yang ditimbulkan juga tidak sepele. Pada tahun 2017 terjadi peningkatan angka kecelakaan sebesar 16%, dimana pada tahun 2016 terdapat sebanyak 63 kecelakaan pada enam hari sebelum lebaran sedangkan pada tahun 2017 terdapat 73 kecelakaan. Penelitian ini menggunakan algotritma YOLOv5 untuk mendeteksi wajah mengantuk yang diterapkan pada perangkat Jetson Nano. Jetson Nano dapat memberikan kemudahan proses implementasi untuk deteksi dan pelatihan model kepada penggunanya. Dalam mengklasifikasi objek yang terdeteksi sistem ini menggunakan You Only Look Once (YOLOv5). Hasil gambar objek tersebut akan diolah menjadi pengolahan citra oleh computer vision, dan menghasilkan serta menampilkan data real-time wajah yang mengantuk dan tidak mengantuk. Hasil penelitian dievaluasi berdasarkan tabel perhitungan rata rata keakuratan klasifikasi oleh model untuk mendapatkan hasil deteksi yaitu Drowsy (Mengantuk) dan Awake (Sadar). Hasil penelitian menunjukkan bahwa dari semua pengujian pada keadaan dengan jarak 50 cm, 70 cm dan cahaya redup adalah sekitar 61,9 % untuk wajah mengantuk dan 77,84% untuk wajah sadar.
Kata kunci : Kantuk, Jetson Nano, YOLOv5

ABSTRACT

Working in a drowsy state causes the work to be done to be less efficient and the impact is not trivial. In 2017 there was an increase in the number of accidents by 16%, where in 2016 there were 63 accidents in the six days before Eid, while in 2017 there were 73 accidents. This study uses the YOLOv5 algorithm to detect sleepy faces which is applied to the Jetson Nano device. Jetson Nano can provide users with an easy implementation process for model detection and training. In classifying detected objects this system uses You Only Look Once (YOLOv5). The results of the object images will be processed into image processing by computer vision, and produce and display real-time data on sleepy and not sleepy faces. The results of the study were evaluated based on the calculation table for the average classification accuracy by the model to obtain detection results, namely Drowsy and Awake. The results showed that of all tests in conditions with a distance of 50 cm, 70 cm and dim light, it was around 61.9% for sleepy faces and 77.84% for conscious faces.

Keywords : Drowsy, Jetson Nano, YOLOv5

BAB I
PENDAHULUAN

1.1 Latar Belakang

Bekerja dalam keadaan mengatuk mengakibatkan pekerjaan yang dilakukan menjadi kurang efisien dan dampak yang ditimbulkan juga tidak sepele. Contohnya berkendara dalam keadaan mengantuk yang mengakibatkan kecelakaan. Tidak hanya kerusakan pada harta benda kecelakaan saat berkendara dapat merenggut korban jiwa. Terlebih lagi jika kecelakaan tersebut terjadi pada transportasi umum. Faktor utama yang menyebabkan adanya kejadian kecelakaan yakni disebabkan dari adanya rasa lelah tubuh yang mengakibatkan kelelahan dan mengantuk, hal ini banyak terjadi pada tiap pukul 21.00-24.00 lalu disusul pukul 03.00-06.00 (Puteri & Utaminingrum, 2020).

Pada tahun 2017 terjadi peningkatan angka kecelakaan sebesar 16%, dimana di tahun 2016 terdapat sebanyak 63 kecelakaan pada enam hari sebelum lebaran sedangkan pada tahun 2017 terdapat 73 kecelakaan. Karena besarnya peningkatan angka kecelakaan setiap tahunnya dibutuhkan sebuah upaya yang dapat menurunkan dan mengurangi tingkat kecelakaan yang akan terjadi pada tahun-tahun selanjutnya.

Dalam merealisasikan upaya tersebut dibutuhkan sebuah sistem yang dapat mendeteksi kantuk pada pengemudi dan perilaku seseorang yang bahkan dapat diklasifikasikan lebih lanjut apakah lelah dan atau mengantuk yang digunakan untuk pendeteksian reaksi manusia oleh perangkat cerdas. Algoritma *machine learning* dapat digunakan untuk deteksi mengantuk karena telah terbukti sangat berguna dalam pengenalan dan klasifikasi pola. Kecerdasan *AI modern* sekarang sudah dapat digunakan dan tersedia untuk pembuat, pelajar, serta pengembang dan bisa diterapkan di mana saja. Diantaranya yaitu komputer mini yang sering digunakan para developer untuk membuat sistem *Artifficcial Intellegence* yaitu *Jetson Nano Developer Kit*. Komputer ini adalah komputer kecil yang dapat digunakan untuk menjalankan beberapa jaringan saraf secara paralel untuk aplikasi seperti klasifikasi gambar, deteksi objek, dan segmentasi. Oleh karena itu, *Jetson Nano* dapat memberikan kemudahan proses implementasi untuk deteksi dan pelatihan model

kepada penggunanya. Dalam mengklasifikasi objek yang terdeteksi sistem ini menggunakan You Only Look Once (YOLOv5). Perangkat Jetson Nano Developer Kit akan dihubungkan dengan kamera yang digunakan untuk menangkap dan mendeteksi gambar objek. Hasil gambar objek tersebut akan diolah menjadi pengolahan citra oleh computer vision, dan menghasilkan serta menampilkan data real-time wajah yang mengantuk dan tidak mengantuk.

Dari penjelasan di atas maka penulis mengusulkan judul "Deteksi Kantuk Berbasis *Jetson Nano* Bagi Pengemudi Roda Empat Menggunakan *You Only Look Once (*YOLOv5)*"* yang dapat memberikan solusi dari penyebab kecelakaan yakni mengemudi dengan keadaan mengantuk. Maka dengan dibuatnya penelitian pendeteksi kantuk ini diharapkan kecelekaan yang sering terjadi akibat faktor kelalaian manusia dapat berkurang dan agar hal tersebut dapat diminimalisasi menyesuaikan kondisi kewaspadaan yang ada.

1.2 Rumusan Masalah

Berdasarkan adanya latar belakang diatas, maka dapat dirumuskan permasalahan sebagai berikut:

1. Bagaimana melakukan perancangan sistem deteksi kantuk menggunakan kamera pada Jetson nano ?
2. Bagaimana melakukan perancangan sistem deteksi kantuk menggunakan metode YOLOv5 ?

1.3 Batasan Masalah

Berdasarkan adanya perumusan masalah di atas, maka dapat dibatasi masalah penelitian ini sebagai berikut :

1. Metode yang digunakan yaitu YOLOv5 serta klasifikasi *image processing.*
2. Bahasa pemrograman yang digunakan adalah *Phyton.*

1.4 Tujuan

Berdasarkan rumusan masalah yang diuraikan diatas, maka tujuan penelitian ini adalah sebagai berikut :

1. Merancang sistem deteksi kantuk mengunakan *Jetson nano*.
2. Melakukan implementasi metode YOLOv5 pada sistem deteksi kantuk

berbasis *Jetson nano* untuk memperingatkan pengendara yang mengantuk.

1.5 Manfaat

Manfaat yang diharapkan dari system ini adalah :

1. Mengurangi resiko kecelakaan yang ditimbulkan dari keadaan mengantuk
2. Memberi peringatan sehingga meningkatkan kewaspadaan pengendara

1.6 Sistematika Penulisan

Sistematika penulisan yang akan digunakan oleh penulis dalam sebuah pembuatan laporan tugas akhir adalah sebagai berikut:

BAB 1: PENDAHULUAN

Pada bab ini penulis mengutarakan latar belakang pemilihan judul, rumusanmasalah, batasan masalah, tujuan penelitian, metodologi penelitian, dan sistematika penulisan.

BAB 2: TINJAUAN PUSTAKA DAN DASAR TEORI

Bab ini memuat penelitian-penelitian sebelumnya dan dasar teori yang berguna untuk membantu sebuah penulis untuk memahami bagaimana konsep kerja pada *Jetson nano* untuk sistem deteksi kantuk dengan bahasa pemrograman dan *library* yang sudah dipilih.

BAB 3: METODE PENELITIAN

Bab ini mengungkapkan proses tahapan-tahapan penelitian dimulai dari perancangan sistem, alur kerja sistem, dan pengujian sistem.

BAB 4: HASIL PENELITIAN

Pada bab ini penulis mengungkapkan hasil penelitian yakni implementasi sistem pada komputer kecil menggunakan *Jetson nano* Nvidia dengan bantuan perangkat keras menggunakan kamera sebagai alat mendeteksi wajah yang kemudian di proses pengklasifikasian nya menggunakan metode Yolov5.

BAB 5: KESIMPULAN DAN SARAN

Bab ini penulis memaparkan kesimpulan daripada proses penelitian dari awal hingga akhir dan saran yang akan dikembangkan.

BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Pada tinjauan pustaka ini, penulis akan membahas tentang penelitian sebelumnya yang terkait dengan apa yang dilakukan oleh penulis yaitu tentang sistem deteksi kantuk. Tinjauan pustaka ini nantinya akan dapat digunakan sebagai referensi di dalam melakukan penelitian tersebut.

Pada jurnal yang berjudul Deteksi Mata Mengantuk Pada Pengemudi Mobil Menggunakan Metode *Viola Jones,* menjelaskan cara kerja metode *Viola-Joness*. Metode *Viola-Joness* merupakan metode deteksi mata mengantuk pada pengendara roda 4 untuk mengetahui pengendara tersebut mengantuk atau tidak saat berkendara. Input yang digunakan adalah deteksi mata yang ditangkap oleh kamera digital dan kemudian dimasukkan ke dalam bahasa pemrograman GUI matlab, di mana diambil nilai mata biner, mata ngantuk dan mata tidak mengantuk untuk dijadikan acuan yang nantinya akan diolah agar dapat menghasilkan alarm pengingat kepada pengendara yang sedang mengantuk maupun tidak mengantuk, dengan level BW yang digunakan pada program ini adalah 0.225 sehingga mata dapat dideteksi dengan cepat. Level BW ini bisa berubah tergantung intensitas cahaya yang diterima program ini ketika digunakan, dan dalam hal ini penggunaannya pada tempat dengan intensitas cahaya yang buruk masih dalam tahap penggujian (Imanuddin dkk, 2019).

Pada 2021, Jurnal yang berjudul Deteksi Kantuk Pengendara Roda Empat Menggunakan Haar Cascade Classifier Dan Convolutional Neural Network, Mengembangkan sistem deteksi kantuk pada pengendara roda empat berdasarkan durasi kedip mata secara *real-time* . *Haar Cascade Classifier* digunakan pada tahapan *region of interesting* (ROI) mata dan *convolutional neural network* digunakan untuk mendeteksi keadaan mata terbuka dan tertutup dengan tingkat keberhasilan 100%. Sistem ini setidaknya harus memiliki nilai akurasi rata-rata deteksi kantuk sebesar 93,9%, dan hasil akurasi rata-rata yang didapatkan sebesar 95,4% pada jarak 30-50 cm untuk mendeteksi mata terbuka dan tertutup. Waktu komputasi rata-rata sistem ini adalah 0,1069 detik, dimana

dengan angka tersebut akan mempercepat proses deteksi kantuk. Dengan hasil akurasi tersebut sistem ini cukup akurat untuk mengklasifikasikan pengemudi yang mengantuk (Saputra dkk, 2021).

Pada jurnal yang berjudul Pengolahan Citra Untuk Simulasi Deteksi Kantuk Dengan Metode Cascade Classifier dan Black-White Ratio, Melakukan pengujian dengan mengambil gambar dengan mata terbuka, mata tertutup, dan mata setengah tertutup untuk penelitian mereka dengan judul Sistem Pengolahan Citra untuk Simulasi Deteksi Kantuk Dengan Metode *Cascade Classifier* dan *Black-White Ratio*. Untuk mengetahui mata mengantuk atau tidak, dilakukan pengambilan ketiga gambar tadi dengan *webcam* secara *real time*. Berdasarkan percobaan, sesorang akan dikatakan dalam kondisi mengatuk jika melakukan sebanyak lebih dari 20 kali kedipan dan kedipan akan kurang dari atau sama dengan 20 kali jika seseorang tidak mengantuk. Tetapi, hasil penelitian ini masih dinilai kurang efektif karena pendeteksian orang mengantuk hanya dengan melihat jumlah kedipan mata (Parenreng dkk, 2021).

2.2 Dasar Teori

2.2.1 Kantuk

Rasa kantuk seringkali disebabkan karena seseorang kurang istirahat atau kurang tidur. Orang dewasa membutuhkan setidaknya 8 jam setiap malam pada waktu tidurnya untuk mencapai kinerja optimal. Kurangnya waktu tidur dapat menyebabkan kantuk yang berdampak pada ketidakseimbangan dalam tubuh sehingga kewaspadaan sesorang akan berkurang. Dalam kondisi mengantuk, seseorang akan mengalami *microsleep* atau mata akan terpejam sekitar 0,5 detik atau lebih. Dampak buruk yang disebabkan oleh kantuk adalah turunnya performa tiap manusia serta perubahan perilaku dan *mood.* (Andre dkk, 2020).

2.2.2 *Jetson nano*

NVIDIA *Jetson Nano* adalah kit pengembangan kecerdasan buatan atau *Artificial Intelligence* (*AI*) modern dengan kemampuan yang luar biasa dan dapat digunakan untuk menjalankan berbagai muatan *AI*. Untuk menjalankan kerangka

kerja AI dan pemodelan pada aplikasi yang menggunakan pengenalan gambar, deteksi objek, pengenalan ucapan, segmentasi, dan banyak kerangka kerja AI lainnya, pengembang dapat menggunakan NVIDIA Jetson Nano Developer Kit. Penggunaan micro USB sebagai tenaga dalam menjalankan NVIDIA Jetson Nano Developer Kit serta dilengkapi oleh beberapa pin I/O, dari GPIO hingga CSI. NVIDIA Jetson Nano Developer Kit sangat memudahkan pengembang untuk menghubungkan berbagai macam sensor dalam mengembangkan aplikasi AI karena jumlah pin I/O yang cukup banyak. Dengan adanya NVIDIA JetPack pada NVIDIA Jetson Nano Developer Kit sangat mendukung kinerja dari NVIDIA Jetson Nano Developer Kit ini karena NVIDIA JetPack sudah mencakup Linux OS, Paket Dukungan Board (BSP), cuDNN, NVDIA CUDA, TensorRT untuk deep learning, komputasi GPU, dan computer vision. (Fredianto, 2018).

Gambar 2. 1 Perangkat *Jetson nano*

Pada penelitian ini NVIDIA *Jetson Nano* akan digunakan sebagai komputer mini untuk pengolahan data. NVIDIA *Jetson Nano* ini berbentuk dan berukuran kecil, namun konsumsi daya alat ini hanya 5W dan itu sudah cukup mampu menjalankan beban kerja *AI* modern.

2.2.3 OpenCV

Dalam menggunakan OpenCV pengguna tidak harus mengungkapkan kode sumbernya karena penggunaan OpenCV yang diterbitkan oleh lisensi BSD permisif ini sudah lebih bebas dari GPL, kebebasan yang ditawarkan yaitu kebebasan penuh

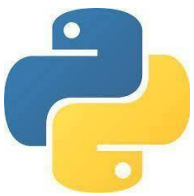
untuk penggunaan secara komersil. Dalam mendukung bahasa pemrograman OpenCV sudah memiliki antarmuka yang mendukung bahasa pemograman C, C++, Java, dan Python serta termasuk dalam sistem operasi Linux, Windows, Android, iOS, dan Mac OS. (Zein, 2018).

Gambar 2. 2 OpenCV

OpenCV pada penelitian ini didisain untuk efisiensi dalam komputasi dan difokuskan pada aplikasi real-time.

2.2.4 Python

Python adalah salah satu bahasa pemograman yang dapat berjalan di berbagai platform dan bersifat terbuka atau opensource. Python dapat digunakan dalam membuat aplikasi dan perintah komputer, serta bisa digunakan untuk melakukan analisis data, bahasa pemograman tingkat tinggi, dan produk yang free. Sering dilakukan reprentasi vektor dan matriks dan operasi terhadapnya, di dalam pemrograman computer vision (Zein, 2018)



Gambar 2. 3 Python

Python berperan penting dalam penelitian ini. Mengumpulkan sekumpulan perintah yang dikelompokkan menjadi satu. Fungsi yang telah dibuat nantinya akan bisa dipanggil berulang kali. Tentu saja fungsi ini dapat mengefisienkan penulisan program, terutama program sistem deteksi kantuk ini.

2.2.5 Pytorch

PyTorch adalah *library* yang dibuat untuk memungkinkan penelitian cepat tentang model pembelajaran mesin. Itu pada sejumlah proyek, terutama Lua Torch, Chainer, dan HIPS Autograd, dan menyediakan lingkungan kinerja tinggi dengan akses mudah ke otomatis *diferensiasi* model yang dijalankan pada perangkat yang berbeda (CPU dan GPU). Untuk memudahkan pembuatan *prototipe*, tidak seperti di banyak kerangka kerja pembelajaran mendalam lainnya PyTorch tidak mengikuti pendekatan simbolis, tetapi berfokus pada diferensiasi imperatif murni program, dengan fokus pada ekstensibilitas dan overhead yang rendah. Perhatikan bahwa pracetak ini adalah draf bagian tertentu dari makalah mendatang yang mencakup semua fitur PyTorch (Paszke dkk, 2017)

Peran Pytorch pada pembuatan sistem ini adalah sebagai framework. Berbeda dengan ndarray NumPy, Tensor yang dinilai cukup mirip dengan ndarray NumPy dapat berjalan di GPU untuk mempercepat komputasi. Untuk memanipulasi dan menyimpan input dan output model serta parameter model dipakai tipe data inti PyTorch yang serupa dengan array multidimensi

2.2.6 Roboflow

Roboflow adalah *platform* yang dirancang khusus untuk membantu *AI engineer* untuk memproses *dataset* untuk pengaplikasian proyek *computer vision*.

Gambar 2. 4 *Roboflow*

Dalam menjalankan proyek computer vision, pengembang computer vision biasanya menggunakan Roboflow karena menyediakan layanan yang dapat membantu dalam menjalankan proyek computer vision. Ketika mengoperasikan Roboflow langkah pertama yang perlu dilakukan yaitu mengupload dataset dimana dataset tersebut diunggah pada platfrom roboflow kemudian data tersebut akan diorganisir agar tidak menyulitkan dalam melakukan anotasi atau pemberian bouding box. Selanjutnya data yang sudah teroganisir akan menjadi imput untuk latih model. Model yang telah dilatih dapat dilakukan deployment dan dapat dilihat hasil deteksi dari model yang sudah dilatih (Levina, 2021).

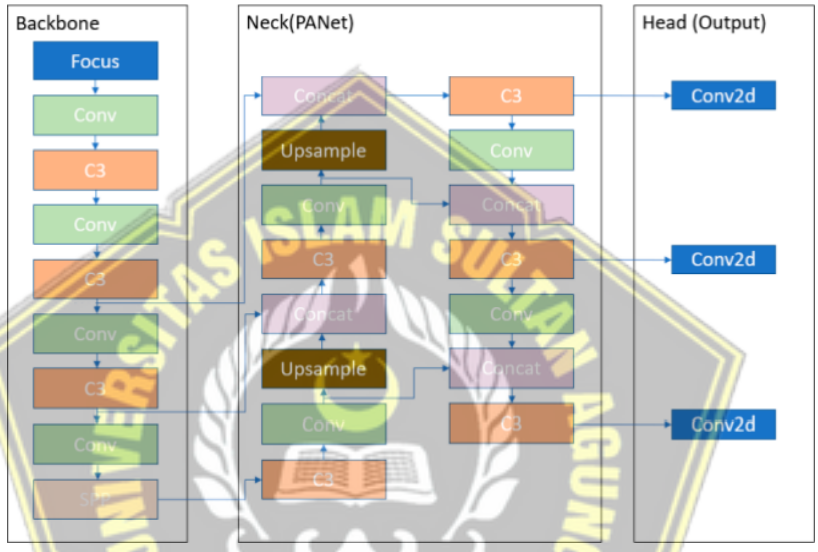
2.2.7 CNN

Terdapat beberapa metode Deep Learning untuk computer vision salah satunya Convolutional neural network (CNN). Karena Convolutional neural network ini terdiri dari beberapa lapisan tersembunyi sehingga pada setiap lapisan Convolutional neural network terjadi komputasi matematika pada imput dan menghasilkan output. Hasil dari output tadi akan dijadikan untuk lapisan selanjutnya. Metode yang sering dihasilkan dari Convolutional neural network ini yaitu berupa klasifikasi gambar. Sebagai contoh pembuatan kotak pembatas (bounding boxes) pada sebuah gambar yang dikerjakan oleh R-CNN (Region with CNN) serta adanya pengklasifikasian pada kotak tersebut. Cara kerja You Only Look Once (YOLO) berbeda dengan cara kerja sliding window atau R-CNN. You Only Look Once (YOLO) bekerja berbeda dengan metode sebelumnya dan merupakan metode yang bekerja dengan cepat (Khairunnas dkk, 2021).

2.2.8 You Only Look Once

YOLO merupakan metode deep learning yang ditujukan untuk mendeteksi objek dan menyatukan komponen-komponen objek deteksi menjadi satu jaringan saraf di seluruh gambar. Pada April 2020 model pendeteksi objek generasi kelima yang dirilis yaitu YOLOv5. Arsitektur pada model ini pada dasarnya tidak jauh berbeda dengan generasi YOLO sebelumnya. Yang membedakan YOLOv5 dengan YOLO sebelumnya yaitu YOLOv5 ditulis dalam bahasa pemrograman Python sedangkan YOLO sebelumnya menggunakan bahasa C. Versi terbaru saat ini dari

pengembangan YOLO adalah YOLO-v5, berdasarkan YOLO-v4. Pada YOLO-v5 dengan kecepatan tercepat mencapai 140 Frames per Second (FPS), dimana kecepatan pemrosesan meningkat dengan drastic. YOLO-v5 berukuran 90% lebih kecil dari YOLO-v4, sehingga memungkinkan YOLO-v5 dapat di-deploy ke perangkat yang disematkan. Peningkatan selanjutnya pada YOLOv5 yaitu tingkat akurasi yang lebih tinggi dan kemampuan yang lebih baik untuk mengenali benda-benda kecil.



Gambar 2. 5 Arsitektur YOLOv5

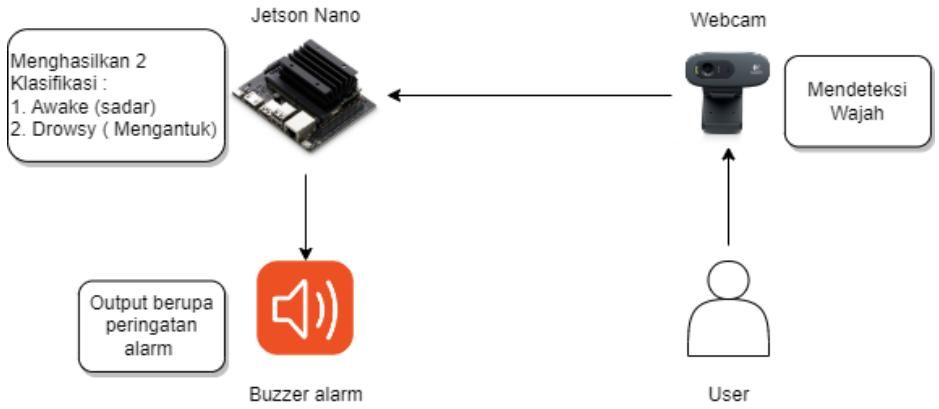
Lapisan konvolusi ditunjukkan oleh Conv. C3 merupakan modul yang dialiri oleh berbagai hambatan dan terdiri dari tiga lapisan konvolusi. Untuk menghilangkan batasan ukuran tetap pada jaringan digunakan lapisan pooling, hal ini disebut Spatial Pyramid Pooling (SPP). Pada node terdekat di upsampling fusi lapisan sebelumnya terjadi penggunaan upsample. Ketika mengiris layer sebelumnya dibutuhkan concat yang merupakan lapisan pengiris Lapisan pooling yang digunakan untuk menghilangkan batasan ukuran tetap pada jaringan disebut Spatial Pyramid Pooling (SPP). Upsample digunakan dalam upsampling fusi lapisan sebelumnya di node terdekat. Concat merupakan lapisan pengiris yang digunakan untuk mengiris layer sebelumnya. Pada head dari jaringan terjadi penggunaan modul deteksi objek atau dapat disebut dengan Tiga Conv2d terakhir (Levina, 2021).

BAB III
METODE PENELITIAN

3.1 Deskripsi Sistem

Program yang nantinya akan dibuat penulis yaitu sistem berupa aplikasi yang dibuat dengan bahasa pemrograman python kemudian dijalankan pada perangkat komputer kecil adalah *Jetson nano* Nvidia dengan bantuan perangkat keras menggunakan kamera sebagai alat mendeteksi wajah yang kemudian di proses pengklasifikasian nya menggunakan metode Yolov5. sistem ini adalah pengimplementasian *deep learning* yaitu pengenalan wajah, metode yang digunakan dalam pengidentifikasian atau memverifikasikan identitas seseorang menggunakan wajah tersebut.

Kamera disini digunakan untuk merekam wajah *user* secara *realtime*, Lalu pendeteksian wajah *user* dari webcam secara realtime, kemudian di proses menggunakan *computer* kecil *jetson nano* untuk pengelolaan informasi wajah user tersebut memvalidasi sebelum sistem melakukan klasifikasi apakah wajah *user* ini sedang mengantuk, atau tidak mengantuk. menampilkan nilai akurasi dari hasil deteksi, dan alarm berbunyi jika terdeteksi mengantuk sehingga sistem ini nantinya dapat mengurangi kecelakaan yang disebabkan mengantuk saat mengemudikan kendaraan roda empat. Lebih jelasnya dapat dilihat sesuai dengan diagram konteks.

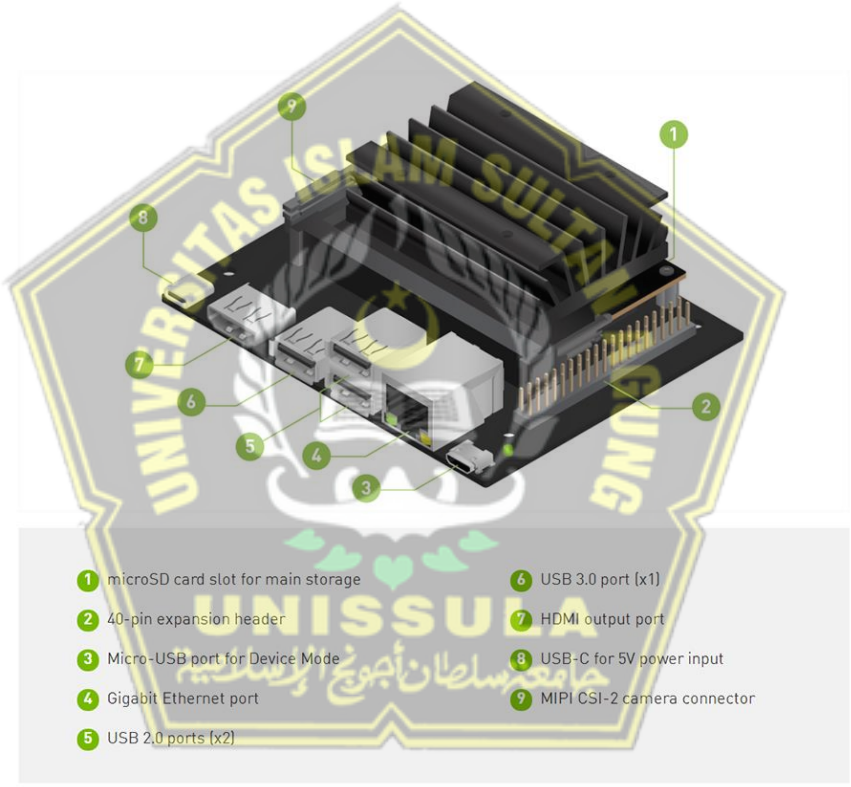


Gambar 3. 1 Perancangan *hardware*

3.2 Kebutuhan Sistem

Pada implementasi sistem deteksi kantuk ini dibutuhkan perangkat kerja dan perangkat lunak sebagai berikut :

3.2.1 *Jetson nano* Developer Kit 2GB



Gambar 3. 2 Spesifikasi *Nvidia Jetson nano Developer Kit 2GB*

Perangkat yang digunakan pada penelitian ini yakni *Jetson nano* Nvidia sebagai mikrokontroler. *Jetson Nano Developer Kit* adalah mikrokontroler yang nantinya dimanfaatkan untuk menjalankan deteksi kantuk. Kemudian *Jetson nano* dijalankan yang nantinya dapat menjalankan sistem pengklasifikasian data *input real time* dari Webcam Logitech C270. Pada perangkat *Jetson nano* yang dipergunakan mempunyai spesifikasi yang ditunjukan pada Tabel 3.1 berikut ini :

Tabel 3. 1 Spesifikasi Jetson Nvidia Nano

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

3.2.2 Webcam Logitech C270

Webcam Logitech C270 memperoleh data yang berupa wajah *real time* yang dihasilkan dari merekam wajah secara *real time* yang nantinya akan diproses dan diklasifikasikan oleh *Jetson nano* Nvidia.

Gambar 3. 3 *Webcam Logitech C270*

Pada Webcam Logitech C270 memiliki spesifikasi yang ditunjukan pada Tabel 3.2 berikut ini :

Tabel 3. 2 Spesifikasi Webcam

Resolusi	1080px720p
Kecepatan	30 frame rate per second
Permukaan Penglihatan	55°
Fokus Tetap	Lensa Plastik
Mikrofon Mono Omnidirectional	Terintegrasi
Jenis Teknologi	Teknologi Peredam Kebisingan
Koreksi Cahaya	Otomatis
Klip Pemasangan	Universal
Pasang dan Mainkan melalui	USB 2.0 Tipe-A
Kompatibel	Windows, macOS, dan Chrome

3.2.3 *Library Pytorch*

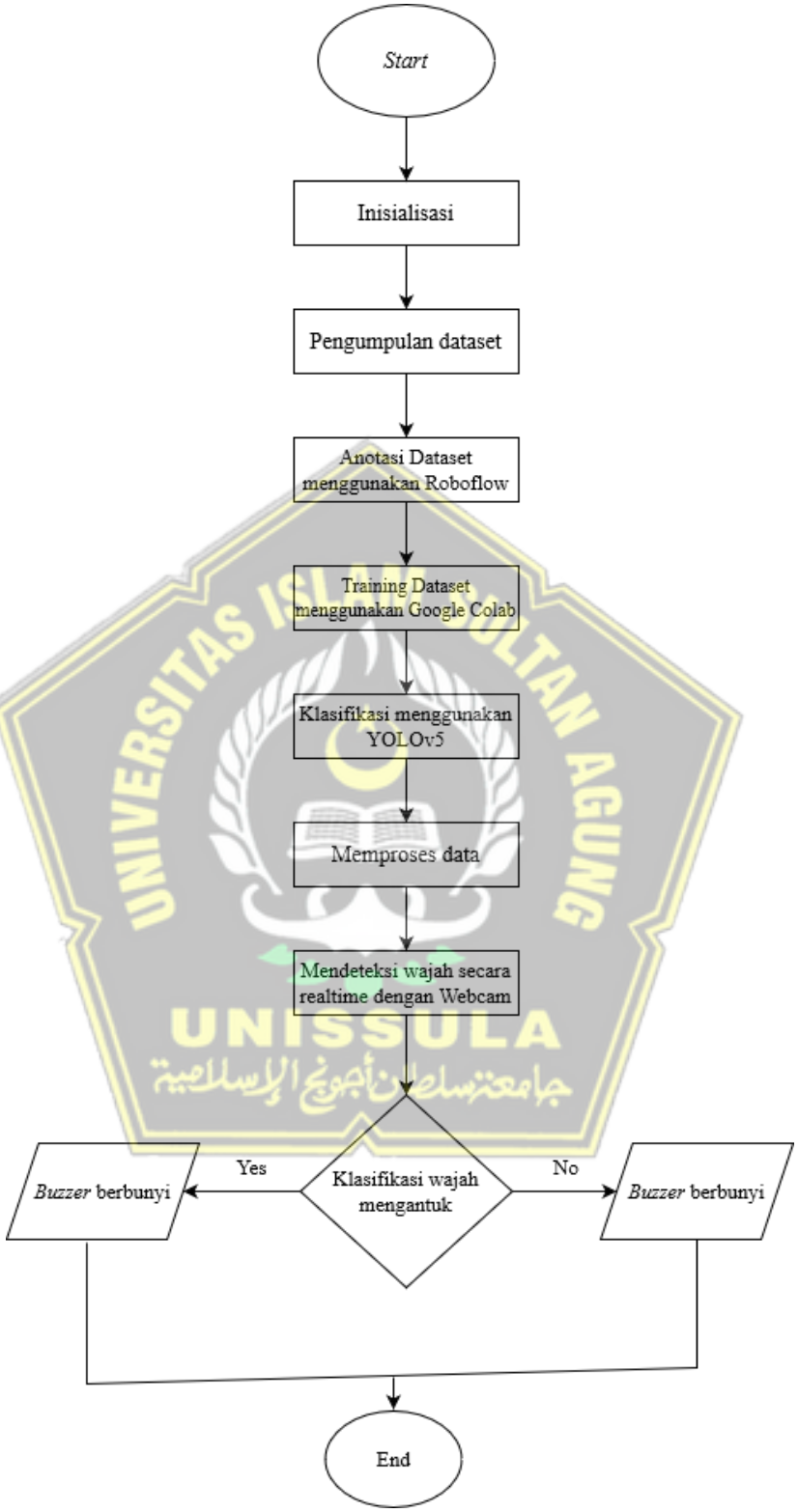
Gambar 3. 4 *Pytorch*

PyTorch adalah *library* tensor yang dioptimalkan untuk deep learning menggunakan GPU dan CPU. Fitur yang dijelaskan dalam dokumentasi ini diklasifikasikan berdasarkan status rilis:

1. Stabil: Fitur-fitur ini akan dipertahankan dalam jangka panjang dan umumnya tidak boleh ada batasan atau kesenjangan kinerja yang besar dalam dokumentasi. Kami juga berharap untuk mempertahankan kompatibilitas mundur (walaupun perubahan yang merusak dapat terjadi dan pemberitahuan akan diberikan satu rilis sebelumnya).
2. Beta: Fitur-fitur ini ditandai sebagai Beta karena API dapat berubah berdasarkan umpan balik pengguna, karena kinerja perlu ditingkatkan, atau karena cakupan antar operator belum lengkap. Untuk fitur Beta, kami berkomitmen untuk melihat fitur tersebut hingga klasifikasi Stabil. Namun, kami tidak berkomitmen untuk kompatibilitas ke belakang.
3. Prototipe: Fitur-fitur ini biasanya tidak tersedia sebagai bagian dari distribusi biner seperti *PyPI* atau *Conda*, kecuali terkadang di belakang tanda run-time, dan berada pada tahap awal untuk umpan balik dan pengujian.

3.3 Metode Penelitian

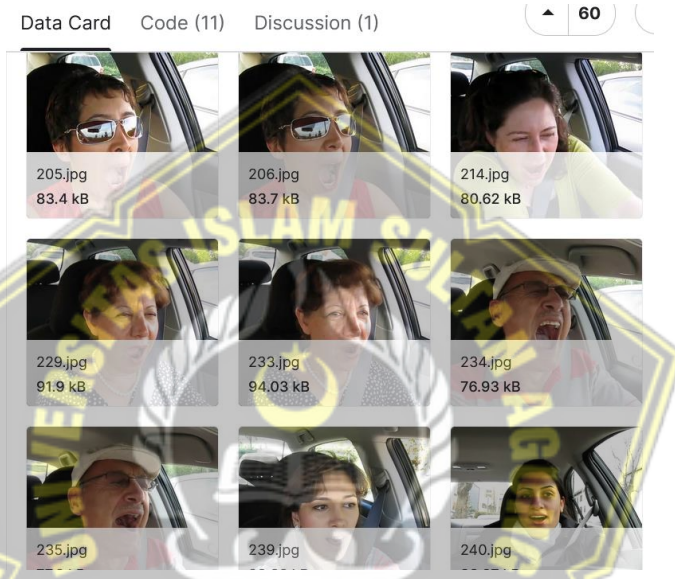
Pada sistem yamg dirancang menerapkan metode YOLOv5 dari *library Pytorch* untuk mendeteksi wajah dan pengenalan wajah menggunakan OpenCV. Pada tahap pengklasifikasian wajah akan dilakukan pembagian menjadi tiga tahap yakni pertama tahap mengumpulkan *dataset* yang ada pada google *dataset roboflow universe*, tahap kedua adalah menganotasikan dataset yang terdapat kumpulan gambar-gambar wajah dengan menggunakan *Roboflow*, dan terakhir *training dataset* gambar wajah yang menggunakan *library pytorch* :



Gambar 3. 5 *Flowchart* perancangan alur sistem

3.3.1 Pengumpulan Data

Adanya pembuatan program pada sistem ini diawali dengan mengumpulkan banyaknya dataset yang akan di t*raining*, semakin ditemukan banyaknya dataset yang digunakan maka akan menghasilkan semakin lebih baik juga hasil yang akan diperoleh dari sistem-sistem pendeteksi masker. Dataset yang ada menggunakan data dari sumber *Google Dataset Search* yaitu *Kaggle* dataset lebih dari 100 gambar yang memiliki 2 kelas *Awake* dan *Drowsy* yang belum diberikan *label.*



Gambar 3. 6 Kumpulan Dataset

Setelah itu, 70% dataset digunakan untuk data training, dan 20% dataset akan digunakan sebagai data validating, dan 10 % digunakan untuk data testing.

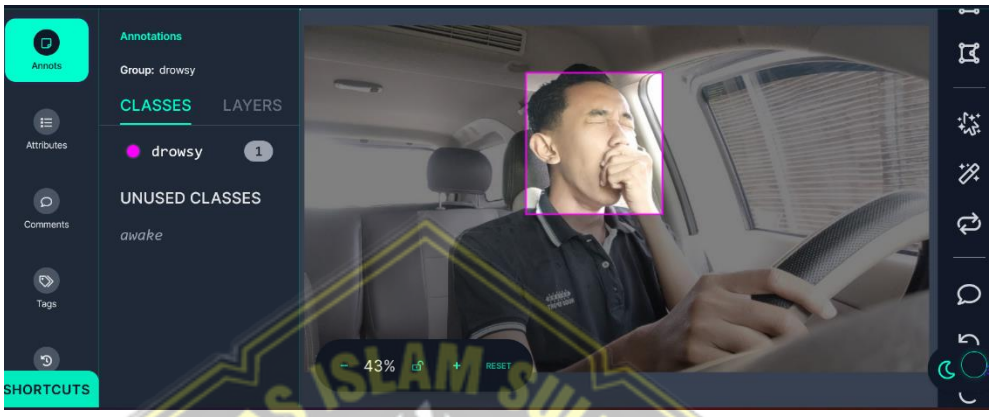
Tabel 3. 3 Pembagian data training, testing dan validation

Nama Data	Jumlah (%)	Jumlah Angka
Data Training	70%	70
Data Validation	20 %	20
Data Testing	10%	10

Total data : 100

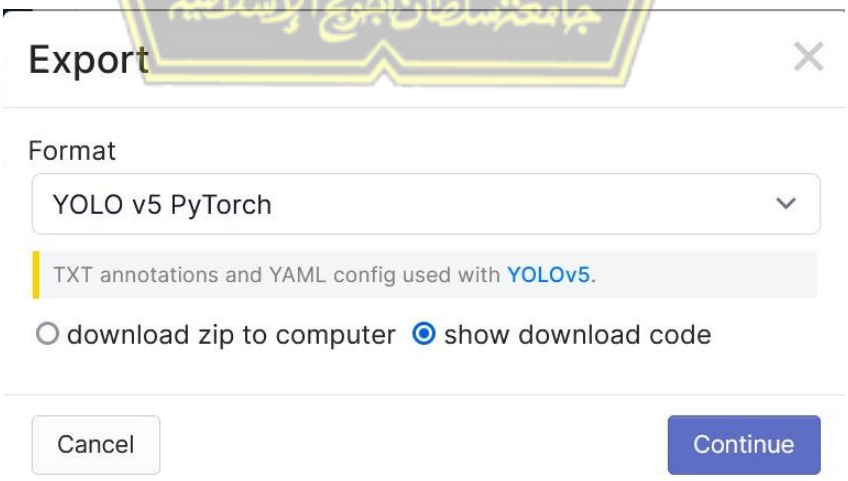
3.3.2 Anotasi Dataset (*labelling)*

Pada tahap ini adalah proses melakukan anotasi atau pelabelan pada dataset untuk memberikan pembeda antara gambar wajah mengantuk (*drowsy*) dan sadar (*awake*).



gambar 3. 7 Proses Anotasi Dataset

. Seperangkat label gambar yang ada, menggunakan platform *Roboflow* yang dapat berfungsi untuk mengolah data-data *raw* yang akan menjadi data siap untuk diuji coba atau *training* lalu dapat di *export* dengan bentuk dan ukuran yang disesuaikan kebutuhan. Pada ketentuan penelitian ini bentuk dan ukuran pada gambar yaitu yolov5 pytorch.



Gambar 3. 8 Pilih Format YOLOv5 Pytorch

Setelah Proses anotasi dan hasil dataset dengan format tersebut maka dapat melakukan tahap yang selanjutnya yaitu trainig dataset menggunakanan *google colab.*

3.3.3 Training Dataset Menggunakan *Google Colab*

Tahap ketiga yang akan dilakukan dalam proses train yakni pada gambaran yang telah diberi label dengan format sesuai yang dibutuhkan yaitu *Yolov5 Pytorch* untuk menggunakan Yolov5 diperlukan *library pytorch* dapat menjadi sistem yang genius dengan buatan yang bisa mendeteksi adanya suatu gambaran objek yakni wajah. Pada tahapan ini pun diperlukan percobaan sebuah *training* untuk bisa melatih *dataset* yang akan digunakan guna kepentingan dalam penelitian deteksi kantuk. Pada tahap ini, telah sampai pada pengumpulan data dengan 100 dataset yang terdiri dari wajah-wajah yang mengantuk (*drowsy*), dan sadar (*awake*) yang diberikan label dan kelas sesuai gambarnya akan diproses pada tahap *training dataset* menggunakan *google colab*. Pada proses training akan menghasilkan file result yaitu weights yang nantinya akan digunakan YOLOv5 untuk klasifikasi.

3.3.4 Mendeteksi Wajah dengan Webcam

Pada proses tahapan ini, kamera (Logitech C270) berperan sebagai alat pendeteksi guna menangkap wajah *user* yang akan terhubung dengan alat perangkat dari komputer kecil yakni *Jetson nano* Nvidia yang akan dilanjutkan dengan data pendeteksi wajah untuk selanjutnya di proses lebih mendalam.

3.3.5 Klasifikasi Wajah

Proses terakhir yaitu sistem mampu mengklasifikasi wajah yang merupakan proses pengujian pendeteksi wajah secara *realtime* dimana hasil tersebut didapatkan pada dataset ratusan gambar. Tahap klasifikasi deteksi pada wajah terdiri dari 2 yaitu *Drowsy* dan *Awake.* Jika sistem mendeteksi wajah pengemudi mengantuk (*drowsy*) akan memberikan *ouput* buzzer berbunyi sebagai tanda alarm peringatan.

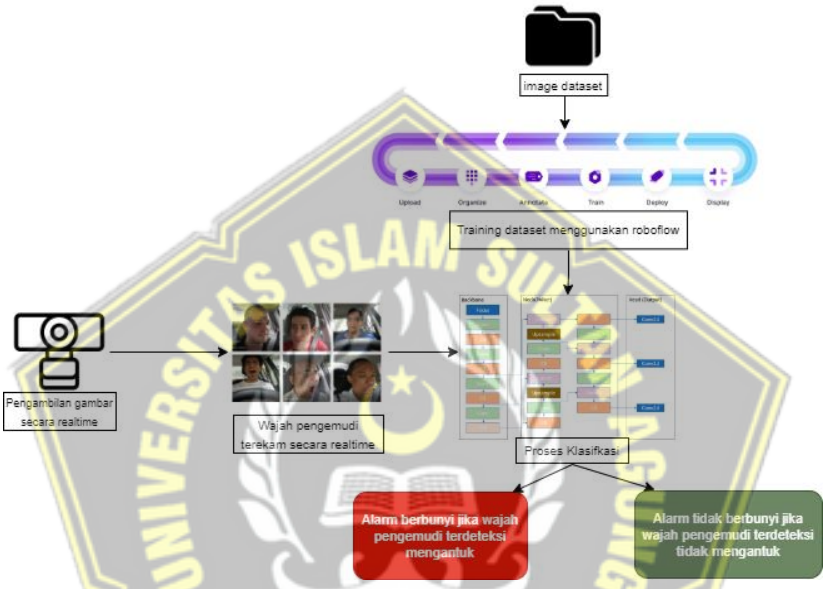
3.3.6 Memproses Data

Pada tahap ini wajah telah terdeteksi dan sistem akan memproses agar dapat

melakukan proses klasifikasi wajah dari dataset yang telah dikumpulkan, dengan melatih dataset dari gambar "*drowsy*", dan "awake" Untuk training dataset menggunakan *library pytorch*.

3.3.7 Pembuatan sistem pada *Jetson nano*

Setelah model siap digunakan, selanjutnya proses pembuatan sistem pada *jetson nano*. Hasil dan alur penggunaan sistem dapat dilihat pada gambar 3.9.



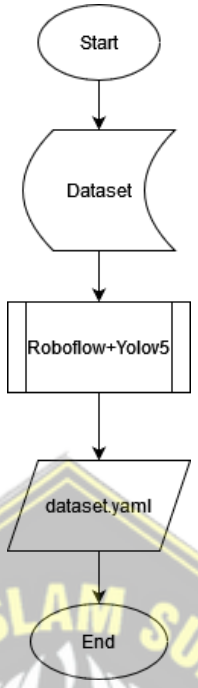
Gambar 3. 9 Deskripsi Sistem

Gambar 3.9 menunjukkan alur penggunaan sistem deteksi kantuk berbasis *jetson nano*, dimana pengguna akan terekam wajahnya secata realtime saat mengemudi yang dapat diambil langsung dari kamera *webcam*, kemudian wajah akan dideteksi oleh sistem dan masuk ke proses klasifikasi oleh model, setelah itu sistem akan menampilkan hasil deteksi atau klasifikasi yaitu *drowsy* atau *awake.*

3.4 Desain Sistem

3.4.1 Flowchart Sistem

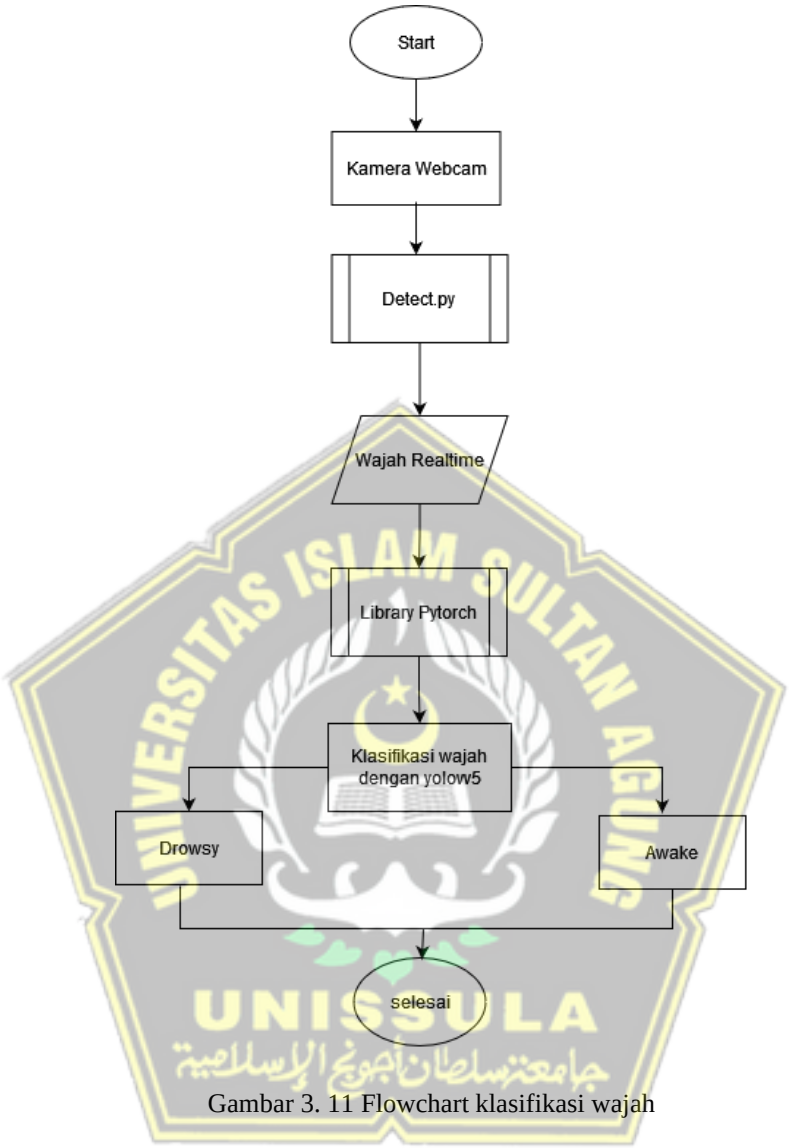
a. Flowchart training data



gambar 3. 10 Flowchart Training Data

Ditunjukkannya gambar 3.10 adalah proses yang kedua dari sistem deteksi kantuk. Dibutuhkan dataset berupa gambar wajah yang sebelumnya telah disimpan dari proses pengambilan data pada proses ini. Dari *dataset* dengan proses label anotasinya menggunakan roboflow lalu diolah dengan algoritma YOLOv5 dari *library pytorch*. Proses training nantinya akan menghasilkan histogram-histogram berupa matrik yang mewakili gambar dari *dataset*.

b. *Flowchart* pengklasifikasian wajah

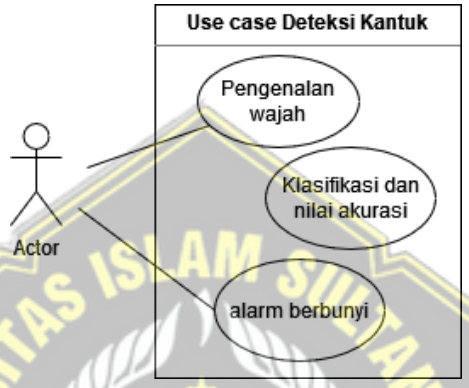


Gambar 3. 11 Flowchart klasifikasi wajah

Proses pada gambar 3.11 adalah pengenalan wajah yang diawali saat kamera terkoneksi ke jetson sebagai inisialisasi mendeteksi wajah. Dengan menggunakan algoritma YOLOv5 di peroleh objek wajah yang kemudian diolah lagi menggunakan framework *library pytorch.*

3.4.2 *Use Case Diagram*

Berikut adalah diagram pengguna yang tedapat pada sistem penelitian ini ditunjukkan pada gambar 3.13 yakni hanya user, dimana hal tersebut dapat dilakukan user untuk melakukan pendeteksi wajahnya di depan kamera terdeteksi secara otomatis dan hasil pun dapat dilihat yang dimana hasil tersebut berupa klasifikasi dan nilai akurasi.



gambar 3. 12 *Usecase diagram*

BAB IV
HASIL DAN ANALISIS PENELITIAN

4.1 Implementasi Sistem

4.1.1 Menyiapkan Kebutuhan *Jetson nano*

Sebelum ketahap pengerjaan program dan *training* data, langkah awal untuk menggunakan *jetson nano* membutuhkan tahap instalasi agar perangkat tersebut bisa digunakan dalam pembuatan sistem. Berikut tahap-tahap menyiapkan kebutuhan pada *jetson nano* :

1. *Jetson nano Developer Kit 2GB*

Gambar 4. 1 *Jetson nano Developer Kit Box*

Perlengkapan yang ada di dalam Box *Jetson nano* Developer Kit Anda meliputi Modul NVIDIA Jetson dan Manual Book. Kemudian Item tidak Termasuk dalam box dan perlu disiapkan yaitu Kartu microSD, Keyboard dan mouse USB, Tampilan komputer (HDMI atau DP), Catu daya micro-USB.

2. *Power Supply*

Jetson nano memerlukan daya pada kit pengembang dengan catu daya berkualitas baik yang dapat menghasilkan 5V⎓2A di port Micro-USB kit pengembang. Tidak semua catu daya yang menjanjikan "5V⎓2A" benar-

benar akan melakukan ini.

Sebagai contoh catu daya yang baik, NVIDIA telah memvalidasi Catu Daya Switching 5V 2.5A Adafruit dengan Kabel MicroUSB 20AWG (GEO151UB-6025).

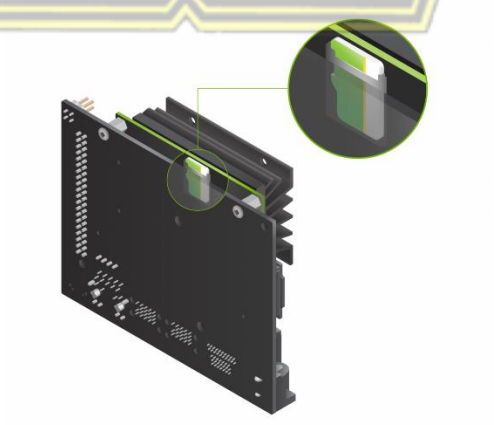
3. *Write Image* pada *microSD Card*

Untuk menyiapkan kartu microSD, Anda memerlukan komputer dengan koneksi Internet dan kemampuan untuk *read* dan *write* SD card, baik melalui slot atau adaptor SD card bawaan.

a. Unduh image file SD card Kit Pengembang *Jetson nano*, dan catat di mana gambar itu disimpan di komputer.
b. Write image ke kartu microSD sesuai dengan sistem operasi komputer Anda: Windows, macOS, atau Linux.

Langkah-langkah Pengaturan :

a. Buka lipatan dudukan kertas dan tempatkan di dalam *Developer Kit Box*.
b. Masukkan kartu microSD (dengan gambar sistem yang sudah tertulis di dalamnya) ke dalam slot di bagian bawah modul *Jetson nano* seperti gambar dibawah ini.



Gambar 4. 2 *Insert SDcard*

c. Nyalakan layar komputer Anda dan sambungkan.
d. Sambungkan keyboard dan mouse USB.
e. Sambungkan catu daya Micro-USB Anda .*Developer Kit* akan hidup dan boot secara otomatis

4. Boot Pertama

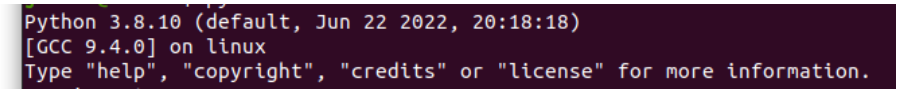
LED hijau di sebelah konektor Micro-USB akan menyala segera setelah kit pengembang menyala. Saat Anda melakukan booting pertama kali, kit pengembang akan membawa Anda melalui beberapa penyiapan awal, termasuk:

a. Tinjau dan terima EULA perangkat lunak NVIDIA Jetson
b. Pilih bahasa sistem, tata letak keyboard, dan zona waktu
c. Buat nama pengguna, kata sandi, dan nama komputer
d. Pilih ukuran partisi APP—disarankan untuk menggunakan ukuran maksimal yang disarankan.

Setelah proses boot pertama selesai, layar akan menampilkan halaman utama dekstop ubuntu seperti yang ditunjukkan pada gambar 4.3.

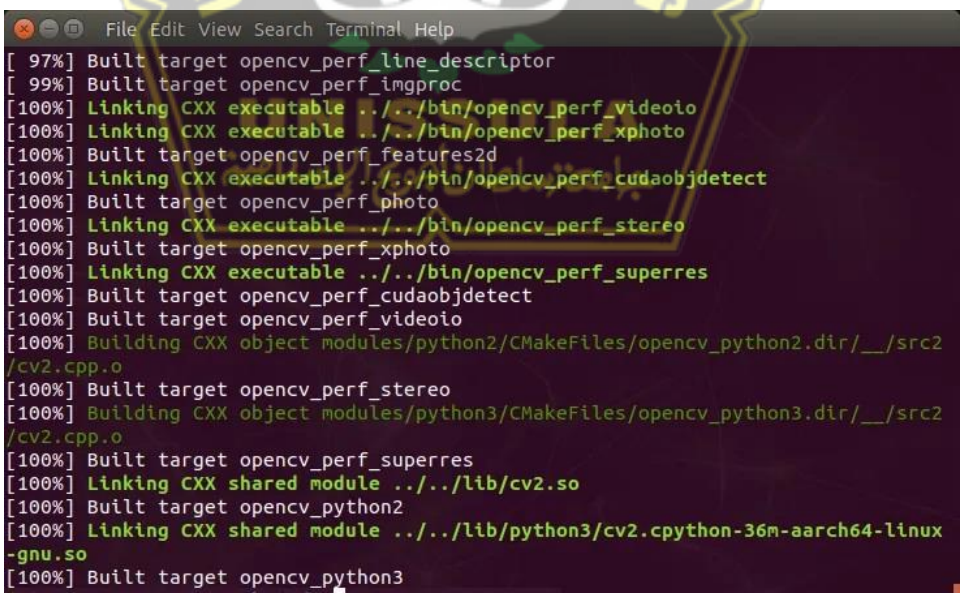
Gambar 4. 3 Tampilan awal ubuntu pada jetson nano

5. Kegunaan menginstall python pada penelitian ini adalah sebagai pembagian dataset yang digunakan sebagai tahap *Training Custom Object*. Versi pyhton yang digunakan minimal menggunakan versi 3.8 untuk mendukung keperluan *depencencies requirements* YOLOv5 nantinya.



Gambar 4. 4 Python telah terinstall

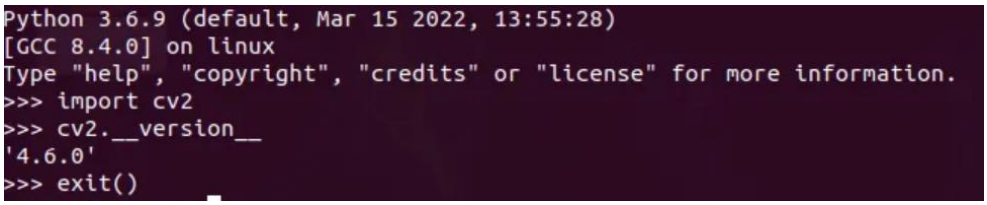
6. Instalasi Opencv CUDA, digunakan untuk mendukung kinerja GPU dengan 128 core yang dimiliki *jetson nano developer kit 2gb*. Proses komputasi secara bersamaan CPU dan GPU mampu dilakukan oleh CUDA dengan arstitekturnya tersebut. Daripada hanya memakai CPU, GPU yang dilengkapi CUDA jauh lebik baik dalam melakukan proses simulasi grafis. Gambar 4.5 adalah proses *compile* cuda



Gambar 4. 5 Proses *compile* CUDA

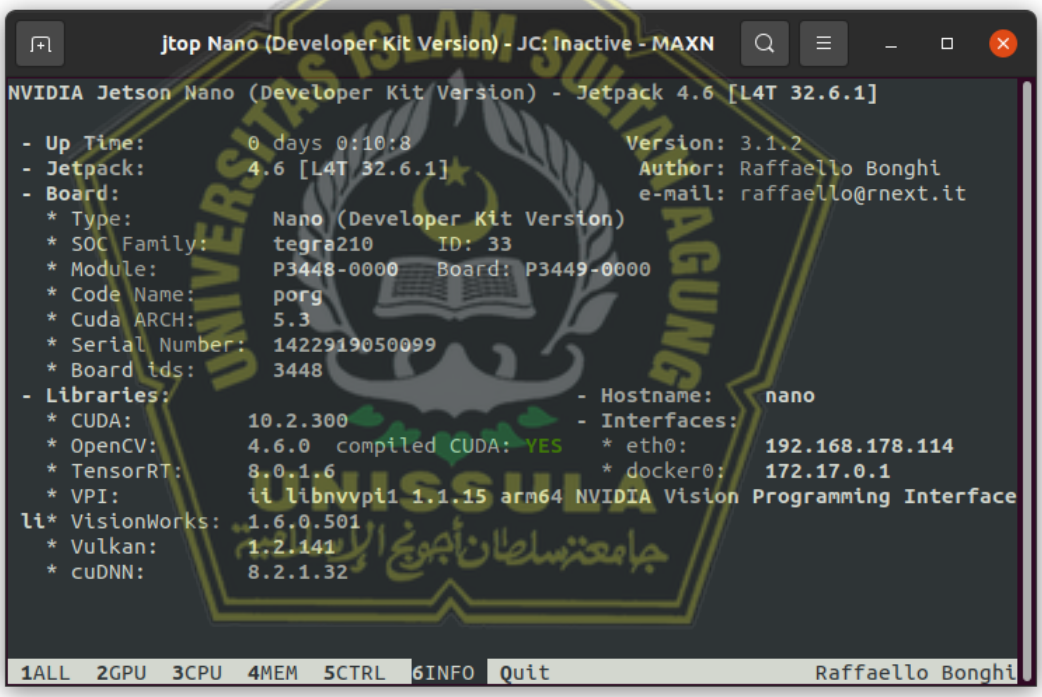
Setelah selesai proses compile opencv cuda, kemudian cek opencv untuk

memastikan opencv sudah terinstal, seperti yang ditunjukkan pada gambar 4.6



Gambar 4. 6 Opencv telah terinstall

Jika sudah benar terinstal , kemudian cek versi CUDA dengan mengetikkan command *jtop* pada terminal. Lalu akan muncul informasi seperti pada gambar 4.7.



Gambar 4. 7 Informasi library yang sudah terinstall

Dari gambar 4.7 menunjukkan informasi bahwa opencv CUDA sudah ter-*compiled* dengan versi 10.2. itu artinya CUDA sudah siap mendukung untuk menggerakkan kinerja GPU pada *jetson nano*.

7. Pytorch pada pada *yolov5* disini berperan sebagai framework dalam berbagai versi, yaitu small, medium, large, dan Xlarge yang masing-masing memiliki

spesifikasi dan kegunaannya masing-masing. Versi pytorch yang diperlukan menyesuiakan spesifikasi perangkat yang digunakan. Untuk perangkat *jetson nano* dengan *operating system ubuntu* 20.04 membutuhkan *torch* dengan *versi* 1.12.0 dan *torchvision* dengan versi 0.13.0 seperti yang ditunjukkan pada gambar 4.8.

Gambar 4. 8 Torch dan torchvision telah terinstall

4.1.2 Instalasi Kebutuhan YOLOv5

Setelah semua kebutuhan yang diperlukan pada *jetson nano* sudah terinstall, kemudian langkah yang selanjutnya menginstal YOLOv5 dan requirements yang diperlukan.

1. Konfigurasi CUDA yang bertujuan untuk memanggil library CUDA yang sudah di *compile*. Untuk mengkonfigurasi CUDA dapat dilakukan dengan mengetikkan *command* yang ditunjukkan pada gambar 4.9.

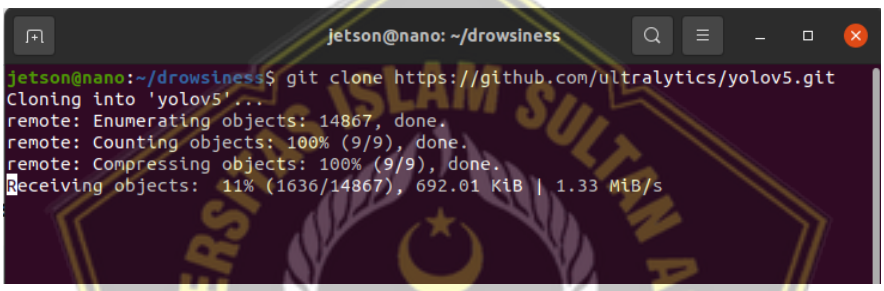
Gambar 4. 9 *command* untuk masuk file bashrc

Command pada gambar 4.9 adalah untuk masuk kedalam text editor yang bernama vim. Lalu menambahkan line seperti yang ditunjukkan pada gambar 4.10 didalam file .bashrc untuk konfigurasi CUDA.

```
##############add line #############
export PATH=/usr/local/cuda/bin${PATH:+:${PATH}}
export
LD_LIBRARY_PATh=/usr/local/cuda/lib64${LD_LIBRARY_PATH:+:${LD_LIBRARY_P
ATH}}
export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
```

Gambar 4. 10 *Add line* untuk konfigurasi CUDA

2. *Clone* repositori YOLOv5 Ultralitys dengan mengetikkan command pada gambar 4.11.



Gambar 4. 11 *Clone* repositori YOLOv5

3. Instalasi requirements yang dibutuhkan YOLOv5

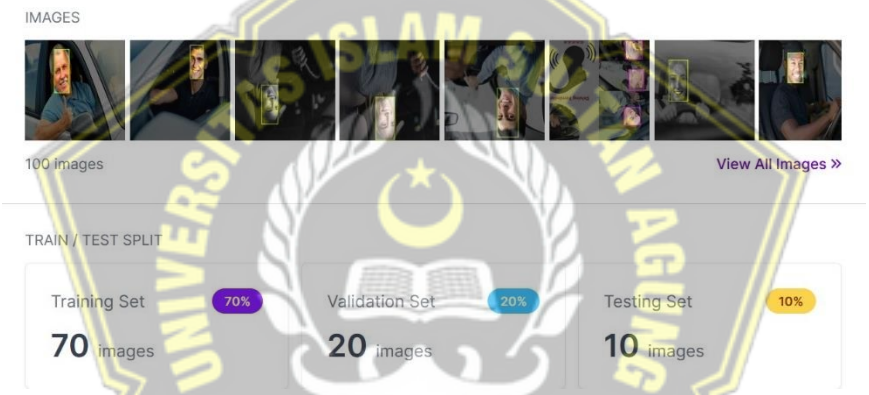
```
# YOLOv5 🚀 requirements
# Usage: pip install -r requirements.txt

# Base ------------------------------------------------------------------------
ipython  # interactive notebook
matplotlib>=3.2.2
numpy>=1.20.3
opencv-python>=4.1.1
Pillow>=7.1.2
psutil  # system resources
PyYAML>=5.3.1
requests>=2.23.0
scipy>=1.4.1
thop>=0.1.1  # FLOPs computation
torch>=1.7.0  # see https://pytorch.org/get-started/locally (recommended)
torchvision>=0.8.1
tqdm>=4.64.0
# protobuf<=3.20.1  # https://github.com/ultralytics/yolov5/issues/8012

# Logging ---------------------------------------------------------------------
tensorboard>=2.4.1
# clearml>=1.2.0
# comet

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

# Export ----------------------------------------------------------------------
# coremltools>=6.0  # CoreML export
# onnx>=1.9.0  # ONNX export
# onnx-simplifier>=0.4.1  # ONNX simplifier
# nvidia-pyindex  # TensorRT export
# nvidia-tensorrt  # TensorRT export
# scikit-learn<=1.1.2  # CoreML quantization
# tensorflow>=2.4.1  # TF exports (-cpu, -aarch64, -macos)
# tensorflowjs>=3.9.0  # TF.js export
```

Plain Text Tab Width: 8 Ln 7, Col 14 INS

Gambar 4. 12 *requirements.txt*

4.1.3 Anotasi Dataset Menggunakan Roboflow

Setelah semua kebutuhan YOLOv5 sudah tersedia, Setelah itu penganotasian disetiap citra tersebut mengikuti ekstensi format YOLOv5 *Pytorch*. Proses anotasi yang pertama harus dilakukan adalah pelabelan yaitu dengan melakukan pemberian *bounding box* sekaligus membuat *class* pada objek disetiap citra sesuai dengan nama klasifikasi. Untuk penelitian ini penulis menggunakan anotasi dari Roboflow dikarenakan *paltform* Roboflow ini mudah dan mendukung untuk anotasi dengan banyak kelas dan mendukung format YOLOv5 Pytorch. Setelah dataset sudah teranotasi, kemudian dataset tersebut dibagi menjadi 3 bagian yaitu *training*, *validation*, dan *test.*

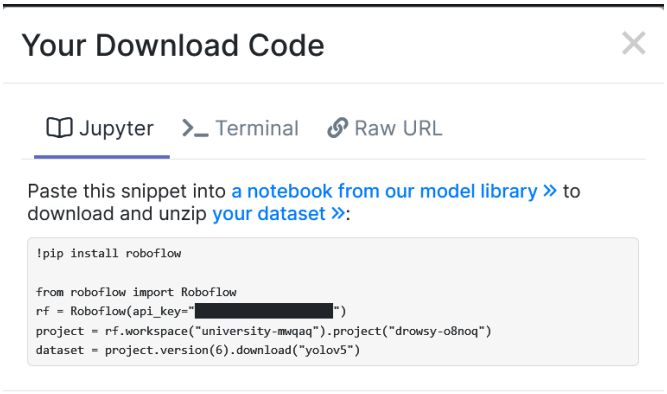


Gambar 4. 13 pembagian *training set*

Komposisi Pembagian Dataset adalah 70% data training, 20% data *validation*, dan 10% data *test*. Dari dataset tersebut menghasilkan file data *train* (data/train.txt), *valid* (data/valid.txt), dan *test* (data/test.txt) dengan format file sesuai kebutuhan YOLOv5 *Pytorch*.

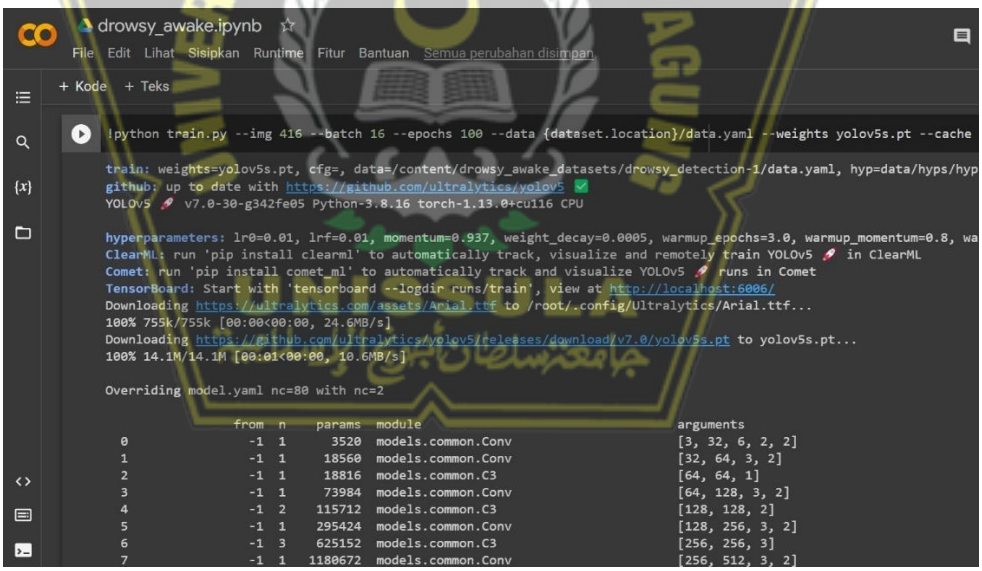
4.1.4 *Training dataset* yang telah dianotasi

Setelah melewati proses anotasi dan labelling , Ekspor dataset pada roboflow dengan format YOLOv5 Pytorch yang ditunjukkan pada bagian bab 3.3.2 setelah itu pilih *show code to download* lalu akan muncul *download code* pada gambar 4.14 :



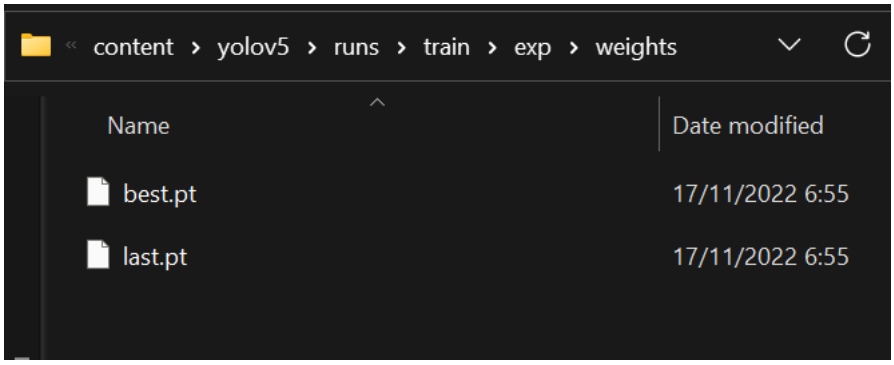
Gambar 4. 14 Kode download file dataset format YOLOv5

Setelah mendapatkan *download code* , selanjutnya memasuki notebook untuk *training model dataset* yang telah di anotasi yaitu *goole colab* untuk mendapatkan bobot atau *weight* dengan format file best.pt. Google colab digunakan untuk melatih dan menguji model untuk berbagai kelas seperti Drowsy, Awake. Dataset tersebut akan di training selama 100 epoch dan mencapai akurasi sekitar 92% .



Gambar 4. 15 Proses *training dataset*

Pada gambar 4.15 adalah proses training dataset menggunakan perintah train.py dengan epoch 100. data.yaml adalah file hasil ekspor pada robloflow dengan ekstensi file yang sesuai dengan yolov5. Bobot atau *weights* yang digunakan untuk training yaitu yolov5s.pt adalah file ekstensi dari framework pytorch. Setelah itu run kernel untuk memulai training dataset



Gambar 4. 16 Hasil *training dataset*

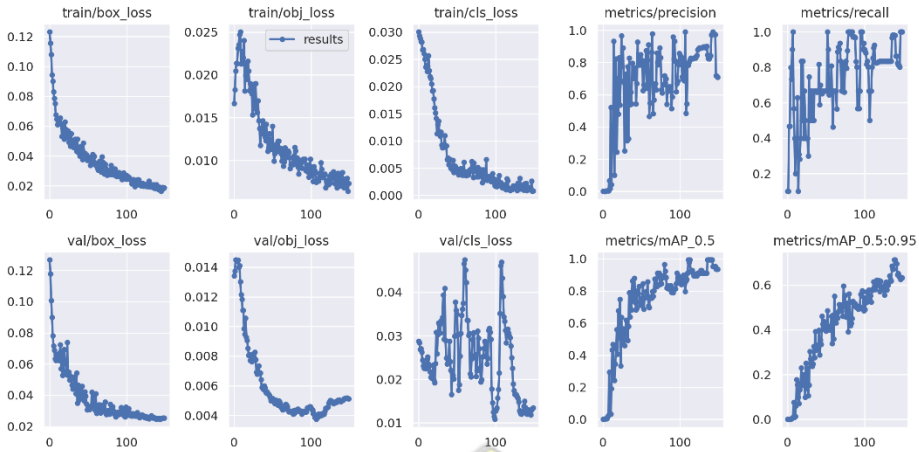
Setelah proses training selesai, gambar diatas merupakan hasil training dataset dengan format best.pt dan last.pt yang akan digunakan YOLOv5 untuk klasifikasi pada image processing.

4.2 Pengujian *Blackbox Testing*

Untuk perancangan sistem telah selesai maka yang selanjutnya adalah melakukan pengujian perangkat dari perangkat yang sudah dipersiapkan. Untuk pengujian pada penelitian ini yaitu menggunakan *blackbox testing*. Pengujian *blackbox* adalah pengujian dengan menguji secara fungsionalitas pada perlencanaan dan perancangan apakah sudah berfungsi dengan baik saat sistem dijalankan. Berikut proses pengujiannya :

4.2.1 Pengujian Perangkat Lunak

Dilakukannya pengujian ini adalah bertujuan untuk mengetahui apakah perangkat lunak dapat tampil dan bekerja sebagaimana desain sistem. Adapun hasil dari pengujian yaitu :



Gambar 4. 17 hasil grafik pengolahan dataset dari roboflow

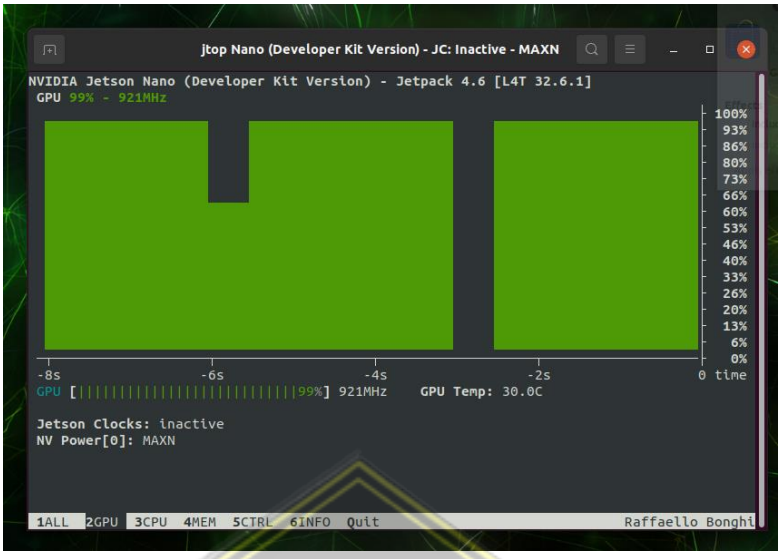
Gambar diatas yakni kemampuan roboflow menampillkan grafik hasil dari data yang diproses. Kelebihan pada roboflow untuk aplikasi end-to-end seperti ini yakni dapat untuk menghasilkan grafik data yang sudah diolah melalui aplikasi tersebut.

Selanjutnya adalah mencoba untuk menjalankan sistem dengan klasifikasi YOLOv5 dengan mengetikkan command pada gambar dibawah ini:

```
jetson@nano:~$ cd Documents
jetson@nano:~/Documents$ cd yolov5
jetson@nano:~/Documents/yolov5$ python3 detect.py --weights runs/train/exp/weights/best.pt --img 416 --conf 0.1 --source
 0
detect: weights=['runs/train/exp/weights/best.pt'], source=0, data=data/coco128.yaml, imgsz=[416, 416], conf_thres=0.1,
iou_thres=0.45, max_det=1000, device=, view_img=False, save_txt=False, save_conf=False, save_crop=False, nosave=False, c
lasses=None, agnostic_nms=False, augment=False, visualize=False, update=False, project=runs/detect, name=exp, exist_ok=F
alse, line_thickness=3, hide_labels=False, hide_conf=False, half=False, dnn=False, vid_stride=1
YOLOv5 🚀 v6.2-246-ge40662f Python-3.8.10 torch-1.12.0a0+git67ece03 CUDA:0 (NVIDIA Tegra X1, 1980MiB)

Fusing layers...
```

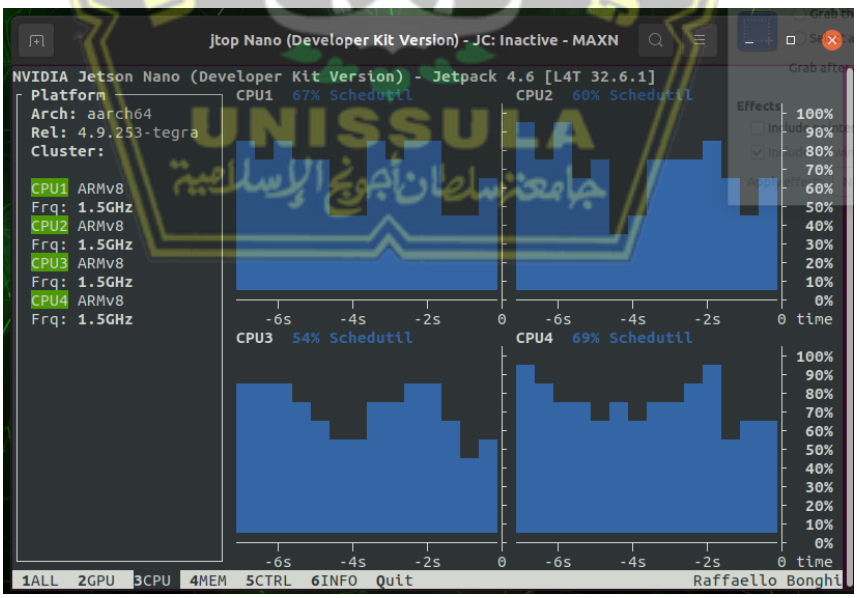
Gambar 4. 18 proses *load model* untuk berjalannya sistem

Gambar 4.18 adalah proses menjalankan program. Pertama, masuk kebagian direktori saat melakukan penginstalan YOLOv5, lalu mengetikkan commad seperti gambar diatas. Proses *load model* memakan waktu sekitar 3 menit.



Gambar 4. 19 Grafik Kinerja CPU

Gambar diatas merupakan grafik kinerja GPU saat menjalankan program, grafik tersebut menunjukkan bahwa GPU bekerja untuk image processing dengan spesifikasi 128 core dan hampir 99% pemanfaatan kinerja CUDA hampir mendekati maksimal dikarenakan frame rate yang bisa ditingkatkan pada penggunaan kinerja *jetson nano yang sudah* maksimal.



Gambar 4. 20 Grafik kinerja CPU

Tidak hanya GPU, *jetson nano* didukung dengan empat CPU yang dimana terlihat pada gambar diatas menunjukkan bahwa CPU juga membantu dalam image

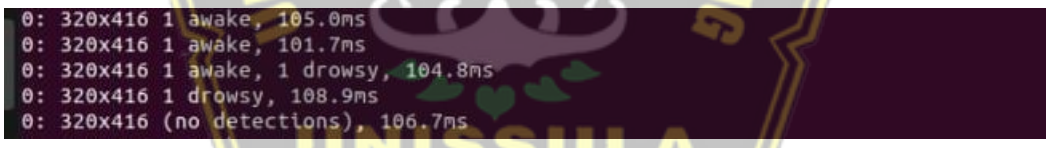
processing. Terlihat grafik CPU hampir bekerja penuh dengan presentase rata rata 90% .

Gambar 4. 21 Contoh hasil wajah sadar

Proses klasifikasi dengan model YOLOv5 telah selesai ditandai dengan *Webcam* hidup, lalu wajah akan dideteksi secara realtime. Gambar 4.21 menunjukkan hasil dari gambar wajah yang terdeteksi *awake* (sadar) dilengkapi dengan keterangan pada label dan juga akurasinya. Lalu klasifikasi yang selanjutnya pada gambar 4.22 merupakan hasil dari *drowsy* (mengantuk).

Gambar 4. 22 Contoh hasil deteksi wajah mengantuk

Hasil perhitungan pada YOLOv5 didapat dari hasil training dataset menggunakan *Google Colab*, sehingga perhitungan akurasi didapat secara otomatis. Saat program sedang berjalan, YOLOv5 juga menampilkan informasi resolusi citra yang berjalan pada latar belakang seperti yang ditunjukkan pada gambar 4.23.

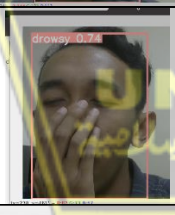


Gambar 4. 23 Resolusi citra

Gambar diatas merupakan informasi yang di tampilkan YOLOv5 berupa image size dengan ukuran 320x416 *pixel* dengan kecepatan frame rate kurang lebih *100.ms*.

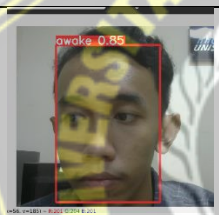
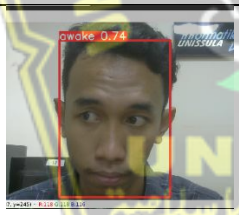
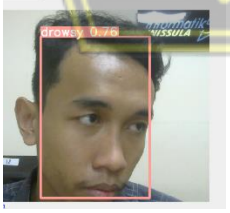
Penulis melakukan percobaan uji sampel deteksi wajah dengan jarak yang berbeda yaitu pada jarak 50 cm dan 70 cm serta percobaan pada intensitas cahaya redup. Berikut percobaan uji deteksi wajah. Dibawah ini merupakan percobaan pendeteksian wajah mengantuk dengan jarak 50 cm :

Tabel 4. 1 Pengujian percobaan wajah mengantuk dengan jarak 50 cm

Percobaan Deteksi Wajah Mengantuk (Drowsy)	Hasil Klasifikasi	Hasil Akurasi
	Berhasil	73%
	Berhasil	76%
	Berhasil	74 %
	Pendeteksian *Double*	-
	Berhasil	68%

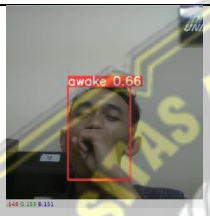
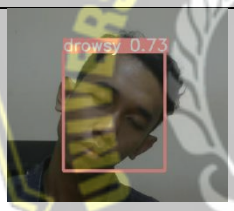
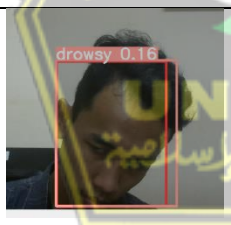
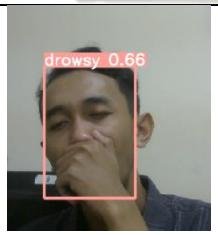
Kemudian pengujian deteksi wajah *awake* (Sadar) dengan jarak 50 cm.

Tabel 4. 2 Pengujian percobaan wajah sadar dengan jarak 50 cm

Percobaan Deteksi Wajah Sadar (Awake)	Hasil Klasifikasi	Hasil Akurasi
	Gagal	-
	Berhasil	88%
	Berhasil	85 %
	Berhasil	74%
	Gagal	-

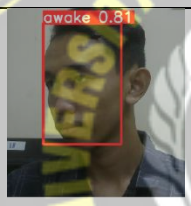
Selanjutnya ke tahap pengujian untuk wajah *drowsy* (mengantuk) dengan jarak 70 cm.

Tabel 4. 3 Pengujian percobaan wajah mengantuk dengan jarak 70 cm

Percobaan Deteksi Wajah Drowsy (Mengantuk)	Hasil Klasifikasi	Hasil Akurasi
	Berhasil	81%
	Gagal	-
	Berhasil	73%
	Pendeteksian *Double*	-
	Berhasil	66%

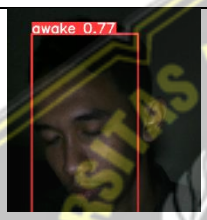
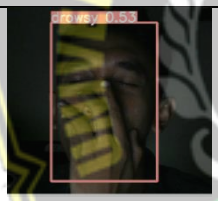
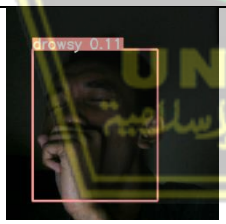
Lalu percobaa pendeteksian wajah Awake (sadar) pada jarak 70 cm

Tabel 4. 4 Pengujian percobaan wajah sadar dengan jarak 50 cm

Percobaan Deteksi Wajah Sadar (*Awake*)	Hasil Klasifikasi	Hasil Akurasi
	Berhasil	85%
	Berhasil	79%
	Berhasil	81%
	Berhasil	81%
	Berhasil	87%

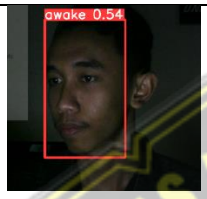
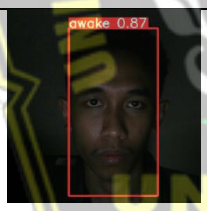
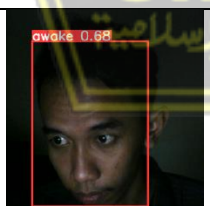
Untuk pengujian selanjutnya pada tabel dibawah ini yaitu pengujian deteksi wajah dengan intensitas cahaya redup. Dibawah ini merupakan tabel percobaan wajah mengantuk dengan keadaan intensitas cahaya redup.

Tabel 4. 5 Pengujian percobaan wajah mengantuk dengan intensitas cahaya redup

Percobaan Deteksi Wajah Mengantuk *(Drowsy)*	Hasil Klasifikasi	Hasil Akurasi
	Gagal	-
	Gagal	-
	Berhasil	53%
	Berhasil	11%
	Berhasil	56%

Kemudian pendeteksian wajah sadar *awake* (sadar).

Tabel 4. 6 Pengujian percobaan wajah sadar dengan intensitas cahaya redup

Percobaan Deteksi Wajah Sadar (*Awake*)	Hasil Klasifikasi	Hasil Akurasi
	Berhasil	82%
	Berhasil	54%
	Berhasil	52%
	Berhasil	87%
	Berhasil	68%

Pengujian yang selanjutnya adalah tahap pengujian hasil keakuratan klasifikasi yang dihasilkan YOLOv5 untuk mengetahui seberapa akurat YOLOv5 dalam melakukan suatu klasifikasi tertentu. Tabel dibawah adalah hasil pengujian keakuratan Tabel perhitungan hasil akurasi klasifikasi.

Tabel 4. 7 Tabel perhitungan hasil akurasi

Pengujian Sampel ke	Hasil Akurasi dengan Berbagai Kondisi Percobaan					
	Drowsy (50 cm)	Awake (50 cm)	Drowsy (70 cm)	Awake (70 cm)	Drowsy (redup)	Awake (redup)
1	73 %	-	81%	85%	-	82%
2	76 %	88%	-	79%	-	54%
3	74 %	85%	73%	81%	53%	52%
4	-	74 %	-	81%	11%	87%
5	68%	-	66%	87%	56%	68%
Rata-rata	72,75%	82,33%	73,33 %	82,6 %	40 %	68,6%

Dari tabel diatas mendapatkan hasil perhitungan dari pengujian. Pengujian dilakukan pada kondisi jarak 50 cm, 70 cm, dan intensitas cahaya redup. Pengujian dengan jarak 50 cm menghasilkan rata rata 72,75% untuk wajah mengantuk. Untuk wajah sadar diperoleh rata-rata 82,33%. Pada keadaan dengan jarak 70 cm diperoleh rata rata yaitu 73,33% untuk wajah mengantuk dan 82,6% untuk wajah sadar. Sedangkan pada keadaan dengan intensitas cahaya yang redup diperoleh rata rata sebesar 40 % untuk wajah mengantuk, dan untuk wajah sadar diperoleh rata-rata 68,6%. Sehingga total rata-rata hasil dari semua pengujian pada keadaan dengan jarak 50 cm, 70 cm dan cahaya redup adalah sekitar 61,9 % untuk wajah mengantuk dan 77,84% untuk wajah sadar.

4.2.2 Pengujian Perangkat Keras

Pengujian perangkat keras yakni menguji perangkat keras seperti *Jetson nano* Nvidia, Kamera, Monitor, dan perangkat-perangkat pendukung agar dapat diketahui apakah dapat berjalan sesuai desain sistem perencanaan yang telah dibuat. Hasil pengujian ditunjukkan pada Tabel 4.8 sebagai berikut.

Tabel 4. 8 Pengujian perangkat keras

Perangkat yang diuji	Skenario uji	Hasil yang diharapkan	Hasil pengujian
Keseluruhan perangkat keras	Merakit seluruh perangkat keras yang diperlukan seperti *Jetson nano,* webcam, sdcard dan perangkat pendukung lainnya seperti mouse , keyboard, dan monitor	Semua perangkat berhasil dirakit dan berfungsi dengan baik	berhasil
Jetson nano	Melakukan booting dengan memasukkan sdcard pada *jetson nano* sebagai *image file*	*Jetson nano* dapat melakukan booting sesuai yang diharapkan	berhasil
Sdcard	Mengunduh image file sebagai *operating system* untuk *jetson nano*	Berhasil melakukan flash image file pada sdcard	Berhasil
Webcam	Menampilkan gambar pada monitor serta mendeteksi wajah secara realtime	Gambar dapat menampilkan hasil dari sistem yang telah dibuat	Berhasil

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Pada penelitian yang telah dilakukan penulis, dapat disimpulkan bahwa sistem deteksi kantuk dapat diimplementasikan dengan cukup baik terhadap sistem berbasis *jetson nano*, dan berikut beberapa kesimpulan yang penulis dapat selama penelitian ini:

1. Dari penelitian yang telah dilakukan, sistem deteksi kantuk sudah berhasil diterapkan pada perangkat *Jetson nano Developer Kit 2GB.*
2. Dari hasil penelitian, sistem deteksi kantuk berbasis *jetson nano* sudah berhasil diimpelentasikan pada algoritma YOLOv5 dengan hasil akurasi diperoleh dari jumlah rata-rata semua pengujian pada keadaan dengan jarak 50 cm, 70 cm dan intensitas cahaya redup adalah sekitar 61,9 % untuk wajah mengantuk (*drowsy*) dan 77,84% untuk wajah sadar (*awake*).

5.2 Saran

Adapun saran untuk peneletian yang telah dilakukan, antara lain :

1. Menambahkan dataset yang lebih banyak untuk meningkatkan keakuratan klasifikasi YOLOv5
2. Memberikan penerangan yang cukup untuk membantu sistem deteksi kantuk dalam melakukan pendeteksian yang akurat.
3. Jarak pendeteksian antara 50 cm – 70 cm.

DAFTAR PUSTAKA

Andre Hartoko Aji Putra Perdana, Susijanto Tri Rasmana, & Heri Pratikno. (2020). Implementasi Sistem Deteksi Mata Kantuk Berdasarkan Facial Landmarks Detection Menggunakan Metode Regression Trees. *JoTI*, *1*(1), 1–9. https://doi.org/10.37802/joti.v1i1.1

Fredianto, A. R. (2018). Rancang Bangun Modul Kamera Deteksi Kecepatan Kendaraan Menggunakan Nvidia Jetson Nano Dengan Metode Image Pixel Manipulation and Calculation. In *Universitas Jember*.

Imanuddin, dkk. (2019). Deteksi Mata Mengantuk Pada Pengemudi Mobil. 18(2), 321–329.

Khairunnas, Eko Mulyanto Yuniarnodan, & Ahmad Zaini. (2021). Pembuatan Modul Deteksi Objek Manusia Menggunakan Metode YOLO untuk Mobile Robot. JURNAL TEKNIK ITS Vol. *10, No. 1*.

Levina Anora. (2021). Implementasi Model Deep Learning Untuk Deteksi Objek Candi Prambanan, Candi Borobudur, Dan Candi Ratu Boko Menggunakan Yolo V5. http://repository.ittelkom-pwt.ac.id/id/eprint/7220

Parenreng, M. M., Ahyar, Muh., Nas, M., & Hamzidah, N. K. (2021). Pengolahan Citra Untuk Simulasi Deteksi Kantuk Dengan Metode Cascade Classifier dan Black-White Ratio. Jurnal Teknologi Elekterika, *18*(2), 65. https://doi.org/10.31963/elekterika.v18i2.3258

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., Facebook, Z. D., Research, A. I., Lin, Z., Desmaison, A., Antiga, L., Srl, O., & Lerer, A. (2017). Automatic differentiation in pytorch.

Puteri, R. T., & Utaminingrum, F. (2020). Deteksi Kantuk Menggunakan Kombinasi Haar Cascade dan Convolutional Neural Network. Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer, *4*(3), 816–821.

Saputra, C. A., Erwanto, D., Rahayu, P. N., & Kadiri, I. (2021). DETEKSI KANTUK PENGENDARA RODA EMPAT MENGGUNAKAN HAAR

CASCADE CLASSIFIER dan CONVOLUTIONAL NEURAL NETWORK. *Jeecom*, *3*(1), 1–7.

Zein, A. (2018). Pendeteksian Kantuk Secara Real Time Menggunakan Pustaka OPENCV dan DLIB PYTHON. *Sainstech: Jurnal Penelitian Dan Pengkajian Sains Dan Teknologi*, *28*(2), 22–26. https://doi.org/10.37277/stch.v28i2.238