Navigasi Menu Berdasarkan Arah Pandangan Mata pada Kursi Roda Pintar menggunakkan Fusion-CNN Berbasis Jetson TX2

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:

Blessius Sheldo Putra Laksono
NIM: 195150300111021

PROGRAM STUDI TEKNIK KOMPUTER
DEPARTEMEN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2022

PENGESAHAN

Navigasi Menu Berdasarkan Arah Pandangan Mata pada Kursi Roda Pintar menggunakkan Fusion-CNN Berbasis Jetson TX2
SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun Oleh :
Blessius Sheldo Putra Laksono
NIM: 195150300111021

Skripsi ini telah diuji dan dinyatakan lulus pada
3 Januari 2023
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing Tunggal

Fitri Utaminingrum, Dr.Eng., S.T., M.T.
NIP: 198207102008122001

Mengetahui
Ketua Departemen Teknik Informatika

Achmad Basuki, S.T., M.MG., Ph.D
NIP. 197411182003121002

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar referensi.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 19 Desember 2022

Blessius Sheldo Putra L.

NIM: 195150300111021

PRAKATA

Puji syukur kepada Tuhan Yang Maha Esa, atar karuia dan berkah-Nya laporan skripsi dengan judul "**Navigasi Menu Berdasarkan Arah Pandangan Mata pada Kursi Roda Pintar menggunakkan Fusion-CNN Berbasis Jetson TX2"** dapat diselesaikan dengan baik dalam waktu sewajarnya.

Penulis menyadari bahwa kesuksesan penulisan laporan skripsi ini berkat dukungan dan bantuan dari beberapa pihak. Oleh karena itu, penulis berterimakasih sedalam-dalamnya kepada:

1. Bapak Bayu Laksono dan Ibu Ekowati Widyaningsih selaku orang tua dari penulis yang telah mendoakan, menyemangati, serta mendukung penulis baik dalam hal rohani dan jasmani sehingga skripsi penulis dapat selesai tepat waktu,
2. Ibu Ibu Dr. Eng. Fitri Utaminingrum, S.T., M.T. selaku dosen pembimbing tunggal penulis yang telah membimbing dan memberikan saran pada setiap langkah penulisan skripsi penulis,
3. Bapak Agung Setia Budi, S.T., M.T., M.Eng., Ph.D selaku dosen penasihat akademik penulis yang selalu memberikan dukungan dan nasihat dalam kehidupan akademik penulis.
4. Bapak Barlian Henryranu Prasetio, S.T., M.T., Ph.D selaku ketua program studi Teknik Komputer di Fakultas Ilmu Komputer,
5. Bapak Achmad Basuki, S.T., M.MG., Ph.D. selaku ketua Jurusan Teknik Informatika,
6. Teman-teman yang membantu penulis dalam memberikan saran dan ide dalam penulisan laporan skripsi.
7. Seluruh asisten laboratorium di Laboratorium Pembelajaran Robotika dan Sistem Tertanam,
8. Semua pihak yang tidak dapat dituliskan satu persatu.

Dengan segala kerendahan hati, penulis menyadari bahwa laporan skripsi ini jauh dari kesempurnaan, penulis mengharapkan kritik dan saran yang bersifat membangun dari dosen penguji dan pembaca. Penulis berharap laporan ini dapat memberikan manfaat bagi yang membutuhkan

Malang, 19 Desember 2022

Penulis

blessiussheldo@student.ub.ac.id

ABSTRAK

Blessius Sheldo Putra Laksono, Navigasi Menu Berdasarkan Arah Pandangan Mata pada Kursi Roda Pintar menggunakkan Fusion-CNN Berbasis Jetson TX2

Pembimbing: Fitri Utaminingrum, Dr.Eng., S.T., M.T.

Dengan kemajuan zaman, teknologi juga mengalami kemajuan pesat. Namun kemajuan ini tidak diiringi dengan adanya kemajuan pada interaksi antara komputer dan manusia. Hal ini tentunya berdampak besar bagi pengguna teknologi kepada penyandang disabilitas gerak. Teknologi kursi roda pintar juga sudah banyak digunakkan untuk membantu penyandang disabilitas namun permasalahan interaksi dengan komputer ini masih mengurangi kenyamanan pengguna. Ada beberapa jenis interaksi baru ditawarkan pada penelitian sebelumnya salah satunya dengan menggunakkan suara, namun metode ini dinilai kurang efektif dikarenakan dibutuhkan lingkungan yang minim *noise*. Cara interaksi lain yang ditawarkan adalah dengan menggunakkan estimasi arah pandangan mata menggunakkan algoritme konvensional maupun *machine learning.* Untuk melakukan hal ini algoritme konvensional dinilai kurang efektif dikarenakan adaptibilitas yang rendah dengan pengguna yang berbeda. Algoritme berbasis CNN dipilih pada penelitian ini karena kemampuannya untuk mengambil fitur dari citra sehingga algoritme ini dapat beradaptasi dengan data baru. Dari penelitian ini didapatkan akurasi model sebesar 96% dengan *loss* sebesar 0.02 pada fase *training*. Sistem dapat menjalankan algoritme ini dalam waktu 0.16 detik menggunakkan akselerasi CUDA. Sistem hanya menggunakkaan daya listrik sebesar 12 Watt yang mendung sistem untuk dijalankan menggunakkan baterai. Dari pengujian yang dilakukan dan hasil yang didapatkan dapat disimpulkan bahwa sistem dapat berjalan dengan baik untuk melakukan estimasi arahan pengguna.

Kata kunci: *Fusion CNN, Computer Vision,* Kursi Roda Pintar, Kecerdasan Buatan, *Gaze estimation*

ABSTRACT

Blessius Sheldo Putra Laksono, Gaze-based Navigation Menu on Smart Wheelchair using Fusion-CNN Based on Jetson TX2

Supervisors: Fitri Utaminingrum, Dr.Eng., S.T., M.T.

As time progresses, technology also experiences rapid advancement. However, this advancement is not accompanied by an improvement in the interaction between computers and humans. This is certainly a significant impact on users of technology, particularly those with mobility disabilities. Smart wheelchair technology has already been widely used to assist people with disabilities, but the problem of interaction with computers still reduces user comfort. Some new types of interaction have been offered in previous research, including using voice, but this method is considered less effective due to the need for a minimally noisy environment. Another offered method of interaction is using eye gaze estimation using conventional algorithms or machine learning. For this, conventional algorithms are considered less effective due to their low adaptability with different users. CNN-based algorithms are chosen in this research because of their ability to extract features from images, allowing the algorithm to adapt to new data. The accuracy of the model in this research was 96% with a loss of 0.02 during the training phase. The system can run the algorithm in 0.16 seconds using CUDA acceleration. The system only uses 12 watts of electricity, making it possible to run the system using a battery. From the testing that was carried out and the results obtained, it can be concluded that the system runs well to estimate the direction of the user.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Dengan kemajuan zaman membuat penggunaan teknologi menjadi sangat penting bahkan menjadi kecakapan yang perlu dikuasai. Penggunaan teknologi saat ini tidak hanya terfokus pada pekerjaan-pekerjaan khusus di bidang tertentu saja, penggunaan perangkat seperti komputer dan ponsel pintar menjadi kebutuhan keseharian banyak orang. Sayangnya, kemajuan teknologi yang pesat tidak diimbangi dengan kemudahan akses penggunaan teknologi tersebut bagi penyandang disabilitas fisik. Penyandang disabilitas fisik yang dimaksud adalah seseorang dengan gangguan fungsi gerak seperti lumpuh layu atau kaku terutama pada bagian tangan. Saat ini interaksi manusia dengan komputer mengandalkan penggunaan tangan pengguna untuk memberikan input kepada komputer seperti penggunaan layar sentuh maupun papan ketik. Namun penyandang disabilitas fisik terutama pada bagian tangan tidak dapat menggunakkan cara interaksi tersebut.

Salah satu alat bantu yang sering digunakkan oleh beberapa orang dengan disabilitas fisik adalah kursi roda. Banyak jenis kursi roda pintar yang dikembangkan untuk membantu penyandang disabilitas fisik dengan berbagai macam fitur seperti pengeendalian dengan suara (Rockland & Reisman, 1998) (Anam & Saleh, 2020), pengendalian menggunakkan arah pandangan (Araujo et al., 2020) (Pangestu & Utaminingrum, 2020), dan sistem pengendalian lain seperti menggunakkan pergerakan otot pengguna (Aihara et al., 2022). Penelitian sebelumnya mengenai pengembangan kursi roda pintar pengguna kursi roda diharuskan untuk berinteraksi dengan menu pada display menggunakkan layar sentuh. Hal ini tentu tidak mudah bagi beberapa penyandang disabilitas ganda. Dengan adanya keterbatasan pada tangan, mempersulit penyandang disabilitas fisik dalam memilih menu yang ada pada layar.

Beberapa cara interaksi lain sudah pernah diusulkan sebelumnya, seperti menggunakkan suara atau *speech detection* untuk mengendalikan menu pada layar. Namun cara ini dinilai kurang efektif terutama pada penggunaan di lingkungan luar dengan tingkat *noise* yang tinggi. Salah satu cara paling efektif adalah dengan menggunakkan gaze estimation yang memperkirakan arah pandang pengguna terhadap antarmuka pengguna pada layar. Gaze estimation dapat menggunakkan beberapa metode seperti penggunaan elektroda pada kulit sekitar bagian mata (electroocoulography), penggunaan kamera infra merah, dan penggunaan kamera RGB.

Penggunaan metode EOG dilakukan dengan mengukur sinyal potensial elektrik antara dua titik pada bagian sekitar mata. Pergerakan otot mata akan menghasilkan sinyal elektrik yang mengubah pembacaan pada EOG. Dibutuhkan paling sedikit empat titik pengukuran EOG untuk mendapatkan estimasi pandangan pengguna, dua titik untuk mengukur pergerakan horizontal dan dua titik untuk mengukur pergerakan vertikal (Mowrer et al., 1935). Namun

penggunaan metode EOG dapat menyebabkan ketidaknyamanan bagi pengguna terutama apabila digunakkan pada jangka waktu lama. Penggunaan elektroda EOG juga mempersulit, dikarenakan perlu adanya pemasangan oleh orang lain ketika pemakaian pertama. Elektroda EOG juga memperlukan perawatan yang lebih banyak, karena elektorda ini rentan mendapatkan gangguan yang diakibatkan oleh elektroda yang sudah terlalu lama digunakkan.

Alternatif lain dari penggunaan EOG adalah dengan menggunakkan kamera untuk melakukan gaze estimation pada video stream yang dihasilkan. Gaze estimation dapat dilakukan dengan menggunakkan kamera IR dan sumber cahaya infra merah (Naqvi et al., 2018). Sumber cahaya ini kemudian akan menghasilkan pantulan atau biasa disebut *glint* pada mata pengguna. *Glint* ini akan membantu sistem untuk menentukan arah pandangan pengguna relatif terhadap kamera (Mohan & Phirke, 2020). Untuk mendapatkan hasil yang optimal dapat digunakkan lebih dari 1 buah sumber cahaya dengan posisi yang berbeda sehingga didapatkan lebih banyak *point of reference* (Yoon et al., 2019).

Alat yang saat ini banyak dipakai untuk *gaze estimation* adalah dengan kamera RGB biasa atau kamera webcam. Untuk mendeteksi arah pandangan pengguna dengan menggunakkan gambar mata pengguna saja tidaklah cukup. Hal ini dikarenakan adanya *Wollaston's gaze illusion*. *Wollaston's gaze illusion* mendeskripsikan bahwa arah pandangan seseorang tidak hanya bergantung pada arah matanya, namun juga pada orientasi kepalanya. Sehingga untuk menyimpulkan arah pandangan seseorang dibutuhkan data mengenai orientasi kepala dan posisi bola mata pada pengguna. Dikarenakan ada dua infromas yang perlu dievaluasi diperlukan juga metode yang dapat mengevaluasi lebih dari satu fitur. Salah satu metode yang dipertimbangkan adalah dengan metode Fusion CNN, dimana Fusion CNN mengambil fitur dari beberapa objek yang kemudian digabungkan seperti pada penelitian (H. Li et al., 2017). Pada penelitian tersebut digunakkan fitur 3D dan 2D untuk menentukan ekspresi pada wajah subjek sementara pada penelitian (Hu et al., 2018) diberikan pengimplementasian *Fusion CNN* pada citra daun yang di lakukan *downsample*.

Penggunaan kamera RGB dapat memperkecil biaya yang digunakkan, bahkan dapat digunakkan kamera RGB di pasaran tanpa perlu ada modifikasi (Akinyelu & Blignaut, 2022). Dengan menggunakkan kamera RGB ada 2 jenis metode yang biasanya digunakkan, metode konvensional dan metode deep learning. Metode konvensional menggunakkan feature yang didapatkan dari gambar wajah pengguna untuk menentukan arah pandangan pengguna. Saat ini metode *appearance based* dengan *deep learning* lebih banyak digunakkan, ini dikarenakan penggunaan *neural network* yang dapat memberikan hasil yang lebih akurat (Fischer et al., 2018). Pemilihan arsitektur *neural network* untuk melakukan *gaze estimation* menjadi salah satu faktor penting untuk mendapatkan model yang akurat dan efektif dalam mendeteksi arah pandangan (Cheng et al., 2021). Dengan mempertimbangkan masalah diatas, dan didasarkan dari studi literatur yang telah dilakukan sebelumnya saya mengusulkan judul penelitian Navigasi Menu

Berdasarkan Arah Pandangan Mata pada Kursi Roda Pintar menggunakkan Fusion-CNN Berbasis Jetson TX2.

1.2 Rumusan Masalah

Dalam penelitian ini ditetapkan beberapa rumusan masalah yang menjadi dasar dari tujuan dilakukannya penelitian Navigasi Menu Berdasarkan Arah Pandangan Mata pada Kursi Roda Pintar menggunakkan Fusion-CNN Berbasis Jetson TX2. Rumusan masalah pada penelitian ini adalah,

1. Bagaimana akurasi dari kebutuhan fungsional *blink detection* dan GUI serta waktu komputasi dari *facial landmark*?
2. Bagaimana hubungan nilai *epoch* terhadap hasil akurasi dan *loss function*?
3. Berapakah nilai akurasi, presisi, *recall*, *specitivity* yang didapatkan pada pengujian sistem?
4. Bagaimana pengaruh penggunaan CUDA pada rata-rata waktu komputasi yang dibutuhkan untuk melakukan keseluruhan proses estimasi arah pandang?
5. Berapa banyak daya komputasi sistem yang digunakkan saat menjalankan model CNN pada inferensi estimasi arah pandang?
6. Berapa besar daya listrik yang digunakkan oleh sistem untuk keseluruhan proses estimasi arah pandang?
7. Berapa besar akurasi yang didapatkan ketika melakukan pengujian estimasi arah pandang berdasarkan antarmuka yang sudah dibuat terhadap ketinggian mata pengguna?

1.3 Tujuan

Berdasarkan rumusan masalah yang telah ditentukan sebelumnya maka dapat ditetapkan tujuan dari penelitian Navigasi Menu Berdasarkan Arah Pandangan Mata pada Kursi Roda Pintar menggunakkan Fusion-CNN Berbasis Jetson TX2. Tujuan penelitian ini adalah,

1. mengetahui akurasi dari algoritme *blink detection*, GUI, dan waktu komputasi dari *facial landmark.*
2. Mengetahui hubungan nilai *epoch* terhadap akurasi dan *loss function* sistem.
3. Menghitung akurasi, presisi, *recall*, dan *specitivity* ketika pengujian estimasi arah pandang pada sistem.

4. Menghitung pengaruh penggunaan CUDA pada rata-rata waktu komputasi sistem.
5. Menghitung penggunaan memori sistem dan utilisasi CPU pada sistem saat menjalankan model CNN pada inferensi estimasi arah pandang.
6. Menghitung besar daya listrik yang digunakkan sistem untuk proses estimasi arah pandang dan membandingkan dengan sistem pada saat *idle*.
7. Menghitung akurasi sistem berdasarkan antarmuka pengguna yang sudah dibuat terhadap ketinggian mata pengguna.

1.4 Manfaat

Pengembangan serta pengimplementasian penelitian Navigasi Menu Berdasarkan Arah Pandangan Mata pada Kursi Roda Pintar menggunakkan Fusion-CNN Berbasis Jetson TX2 diharapkan dapat memberikan dampak positif kepada penggunanya terutama kepada penyandang disabilitas fisik untuk mengoperasikan *menu display* pada kursi roda pintar dengan menggunakkan arah pandangan matanya sehingga dapat membantu mobilitas pengguna kursi roda pintar tersebut. Dengan penelitian ini juga diharapkan dapat membantu panyandang disabilitas fisik dalam melakukan kegiatan terutama di ruang lingkup masyarakat luas, sehingga tidak diperlukan lagi bantuan dari orang lain dan dapat hidup dengan mandiri.

1.5 Batasan Masalah

Pada penelitian ini ditetapkan beberapa batasan masalah untuk membantu agar penelitian dapat berjalan dengan efektif dan efisien. Batasan masalah yang dimaksud adalah,

1. Layar yang digunakkan untuk *gaze estimation* berukuran 7 *inch* dengan resolusi layar 1024x600.
2. Kamera yang digunakkan adalah kamera dengan interface USB.
3. Jarak wajah pengguna dengan layar adalah 35cm dengan kamera ditempatkan tepat diatas layar, menghadap ke arah pengguna.
4. Pembagian menu pada tampilan dibagi menjadi 4 tombol yaitu 2 tombol pada bagian atas kanan dan kiri untuk memilih fitur yang akan digunakkan dan 2 tombol di kiri bawah dan kanan bawah untuk mengakses fitur selanjutnya.
5. Pengguna tidak menggunakkan kacamata dengan bentuk dan jenis apapun.

1.6 Sistematika Pembahasan

Bagian sistematika pembahasan ini bertujuan menjelaskan secara umum gambaran dari masing masing bab yang akan ada pada penelitian ini. Sistematika pembahasannya adalah sebagai berikut:

BAB 1 Pendahuluan

Pada bab 1 akan dibahas pendahuluan menuju penelitian ini, disini akan dibahas latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah, serta sistematika pembahasan dari penelitian "Navigasi Menu Berdasarkan Arah Pandangan Mata pada Kursi Roda Pintar menggunakkan Fusion-CNN Berbasis Jetson TX2".

BAB 2 Landasan Kepustakaan

Bab kedua akan membahas landasan kepustakaan yang digunakkan dalam mengembangkan penelitian ini. Beberapa landasan pustaka yang akan dibahas antara lain mengenai citra digital, *facial landmark*, *convolutional neural network*, dan *gaze estimation.* Pada bab ini juga akan dibahas tentang beberapa penelitian sebelumnya sebagai dasar teori yang kemudian akan digunakkan.

BAB 3 Metodologi

Bab yang ketiga akan membahas tahapan yang dibutuhkan pada penelitian ini. Bab ini akan membahas secara detail metode penelitian yang akan digunakkan, subjek dan lokasi penelitian, teknik pengumpulan data, teknik analisis data serta perangkat pendukung yang akan digunakkan kemudian pada penelitian.

BAB 4 Rekayasa Kebutuhan Sistem

Bab ini berisi tentang syarat-syarat akan menjadi penunjang kinerja sistem agar sistem dapat bekerja dengan baik dan semestinya. Disini akan dibahas kebutuhan pengguna, penggambaran ruang lingkup pengoperasian secara umum, analisis kebutuhan fungsional, serta analisis kebutuhan non fungsional.

BAB 5 Perancangan dan Impelementasi

Bab kelima membahas bagaimana proses perancangan sistem dari segi perangkat lunak dan perangkat keras, dan proses integrasi dari perangkat lunak yang digunakkan dengan perangkat keras yang akan digunakkan.

BAB 6 Pengujian dan Analisis

Bab ini akan membahas lebih dalam hasil dari pengujian yang telah dilakukan terhadap sistem yang telah dibangun. Pada bab ini juga hasil pengujian tersebut akan dianalisis untuk menjawab pertanyaan pada rumusan masalah.

BAB 7 Penutup

Bab ini terdiri dari kesimpulan serta saran yang didapatkan dari analisis hasil pengujian yang telah dilakukan. Tujuan dari bab ini adalah agar penelitian setelahnya dapat memperbaiki kekurangan ataupun mengembangkan lebih jauh penelitian ini dengan lebih baik.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Pada bagian ini akan dibahas penelitian sebelumnya yang telah dilakukan namun masih memiliki keterkaitan dengan penelitian yang akan dilakukan. Hal ini dilakukan sebagai penunjang dasar teori dari penelitian ini. Berikut pada Tabel 2.1 ditunjukkan perbandingan dengan penelitian yang sudah dilakukan sebelumnya

Tabel 2.1 Tinjauan Penelitian Sebelumnya

No	Nama Penulis (Tahun), Judul Penelitian	Persamaan	Perbedaan	
			Penelitian Terdahulu	Rencana Penelitian
1	Aniwat Juhong (2018), Smart eye-tracking system	Mendeteksi mata untuk mengendalikan layar yang digunakkan oleh orang dengan keterbatasan fisik.	Menggunakkan metode *Circular Hough Transform*	Menggunakkan *Facial Landmark* dan *CNN*
			menggunakkan kamera yang ditempatkan pada kacamata pengguna	Menggunakkan kamera webcam yang ditempatkan diatas layar pengguna
2	Fitri Utaminingrum (2021), Eye Movement And Blink Detection For Selecting Menu On-Screen Display Using Probability Analysis Based On Facial Landmark	Menggunakkan *facial landmark* untuk melakukan *gaze estimation* terhadap layar untuk mengendalikan UI	Menggunakkan *Probability Analysis* untuk melakukan klasifikasi arah pandangan pengguna	Menggunakkan CNN untuk melakukan klasifikasi arah pandangan pengguna
3	Bin Li (2018), Etracker: A Mobile Gaze-Tracking System with Near-Eye Display Based	Menggunakkan pergerakan bola mata untuk mengendalikan	Layar yang digunakkan adalah layar *Near Eye Display*	Layar yang digunakkan adalah layar biasa dengan ukuran 7 inchi

	on a Combined Gaze-Tracking Algorithm	antarmuka pada layar	Menggunakkan kamera yang ditempelkan pada kacamata yang harus digunakkan pengguna	Menggunakkan kamera webcam yang sejajar dengan layar
4	Imam Faris (2022), Rancang Bangun Sistem Deteksi Gerakan Mata untuk Pemilihan Enam Menu Display menggunakan Circular Hough Transform berdasarkan Facial Landmarkberbasis NVIDIAJetson Nano	Mendeteksi gerakan mata untuk memilih menu pada display	Menggunakkan Circular Hough Transform untuk mendeteksi arah gerakan mata pengguna	Menggunakkan Facial Landmark dan CNN untuk mendeteksi arah pandangan mata pengguna
			Mengimplementasikan algoritme pada NVIDIA Jetson Nano	Pengimplementasian dilakukan pada NVIDIA Jetson TX2

2.1.1 Smart eye-tracking system

Penelitian ini menggunakkan posisi bola mata pengguna relatif terhadap matanya untuk memilih menu dan menggerakkan kursi roda. Penelitian ini menggunakkan kamera yang terpasang langsung ke kacamata yang kemudian akan digunakkan oleh pengguna untuk mendapatkan gambar mata pengguna. Kemudian data yang didapatkan oleh kamera akan diproses oleh Raspberry Pi dengan menggunakkan algoritme *Circular Hough Transform* untuk mendapatkan posisi bola mata dan menentukan arah pandangan pengguna terhadap layar. (Juhong et al., 2018). Hasil perhitungan dari Raspberry Pi tadi juga akan digunakkan untuk berkomunikasi dengan perangkat mikrokontroller untuk kemudian terhubung dengan API Blynk.

Implementasi dari penelitian ini menghasilkan alat yang dapat mendeteksi arah pandangan pengguna dengan harga yang lebih murah dibandingkan dengan solusi lain yang ada di pasaran seperti *Gazepoint GP3 eye tracker module*. Alat ini juga dapat dijalankan pada perangkat keras yang memiliki daya komputasi kecil seperti Raspberry Pi dibandingkan dengan solusi lain yang membutuhkan daya komputasi yang lebih besar.

2.1.2 EYE MOVEMENT AND BLINK DETECTION FOR SELECTING MENU ON-SCREEN DISPLAY USING PROBABILITY ANALYSIS BASED ON FACIAL LANDMARK

Pada penelitian ini digunakkan kamera *webcam* untuk melakukan deteksi arah pandangan pengguna dan kedipan mata pengguna untuk kontrol antarmuka pada layar. Digunakkan metode *facial landmark* untuk mendapatkan titik-titik bagian muka yang kemudian dapat digunakkan untuk melakukan *cropping* hanya pada mata pengguna. Setelah mendapatkan hanya gambar mata, gambar akan di proses hingga menjadi gambar biner dan disegmentasi untuk kemudian dihitung jumlah piksel hitam dan putih yang membentuk mata (Utaminingrum et al., 2021).

Dengan metode *probability analysis* untuk melakukan *gaze estimation* didapatkan akurasi hingga 88.1% untuk mendeteksi arah mata ke kiri, tengah, dan kanan. Metode ini juga mampu untuk mendeteksi kedipan dengan akurasi hingga 90% pada beberapa pengujian. Implementasi dari penelitian ini juga dapat digunakkan pada kondisi pencahayaan yang variatif yang membuat penggunaannya lebih fleksibel.

2.1.3 Etracker: A Mobile Gaze-Tracking System with Near-Eye Display Based on a Combined Gaze-Tracking Algorithm

Pada penelitian ini digunakkan kamera yang ditempelkan pada kacamata untuk mendeteksi gerakan mata pengguna. Hasil deteksi gerakan mata ini kemudian akan dimasukkan ke dalam *convolutional neural network* untuk selanjutnya diekstraksi fiturnya dan menghasilkan arah pandangan pada *near-eye display* . Dikarenakan layar yang digunakkan ada di depan mata pengguna dan selalu mengikuti pengguna maka tidak diperlukan untuk mengetahui arah gerakan kepala pengguna (B. Li et al., 2018).

Dengan menggunakkan CNN didapatkan hasil yang baik dalam implementasi deteksi gerakan mata pengguna. Pengujian yang dilakukan juga mendapatkan hasil *error* rata-rata sebesar 0.74° pada pendeteksian secara *coarse* dan 0.54° pada pendeteksian secara *accurate* dengan metode kombinasi. Hasil pengujian secara lengkap ditampilkan pada Tabel 2.2.

Tabel 2.2 Hasil Pengujian Etracker

Gaze Point	User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9
1	0.78	0.83	0.76	0.87	0.88	0.87	0.81	0.72	0.82
2	0.75	0.97	0.6	0.73	0.62	0.63	0.65	0.65	0.7
3	0.95	0.81	0.87	0.98	0.67	0.73	0.76	0.69	0.81
4	0.53	0.64	0.72	0.52	0.68	0.73	0.65	0.65	0.64
5	0.64	0.6	0.53	0.54	0.62	0.63	0.62	0.62	0.6
6	0.63	0.51	0.78	0.52	0.63	0.55	0.88	0.65	0.64
7	0.8	0.72	0.84	0.87	1	0.83	0.74	0.93	0.84

8	0.74	0.81	0.77	0.76	0.7	0.73	0.95	0.72	0.77
9	0.68	0.94	0.69	0.97	0.88	0.91	1.01	0.97	0.88
Average	0.72	0.76	0.73	0.75	0.74	0.73	0.79	0.73	0.74

Sumber: Bin Li (2018)

2.1.4 Rancang Bangun Sistem Deteksi Gerakan Mata untuk Pemilihan Enam Menu Display menggunakan Circular Hough Transform berdasarkan Facial Landmark berbasis NVIDIAJetson Nano

Pada penelitian ini peneliti menggunakkan metode *Hough circular transform* untuk melakukan deteksi arah pandang pengguna. Metode ini bekerja dengan menyimpulkan titik koordinat bola mata pengguna. Informasi ini yang kemudian akan menentukan arah pandangan pengguna berdasarkan lokasi bola mata pengguna relatif terhadap matanya (Faris & Utaminingrum, 2022). Pada penelitian ini dengan *w* adalah lebar dari frame, *h* adalah tinggi dari frame, *x* adalah posisi titik pusat bola mata pengguna pada sumbu *x*, dan *y* adalah posisi titik pusat bola mata pengguna pada sumbu y, peneliti menyimpulkan bahwa arah pandangan pengguna diklasifikasikan berdasarkan Tabel 2.3 dibawah

Tabel 2.3 Klasifikasi gerakan mata pengguna

Pergerakan Mata	Klasifikasi
Serong Kiri	$x > w*0.7$ dan $y < h*0.4$
Atas	$w*0.3 > x > w*0.7$ dan $y < h*0.4$
Serong Kanan	$x < w*0.3$ dan $y < h*0.4$
Kiri	$x > w*0.7$ dan $y < h*0.4$
Bawah	$w*0.3 > x > w*0.7$ dan $y > h*0.58$
Kanan	$x < w*0.3$ dan $y > h*0.4$

Sumber: Diadaptasi dari Imam Faris (2022)

2.2 Dasar Teori

Pada subab dasar teori akan dibahas apa saja yang kemudian akan digunakkan dalam penelitian. Dasar teori akan digunakkan dalam penelitian ini untuk mendukung penelitian " Navigasi Menu Berdasarkan Arah Pandangan Mata pada Kursi Roda Pintar menggunakkan Fusion-CNN Berbasis Jetson TX2”

2.2.1 Citra Digital

Citra digital adalah suatu citra yang terbagi menjadi elemen-elemen yang biasa disebut dengan piksel. Tiap piksel dalam suatu citra memiliki nilai berupa angka *finite* dengan suatu batas yang merepresentasikan intensitas piksel tersebut.

Angka ini yang kemudian akan menjadi keluaran dari fungsi dua dimensional dengan koordinat piksel pada sumbu x dan sumbu y sebagai input dari fungsi tersebut. Suatu citra digital secara intrinsik hanyalah suatu representasi digital dari objek yang diamati (Aaron et al., 2019). Sehingga akan ada disparitas antara objek yang diteliti dengan citra digital yang dihasilkan.

Citra digital dapat dibagi menjadi beberapa jenis berdasarkan cara penyimpanan data instrinsi tiap piksel pada suatu citra digital. Pada Gambar 2.1 dapat dilihat visualisasi perbedaan citra digital. Berdasarkan hal ini citra digital dapat dibagi menjadi:

1. Citra biner, adalah suatu citra digital dimana penyimpanan nilai intrinsik tiap pikselnya membutuhkan 1 angka biner atau bit. Angka ini kemudian akan merepresentasikan warna piksel pada suatu citra digital. Apabila suatu piksel f dengan fungsi f(x,y) memiliki nilai keluaran 0 maka piksel tersebut memiliki warna hitam, namun jika piksel tersebut memiliki nilai keluaran 1 maka piksel tersebut akan memiliki warna putih.
2. Citra *grayscale*, adalah citra digital dimana nilai instrinsik tiap pikselnya terdiri dari hanya 1 nilai. 1 nilai ini dapat terdiri dari 1 hingga 8 bit angka biner, rentang nilai pada citra *grayscale* 8 bit adalah 0 sampai dengan 255 yang memungkinkan 256 jenis warna. Citra *grayscale* juga dapat dibuat dengan mencari rata rata dari nilai piksel pada citra warna.
3. Citra warna, sama seperti citra *grayscale* yang memiliki nilai yang dapat berentang dari 1 bit hingga 8 atau lebih bit. Namun citra warna memiliki 3 nilai yang berbeda untuk tiap pikselnya, tiga nilai ini yang biasa juga disebut dengan *channel.* Tiap *channel* merepresentasikan intensitas piksel tersebut pada warna merah, hijau, dan biru. Tiap *channel* pada citra warna 24 bit (8 bit untuk setiap *channel*) dapat menghasilkan 256 gradasi warna dan apabila digabungkan dengan *channel* lainnya pada piksel yang sama maka tiap piksel pada citra warna dapat menghasilkan 2^{24} atau 16.777.216 warna berbeda.

Gambar 2.1 Contoh Citra Digital

Sumber: (https://www.universityofcalifornia.edu/news/human-faces-are-so-variable-because-we-evolved-look-unique)

2.2.2 *Wollaston Illusion*

Wollaston illusion atau ilusi wollaston adalah suatu fenomena yang diamati oleh William H. Wollaston pada awal abad ke-19. Pada saat itu *royal society of london* disuguhkan dengan gambar milik Wollaston. Gambar ini adalah sebuah gambar wajah seseorang dengan sepasang mata. Kemudian wollaston menggambar sepasang mata yang idetik dengan detail muka yang berbeda, sehingga pada gambar tersebut arah pandangan orang pada lukisannya berubah (Hecht et al., 2020). Hal ini yang kemudian akan dikenal sebagai *wollaston illusion*. Pada Gambar 2.2 dapat dilihat gambar ilustrasi yang ditunjukkan Wollaston pada penelitiaannya.

Gambar 2.2 Gambar ilustrasi milik Wollaston

Sumber: (https://royalsociety.org/)

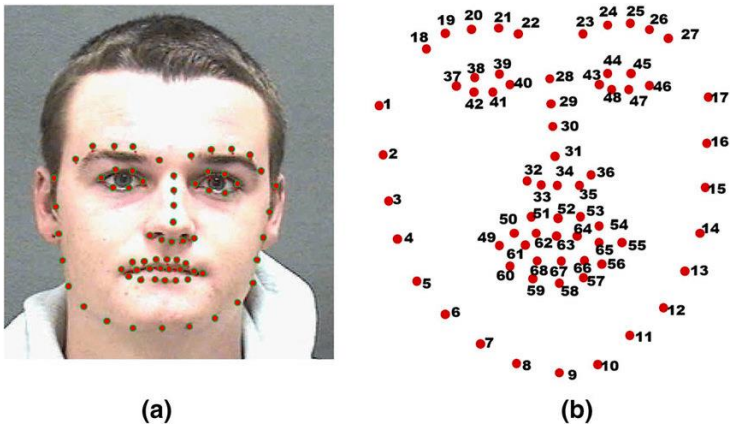
Wollaston illusion sudah diteliti oleh banyak ahli hingga saat ini. dari penelitian inilah didapatkan bahwa fitur wajah seseorang dapat mengubah persepsi arah pandangannya hingga 20°. Hal ini lah yang akhirnya menjadi dasar algoritme *gaze estimation* tidak hanya menggunakkan citra mata sebagai acuan tapi juga citra keseluruhan wajah pengguna.

2.2.3 Facial Landmark

Facial landmark adalah suatu metode yang sering digunakkan pada pengolahan citra digital, terutama pada pemrosesan yang memerlukan adanya deteksi wajah manusia dan segmentasi fitur-fitur dari wajah yang dideteksi. *Facial landmark* dapat mendeteksi beberapa fitur pada wajah manusia seperti ujung hidung, mata, dagu, bibir, dan mulut. Beberapa jenis *facial landmark* bahkan dapat dengan akurat memetakan bagian bagian mata manusia secara terpisah, seperti bola mata, iris, dan bagian putih mata (Wu & Ji, 2019).

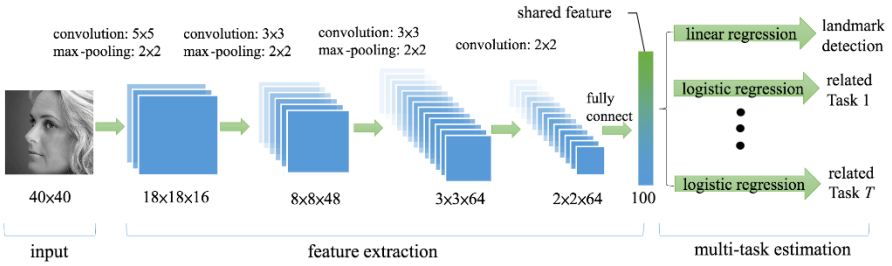
Pada awal tahun 2000an topik *face recognition* dan *facial landmark* menjadi salah satu topik yang paling banyak diperbincangkan dalam pengembangan

computer vision. Pada awalnya *facial landmark* dapat dicapai menggunakkan cara cara klasik seperti *cascade classifier* yang tidak memerlukan daya komputasi yang terlalu besar. Namun saat ini ada banyak cara baru untuk melakukan pendeteksian wajah dengan menggunakkan *neural network*. Dengan bantuan *neural network* pendeteksian wajah manusia menjadi semakin akurat dan dengan daya komputasi yang semakin meningkat pada komputer memungkinkan deteksi dengan metode *neural network* berjalan dengan baik (Fard & Mahoor, 2021). Contoh *facial landmark* 68 titik pada wajah manusia dapat dilihat pada Gambar 2.3.



Gambar 2.3 Contoh *Facial Landmark*

Algoritme *neural network* yang biasa digunakkan untuk melakukan *facial landmark* biasanya menggunakkan CNN atau *convolutional neural network* karena kemampuannya untuk memproses fitur-fitur pada citra digital dengan baik. *Layer* konvolusi yang biasa digunakkan untuk mengambil fitur dari citra digital kemudian dilanjutkan *outputnya* menuju *fully connected layer* untuk melakukan regresi linear dan mendapatkan *landmark* pada wajah subjeknya. Pada Gambar 2.4 diberikan salah satu arsitektur jaringan *neural network* yang biasa digunakkan untuk melakukan *facial landmark* dan *face detection.*



Gambar 2.4 Arsitektur CNN yang digunakkan untuk facial landmark

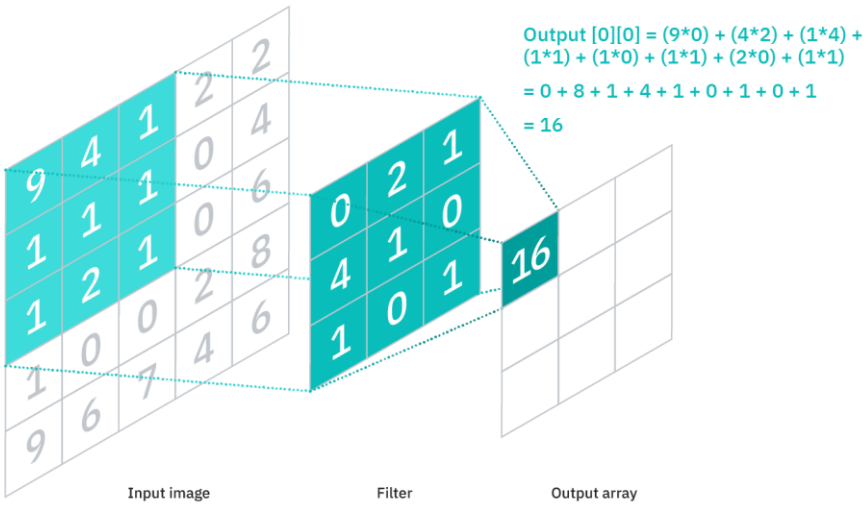
Sumber gambar: (https://wiki.tum.de/display/lfdv/Facial+Landmark+Detection)

2.2.4 Convolutional Neural Network

Convolutional neural network atau CNN adalah salah satu jenis dari algoritme *artificial neural network* yang meniru konsep cara manusia belajar mengenai pola di lingkungan sekitar. Arsitektur *neural network* sendiri diambil dari cara kerja otak manusia dimana setiap *neuron* di dalam otak akan mengeluarkan sinyal (*output)* berdasarkan sinyal masukannya (*input*) yang kemudian sinyal keluaran ini akan kembali menjadi masukan untuk *neuron* lainnya. CNN menggunakkan operasi konvolusi di dalamnya untuk mendapatkan fitur-fitur penting dari suatu citra digital (Z. Li et al., 2021).

CNN sering digunakkan pada pemrosesan citra digital dikarenakan performanya yang baik dalam mendeteksi pola yang rumit pada citra digital dibandingkan dengan melakukan *flattening* kepada citra digital dan memasukkan hasilnya ke ANN. Tugas dari CNN adalah untuk mengurangi kompleksitas dari suatu citra digital tanpa menghapus fitur spasial dari citra digital tersebut. Untuk melakukan ini CNN menggunakkan kernel untuk melakukan operasi konvolusi pada *convolutional layer* dan operasi *max pooling* pada layer selanjutnya.

Convolutional layer akan melakukan operasi konvolusi pada bagian dari citra digital di dilewati oleh kernelnya. Operasi ini akan menghasilkan *output* dengan ukuran yang lebih kecil apabila menggunakkan *valid padding* ataupun dengan ukuran yang sama dengan masukkannya apabila menggunakkan *same padding. Convolutional leyer* bertugas untuk mengekstraksi fitur tingkat tinggi dari citra masukkannya, fitur ini dapat berupa tepi, gradien orientasi ataupun warna dari citra masukkannya. Pada Gambar 2.5 dapat dilihat bagaimana operasi konvolusi mengubah suatu matriks.

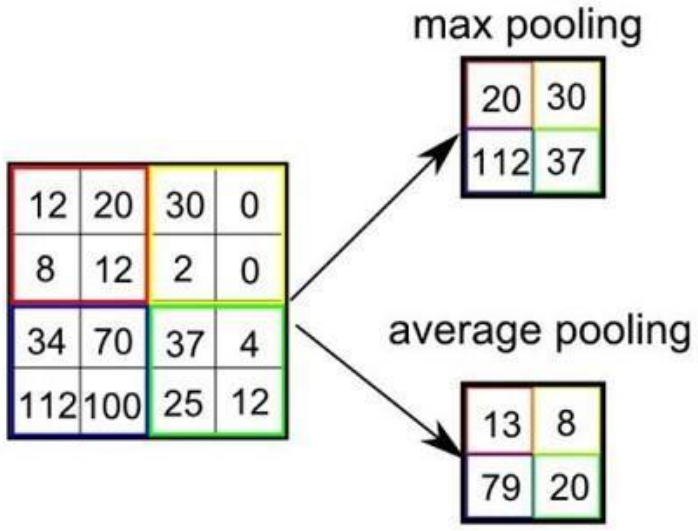


Gambar 2.5 Operasi konvolusi pada matriks

Sumber: (https://www.ibm.com/cloud/learn/convolutional-neural-networks)

Setelah melalui *convolutional layer* maka data akan masuk ke *pooling layer* dimana data masukannya akan dikurangi ukuran spasialnya meggunakkan *dimentionality reduction*. Hal ini dilakukan untuk mengurangi daya komputasi

yang diperlukan untuk memproses data. Biasanya digunakkan *layer max pooling* untuk mempertahankan fitur dominan yang ada pada citra digital. Operasi *max pooling* dianggap lebih efektif dalam mengurangi noise yang ada pada citra digital dibandingkan dengan operasi *average pooling,* hal ini dikarenakan *max pooling* hanya mengambil fitur dominan sementara *average pooling* hanya merata-ratakan nilai yang ada pada kernel. Pada Gambar 2.6 dapat dilihat bagaimana cara kerja operasi *max pooling* pada suatu matriks

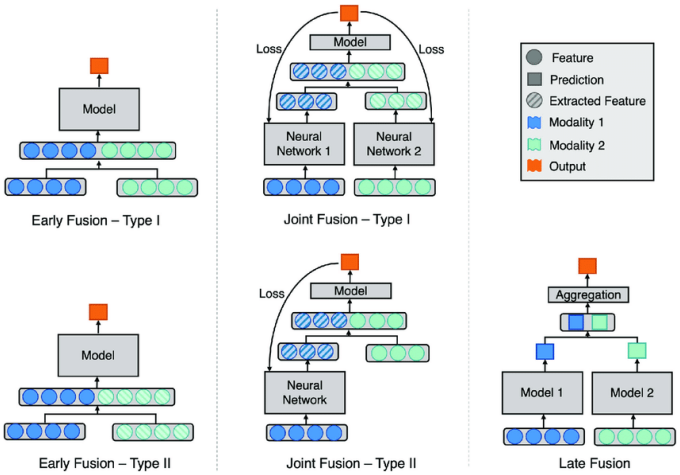


Gambar 2.6 Operasi pooling, *max pooling* dan *average pooling*

Sumber: (https://poojamahajan5131.medium.com/max-pooling-210fc94c4f11)

2.2.5 Fusion CNN

Fusion CNN adalah suatu metode untuk menggabungkan (fusion) beberapa arsitektur *convolutional neural network* yang berbeda untuk membentu satu *network* baru. *Fusion* dapat dilakukan dengan dua sampai dengan 3 arsitetur *deep learning* yang berbeda, metode ini bekerja dengan melakukan *concatenation* pada *layer* akhir dari arsitektur *deep learning* yang akan digabungkan. Pada beberapa penelitian, *fusion cnn* digunakkan untuk mengekstraksi fitur dari citra yang sama untuk mendapatkan fitur yang berbeda beda dari satu citra (Lavinia et al., 2017). Proses penggabunggan ini divisualisasikan pada Gambar 2.7.



Gambar 2.7 Berbagai jenis *fusion model*

Sumber: Huang (2020)

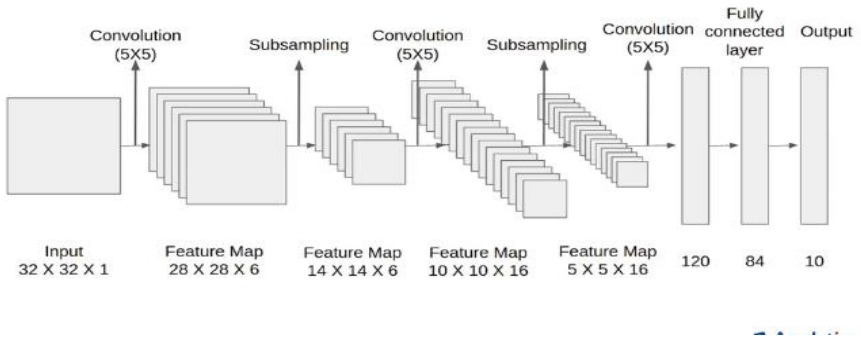
Proses penggabungan dapat dilakukan di beberapa titik pada *neural network*. *Fusion* yang dilakukan di awal atau disebut juga dengan *early fusion* menggabungkan fitur yang sudah maupun belum diekstraksi dari suatu data seperti citra dan menggabungkan keduanya. Setelah digabungkan maka data gabungan inilah yang akan menjadi *input* untuk *classifier*. Metode lainnya yang juga menggabungkan fitur sebelum memasuki *classifier* adalah *joint fusion*. Berbeda dengan *early fusion* metode ini akan melakukan *back-propagation loss function* kepada *layer* ekstraksi fitur seperti CNN. Dan yang terakhir adalah *late fusion* dimana masing masing fitur akan dimsukkan ke *classifier* yang berbeda dan hasilnya kemudian akan digabungkan di diagregasikan untuk mendapatkan *output* secara keseluruhan(Huang et al., 2020).

2.2.6 LeNet

LeNet adalah arsitektur *convlutional neural network* yang dikembangkan pada 1989 oleh Yann LeCun. LeNet merupakan salah satu arsitektur yang dikembangkan paling awal untuk jaringan *convolutional neural network*. LeNet merupakan salah satu arsitektur *convolutional neural network* yang sederhana. Arsitektur ini merupakan jaringan *feed forward* yang tiap neuronnya dapat merespon pada piksel di sekitarnya. Arsitektur ini memiliki performa yang baik dalam memproses gambar dengan skala yang besar (Lecun et al., 1989).

Arsitektur ini merepresentasikan *convolutional neural network* pada awal masa perkembangannya. LeNet menggunakan *layer convolutional* dengan *layer pooling* untuk mengurangi dimensi spasial pada keluaran *layer* sebelumnya. Tiap *convolutional layer* pada LeNet terdiri dari lapisan *convolutional, pooling*, dan fungsi aktivasi non-linier seperti ReLU. Arsitektur ini memiliki 2 *convolutional layer* yang kemudian diikuti oleh *fully connected layer*. Pada awalnya LeNet menggunakkan gambar *grayscale* dengan resolusi 28x28 sebagai masukannya dan keluaran berupa 10 *output neuron* untuk mengklasifikasikan gambar kedalam 10

kelas. Pada Gambar 2.8 digambarkan *layer* yang ada pada suatu *model* CNN dengan arsitektur LeNet5.



Gambar 2.8 Arsitektur CNN LeNet5

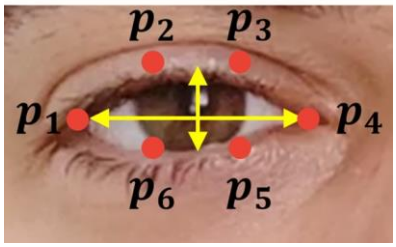
Sumber: (https://www.analyticsvidhya.com/blog/2021/03/the-architecture-of-lenet-5/)

Pada awal perkembangannya arsitektur LeNet digunakkan untuk mengklasifikasikan citra digital dari angka, oleh karena itu LeNet memiliki 10 *output neuron*. Dikarenakan perkembangan awalnya pada tahun 1989 dibutuhkan daya komputasi yang lebih besar untuk menjalankan algoritme CNN. dikarenakan hal ini arsitektur LeNet jarang digunakkan karena dibandingkan dengan metode lain seperti SVM yang dapat memberikan hasil yang sama seperti LeNet.

2.2.7 Eye Aspect Ratio

Eye aspect ratio atau EAR adalah rasio mata yang dikembangkan oleh Czezh Technical University pada 2016 (Soukupová & Cec, 2016). Nilai EAR seringkali dipakai untuk mengestimasikan seberapa terbuka mata seorang subjek. EAR dapat dihitung menggunakkan titik dari algoritme *facial landmark,* terutama titik yang berada pada mata subjek (Kuwahara et al., 2022). Kemudian nilai koordinat keempat titik tersebut dimasukkan kedalam persamaan (2.1), dengan *p1* sampai dengan *p6* merepresentasikan titik *facial landmark* pada Gambar 2.9.

$$EAR = \frac{\|p_2 - p_6\| - \|p_3 - p_5\|}{2 \times \|p_1 - p_4\|} \quad (2.1)$$



Gambar 2.9 Bagian mata dari *facial landmark*

Sumber: (https://datahacker.rs/011-how-to-detect-eye-blinking-in-videos-using-dlib-and-opencv-in-python/)

2.2.8 Disabilitas Fisik

Disabilitas fisik diartikan seseorang dengan batasan fisik ataupun disabilitas yang mempengaruhi fungsi fisik seseorang. Disabilitas fisik dapat berupa disabilitas sementara maupun permanen yang disebabkan dari berbagai macam penyakit baik penyakit turun temurun ataupun tidak secara ataupun kecelakaan. Beberapa kondisi disabilitas fisik dapat muncul begitu saja tanpa ada pola yang menentu ataupun mungkin terdapat pemburukan keadaan pasien secara bertahap. Berdasarkan data dari *World Health Organization* sekitar 15% dari populasi dunia adalah penyandang disabilitas (McClain-Nhlapo, 2022).

Disabilitas fisik dapat didefinisikan melalui beberapa kriteria seperti kesulitan dalam menggunakkan anggota badan seperti tangan, kaki, ataupun bagian lainnya. kriteria kedua adalah pengguna alat pembantu seperti tongkat berjalan dan kursi roda yang dianjurkan dokter untuk membantu mobilitas pasien. dan kriteria ketiga adalah pengguna anggota badan pengganti seperti lengan prostetik (Fergus et al., 2022).

seperti yang sudah dijelaskan sebelumnya penyandang disabilitas fisik juga mencakup pengguna kursi roda karena kesulitan untuk bergerak. pengguna kursi roda dapat bervariasi dalam umur, jenis kelamin, maupun lingkungan penggunaannya, namun mereka semua memiliki tujuan yang sama dalam menggunakkan kursi roda yaitu membantu pergerakan mereka (Armstrong et al., 2008). sekitar 10% dari penduduk dunia atau 650 juta orang memiliki disabilitas fisik dan diestimasikan sekitar 1% darinya membutuhkan kursi roda (Health Organization, 2011).

BAB 3 METODOLOGI

3.1 Tipe Penelitian

Tipe penelitian pada Navigasi Menu Berdasarkan Arah Pandangan Mata pada Kursi Roda Pintar menggunakkan Fusion-CNN Berbasis Jetson TX2 adalah implementatif pengembangan. Yang dimaksud dengan penelitian bersifat implementatif adalah penelitiaan yang menggunakkan metode penerapan baik menggunakkan perangkat lunak dan perangkat keras serta tidak hanya melakukan analisis data saja. Sementara penelitian berjenis pengembangan menunjukkan bahwa penelitian ini didasarkan pada pengembangan dari masalah dan solusi yang didapatkan dari kekurangan penelitian sebelumnya.

3.2 Strategi dan Rancangan Penelitian

Strategi dan rancangan penelitian bertujuan untuk menyesuaikan rancangan dengan kebutuhan yang dimiliki oleh penelitian ini. Pada tahap perancangan ada dua bagian yaitu perancangan hardware dan perancangan software.

Perancangan hardware memiliki tiga bagian didalamnya berupa *input,* proses, serta *output*. Pada penelitian ini *input* yang digunakkan adalah citra digital berwarna dengan format warna RGB. Citra digital ini akan diperoleh menggunakkan *webcam* yang ada pada sistem. Pemrosesan dilakukan dengan menggunakkan *Single Board Computer* Nvidia Jetson TX2 yang kemudian akan memproses citra digital dari *webcam*. Sementara *output* yang dihasilkan dari sistem adalah pergerakan pada menu yang ditampilkan pada layar. *Output*nya juga dapat berupa pilihan dari menu yang ada pada layar pengguna berdasarkan arah mata pengguna.

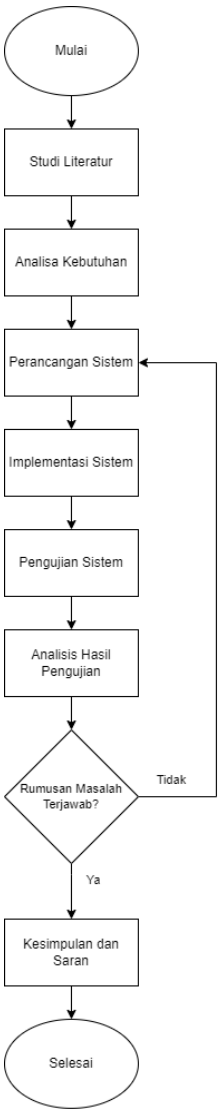
Perancangan secara software menggunakkan *convolutional neural network* untuk melakukan ekstraksi fitur sekaligus klasifikasi arah pandangan mata pengguna. Untuk menghasilkan nilai *input* yang sesuai dengan kriteria yang dimiliki oleh *convolutional neural network* yang penulis gunakkan maka diperlukan adanya proses *preprocessing*. Pada proses ini citra digital yang ditangkap oleh kamera akan dimasukkan kedalam *face predictor* untuk mendapatkan wajah pada gambar. Setelah *face predictor* mendapatkan wajah pengguna dan memberikan *landmark* pada wajah pengguna maka bagian mata kanan pengguna akan di crop dan dilakukan resize serta pengubahan citra menjadi *grayscale*. Setelah itu citra digital yang telah melalui *preprocessing* akan masuk ke *convolutional neural network* dan didapatkan klasifikasi arah pandangan pengguna.

3.2.1 Metode Penelitian

Dalam suatu penelitian harus disesuaikan dengan kebutuhan yang ada. Dimulai dari kebutuhan awal untuk dapat melakukan penelitian ini yaitu diperlukannya informasi mengenai materi, permasalahan, pengumpulan, serta pengkajuan teori yang kemudian akan digunakkan. Selain dari teori yang sudah ada diperlukan juga pengkajian terhadap penelitian sebelumnya yang juga

mendukung penelitian ini hingga dapat tercapai tujuan dari penelitian ini dan permasalahan dapat dipecahkan. Setelah dilakukannya pengkajian dan studi literatur terhadap teori dan penelitian sebelumnya, dapat dilanjutkan dengan perancangan sistem , implementasinya, serta pengujian sistem tersebut.

Setelah melakukan pengujian maka baiknya data yang didapatkan dari hasil pengujian tersebut dianalisis untuk mendapatkan informasi yang lebih mendalam tentang sistem yang diuji. Dari data pengujian dan analisisnya ini juga dapat memberitahukan apakah sistem yang diuji telah memenuhi kebutuhan yang ada pada latar belakang serta menjawab rumusan masalah yang ada. Secara keseluruhan tahapan penelitian yang akan dilakukan digambarkan dengan diagram alir pada Gambar 3.1.



Gambar 3.1 Diagram alir penelitian

3.2.2 Subjek Penelitian

Subjek penelitian pada Navigasi Menu Berdasarkan Arah Pandangan Mata pada Kursi Roda Pintar menggunakkan Fusion-CNN Berbasis Jetson TX2, adalah penyandang disabilitas fisik dengan gangguan alat gerak tangan dan kaki yang menggunakkan kursi roda pintar. Dengan penelitiaan ini pengguna kursi roda pintar dapat dengan mudah memilih fitur yang ingin digunakkan pada layar tanpa bantuan dari pihak lain.

3.2.3 Lokasi Peneltian

Lokasi yang akan digunakkan untuk mengambil dataset adalah lingkungan sekitar Laboratorium *Computer Vision* di Gedung F lantai 9 Falkultas Ilmu Komputer Universitas Brawijaya. Pengambilan data akan dilakukan dengan mengambil langsung citra digital dengan kamera di lengkungan sekitar laboratorium *computer vision* dengan kondisi pencahayaan dan waktu yang bervariatif.

3.2.4 Teknik Pengumpulan Data

Data yang akan digunakkan dalam penelitian ini diambil dengan melakukan observasi terhadap sistem yang dibuat serta dari tujuan dibuatnya sistem. Pengambilan data menggunakkan kamera *webcam* Logitech C920 yang terhubung dengan laptop dan display 7 inchi yang dipasang dengan posisi sebagaimana kemudian akan dipasangkan pada *smart wheelchair*. Kemudian pengguna akan diminta untuk melihat ke layar dan memposisikan badan tegak terhadap layar serta pada jarak yang optimal. Kemudian pengguna akan melihat ke arah yang diminta pada layar tanpa menggerakan kepalanya sehingga kamera *webcam* dapat mengambil citra mata pengguna selama pengguna melihat ke arah yang telah di program.

Citra yang digunakkan akan didapatkan dengan jarak mata sampai dengan layar sebesar 30 hingga 35 cm, mengingat jarak pandang ideal penggunaan layar 7 inchi. Layar dan *webcam* juga ditempatkan sejajar menghadap arah yang sama sehingga tidak ada disparitas antara mata dan layar. Tiap citra yang digunakkan untuk dataset telah dilakukan *resize* sebelumnya sehingga ukuran antara citra digital seragam. Ukuran yang dipilih adalah 100 piksel kali 50 piksel, ukuran ini digunakkan untuk mempertahankan *aspect ratio* dari citra mata dan untuk meringankan komputasi ketika melakukan *training*. Diambil juga citra wajah pengguna untuk mendapatkan hasil yang akurat walaupun dipengaruhi ilusi wollaston.

Keseluruhan dari *dataset* yang kemudian akan digunakan memiliki jumlah 10.000 data dengan pembagian 1000 citra mata dan 1000 citra wajah untuk kelas “kiri atas”, 1000 citra mata dan 1000 citra wajah untuk kelas “kanan atas”, 1000 citra mata dan 1000 citra wajah untuk kelas “Kiri bawah”, 1000 citra mata dan 1000 citra wajah untuk kelas “kanan bawah”, dan 1000 citra mata dan 1000 citra wajah untuk kelas “*unknown*”. Pengambilan data juga dilakukan dengan pencahayaan yang bervariatif dengan mempertimbangkan studi kasus

penggunaan *smart wheelchair*. Pada Gambar 3.2 ditunjukkan *dataset* yang telah penulis peroleh. Pada gambar paling atas ditunjukkan wajah dan mata yang melihat ke arah kiri atas, pada gambar kedua melihat ke arah kanan atas, pada gambar ketiga melihat ke arah kiri bawah, pada gambar keempat melihat ke arah kanan bawah, dan pada gambar paling bawah tidak melihat pada layar.

Gambar 3.2 Contoh dataset yang akan digunakan

3.2.5 Teknik Analisis Data

Teknik analisis data didapatkan dengan menganalisis hasil pengujian sistem secara langsung. Tujuan dari analisis yang dilakukan adalah untuk mengetahui lebih dalam mengenai kemampuan sistem dan batasan batasan yang dimiliki sistem saat diuji. Dalam pengujian terdapat beberapa aspek yang harus diperhatikan, antara lain aspek asperk tersebut adalah:

1. Menghitung akurasi dari algoritme *blink detection*, GUI, dan waktu komputasi dari *facial landmark.*
2. Mengetahui hubungan nilai *epoch* terhadap akurasi dan *loss function* sistem.

3. Menghitung akurasi, presisi, *recall*, dan *specitivity* ketika pengujian estimasi arah pandang pada sistem.
4. Menghitung akurasi sistem berdasarkan antarmuka pengguna yang sudah dibuat (kiri atas, kanan atas, kiri bawah, kanan bawah) terhadap ketinggian mata pengguna.
5. Menghitung pengaruh penggunaan CUDA pada rata-rata waktu komputasi sistem.
6. Menghitung penggunaan memori sistem dan utilisasi CPU pada sistem saat menjalankan model CNN pada inferensi estimasi arah pandang.
7. Menghitung besar daya listrik yang digunakkan sistem untuk proses estimasi arah pandang dan membandingkan dengan sistem pada saat *idle*

3.2.6 Peralatan Pendukung

Peralatan pendukung yang dimaksud adalah peralatan yang dibutuhkan dalam berjalannya penelitian ini dengan tujuan untuk menunjang kinerja sistem untuk menghasilkan *output* yang baik dan benar.

Perangkat Keras:

1. Kamera *webcam* Logitech C920
2. Nvidia Jetson TX2NX
3. Carrier Board jetson TX2NX A203V2
4. Layar 7 Inchi

Perangkat Lunak:

1. *Library* OpenCV
2. *Library* PyTorch
3. *Library* Dlib
4. Visual Studio Code

BAB 4 REKAYASA KEBUTUHAN

4.1 Kajian Masalah

Pada penggunaan kursi roda pintar dibutuhkan adanya proses pemilihan fitur oleh pengguna kursi roda. Namun tidak semua pengguna dapat menggunakkan tangannya untuk memilih menu pada layar sentuh. Untuk membuat kursi roda yang lebih inklusif terhadap pengguna dengan disabilitas fisik lain maka diperlukan adanya metode lain yang terintegrasi dengan kursi roda pintar untuk kemudian digunakkan sebagai sarana interaksi pengguna dalam memilih fitur yang akan digunakkan.

Dengan penelitian sebelumnya yang telah dibahas lebih lengkap pada bab 2 dapat disimpulkan bahwa penggunaan *gaze estimation* untuk memilih menu pada layar adalah salah satu metode interaksi yang paling cocok untuk digunakkan. *Gaze estimation* menggunakkan kamera tidak memerlukan adanya sensor yang ditempelkan pada pengguna dan penggunaannya tidak rentan terhadap noise dan gangguan dari lingkungan sekitarnya.

4.2 Identifikasi Stakeholder

Pada penelitian ini ada beberapa stakeholder yang perlu dipertimbangkan pada perancangan sistem Navigasi Menu Berdasarkan Arah Pandangan Mata Pada Kursi Roda Pintar Menggunakkan Fusion-Cnn Berbasis Jetson Tx2. Salah satu stakeholder utama pada perancangan sistem adalah pengguna sistem. Sistem dirancang untuk digunakkan oleh pengguna kursi roda pintar dengan disabilitas fisik terutama dengan kesulitan untuk berjalan dan menggunakkan tangan.

4.3 Kebutuhan Fungsional

Kebutuhan fungsional akan menjelaskan bagian-bagian sistem yang akan digunakkan serta tujuan dari penggunaan bagian tersebut. Berikut adalah kebutuhan fungsional dari sistem Navigasi Menu Berdasarkan Arah Pandangan Mata pada Kursi Roda Pintar menggunakkan Fusion-CNN Berbasis Jetson TX2:

1. Sistem dapat menjalankan algoritme *facial landmark* untuk menemukan wajah pengguna.

   Pada sistem ini citra keseluruhan dari pengguna akan diproses menggunakkan *facial landmark.* Hal ini dilakukan untuk mendapatkan titik *landmark* pada wajah pengguna yang selanjutnya akan digunakkan untuk melakukan proses *cropping* pada bagian mata dan wajah pengguna.

2. Sistem dapat menunjukkan antarmuka grafik untuk pengguna kursi roda pintar dan memberi tahu pengguna hasil deteksi arah pandangan melalui antarmuka grafik tersebut.

   Sistem dapat menunjukkan antarmuka grafis kepada pengguna kursi roda pintar untuk menunjukkan fitur yang dapat dipilih oleh pengguna.

Kemudian setelah mendapatkan hasil pendeteksian, sistem juga dapat menunjukkan bagian mana yang pengguna pandang saat itu dengan menggunakkan warna yang berbeda pada bagian menu yang dipilih pengguna.

3. Sistem dapat mendeteksi apabila pengguna dengan sengaja mengedipkan mata untuk memilih menu yang ditunjukkan pada layar.

   Untuk memilih fitur pada menu kursi roda pintar pengguna dapat dengan sengaja mengedipkan matanya. Kedipan ini yang kemudian juga akan dideteksi oleh sistem untuk menandakan bahwa pengguna ingin mengakses menu yang dipandang. Sistem juga dapat membedakan antara kedipan yang disengaja oleh pengguna untuk memilih menu pada layar dan kedipan refleks berdasarkan waktu berlangsungnya kedipan.

4. Sistem dapat mengklasifikasikan citra mata dan wajah untuk menentukan arah pandangan pengguna.

   Sistem dapat melakukan ekstraksi fitur dan klasifikasi dengan algoritme CNN sehingga arah pandangan pengguna dapat dideteksi dari citra yang sudah ditangkap oleh kamera. Citra yang didapatkan akan diklasifikasikan kedalam 5 kelas berdasarkan arah pandangan pengguna, kelas yang dideteksi adalah:

   - Kelas 0 yang berarti pengguna memandang ke arah kiri atas layar.
   - Kelas 1 yang berarti pengguna memandang ke arah kanan atas layar.
   - Kelas 2 yang berarti pengguna memandang ke arah kiri bawah layar.
   - Kelas 3 yang berarti pengguna memandang ke arah kanan bawah layar.
   - Kelas 4 yang berarti pengguna tidak memandang ke arah layar.

5. Sistem dapat menangkap citra wajah dan mata pengguna.

   Kamera digunakkan untuk menangkap citra pengguna secara penuh, citra ini kemudian akan dilakukan proses *cropping* terhadap wajah dan mata pengguna serta proses *resize* ke ukuran 100 x 50 untuk mata dan 100 x 100 untuk wajah. Citra inilah yang kemudian akan diproses untuk klasifikasi oleh Nvidia Jetson TX2.

4.4 Spesifikasi Sistem

Spesifikasi sistem akan menjelaskan mengenai alternatif spesifikasi sistem yang agar memenuhi kebutuhan fungsional sistem dengan baik. Bagian ini juga akan membahas mengenai kebutuhan non-fungsional lainnya dari sistem yang dirancang. Dari kebutuhan fungsional yang sebelumnya sudah di definisikan pada bagian 4.3 didapatkan kebutuhan non-fungsinal yang akan ada pada sistem antara lain,

1. Dikarenakan sistem akan dijalankan secara *portable* menggunakkan baterai 24V maka dibutuhkan komputer yang dapat bekerja pada tegangan 24V atau lebih kecil dan tidak menggunakkan daya yang terlalu besar.
2. Sistem akan ditempatkan pada kursi roda pintar sehingga harus dapat dimasukkan pada bagian senderan tangan bagian kanan kursi roda.
3. Algoritme yang digunakkan untuk melakukan komputasi menggunakkan *framework* Pytorch yang memiliki dukungan untuk akselerasi pemrosesan menggunakkan CUDA. Sehingga untuk mendapatkan performa yang optimal sistem harus memiliki kartu grafis dengan kemampuan CUDA.
4. Untuk menangkap citra dengan efisien dibutuhkan kamera dengan protokol yang dapat diadaptasikan dengan mudah oleh komputer yang akan digunakan. Oleh karena itu disarankan untuk menggunakkan kamera dengan protokol USB
5. Dikarenakan layar akan digunakkan dengan jarak yang tidak terlalu jauh dari pengguna atau sekitar 40-50 cm sehingga ukuran yang direkomendasikan adalah 7-10 inci. Mempertimbangkan juga sumber daya yang digunakkan oleh sistem adalah baterai maka layar juga harus dapat dijalankan dengan sumber tegangan dari USB untuk kemudahan dan efisiensi penggunaan daya.

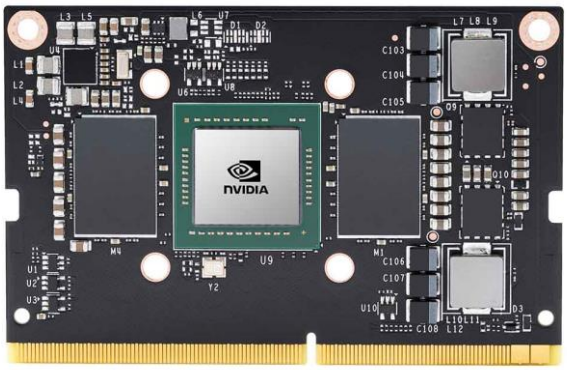
4.5 Analisis Kebutuhan Perangkat Keras Dan Perangkat Lunak

Dari kebutuhan fungsional sistem yang sebelumnya telah di dibuat maka dapat disusun spesifikasi perangkat yang kemudian dapat menunjang kebutuhan fungsional sistem. Berikut adalah spesifikasi dari sistem Navigasi Menu Berdasarkan Arah Pandangan Mata pada Kursi Roda Pintar menggunakkan Fusion-CNN Berbasis Jetson TX2:

4.5.1 Spesifikasi Perangkat Keras

1. Nvidia Jetson TX2 NX dan *Carrier Board*-nya

   Berdasarkan kebutuhan fungsional 2, 3, 4, dan 5 dibutuhkan Nvidia Jetson TX2 untuk menjadi pemroses utama sistem. SBC ini dipilih karena memiliki GPU dengan inti CUDA untuk membantu akselerasi proses inferensi menggunakkan algoritme CNN. Nvidia jetson TX2 sendiri adalah SBC atau *single board computer* buatan Nvidia yang difokuskan pada pengaplikasian *machine learning* dan *artificial learning*. SBC ini dilengkapi dengan GPU buatan Nvidia dengan arsitektur Pascal yang memiliki 256 CUDA *cores*. GPU dapat digunakkan untuk mengakselerasi komputasi *machine learning* dan *artificial intellegent.* Perangkat ini yang kemudian digunakkan untuk melakukan proses *inference* pada CNN. Gambar modul jetson TX2NX dapat dilihat pada Gambar 4.1. Dan secara keseluruhan spesifikasi dari Jetson TX2NX dapat dilihat pada Tabel 4.1.



Gambar 4.1 Modul Jetson TX2NX

Sumber: (https://developer.nvidia.com/embedded/jetson-tx2-nx)

Tabel 4.1 Spesifikasi Teknis Jetson TX2NX

Spesifikasi Teknis	
Performa AI	1.33 TFLOPs
GPU	GPU NVIDIA dengan arsitektur Pascal™ yang memiliki 256 CUDA *cores*.
CPU	*Dual-core* NVIDIA Denver 2 64-bit CPU dan *quad-core* ARM A57 Complex
RAM	4GB 128-bit LPDDR4, 1600 MHz - 51.2 GBs
Penyimpanan	Penyimpanan flash 16GB eMMC 5.1

Sumber: diadaptasi dari (https://developer.nvidia.com/embedded/jetson-tx2-nx)

Agar bisa menggunakkan TX2NX dibutuhkan juga *carrier board* sebagai ekspansi I/O dan *interface* untuk TX2NX. Pada penelitian ini digunakkan *carrier board* dari Seeed Studio yaitu A203V2 karena ukurannya yang kecil namun fitur yang masih mumpuni untuk kursi roda pintar. *Carrer board* ini juga dipilih karena memiliki slot SSD M.2 NVME yang membantu untuk memasukkan kode untuk tiap fitur yang akan diimplementasikan serta slot M.2 key-e yang digunakkan untuk memasangkan *network card* untuk menambahkan kemampuan wifi pada sistem. Pada Gambar 4.2 dapat dilihat *carrier board* yang digunakkan.

Gambar 4.2 *Carrier board* A203V2

Sumber: (https://www.robotshop.com/en/a203-v2-carrier-board-jetson-nano-xavier-nxtx2-nx-w-wifi-bluetooth-ssd.html)

Sistem Jetson TX2NX dipilih karena kemampuan yang dinilai lebih bagus dibandingkan *microcomputer* lainnya di pasaran. Penelitian (Süzen et al., 2020)mendapatkan bahwa Jetson TX2 memiliki performa 28% lebih baik dari Jetson Nano dan 86% lebih baik dibandingkan dengan Raspberry Pi 4 dinilai dari waktu komputasi yang didapatkan ketika pegujian training model CNN dengan dataset sebanyak 5000 gambar. Seperti yang dapat dilihat pada Tabel 4.2, hanya Jetson TX2 yang mampu untuk melakukan *training* dengan 30 ribu dan 45 ribu gambar dikarenakan RAM yang lebih besar pada Jetson TX2

Tabel 4.2 Perbandingan Perangkat Keras

	Akurasi (%)			Waktu Komputasi (detik)		
	TX2	Nano	Pi 4	TX2	Nano	Pi 4
Idle	-	-	-	-	-	-
5K	87.6	87.5	87.2	23	32	173
10K	93.8	93.9	91.6	32	58	372
20K	94.6	94.5	-	52	-	462
30K	96.4	-	-	122	-	-
45K	97.8	-	-	235	-	-

Sumber: Diadaptasi dari (Süzen et al., 2020)

2. Webcam

Sesuai dengan kebutuhan fungsional pertama dari sistem dibutuhkan adanya alat untuk menangkap citra wajah pengguna seperti webcam. Webcam atau *web camera* adalah perangkat yang digunakkan untuk mengambil citra digital yang kemudian akan digunakkan pada aplikasi *web*. Webcam tidak hanya dapat menangkap citra digital dalam bentuk foto namun dapat juga mengambil video. Pada implementasinya terdapat dua

jenis webcam yang dapat digunakkan yaitu webcam dengan koneksi usb dan webcam dengan koneksi *wireless* (biasanya menggunakkan jaringan wifi ataupun bluetooth). Dikarenakan dibutuhkannya komputasi realtime maka *latency* dari kamera sangat diperhatikan. Dengan mempertimbangkan faktor ini maka digunakkan webcam dengan koneksi USB untuk mengurangi *input lag* dan *latency* yang biasa terjadi pada koneksi *wireless.* Logitech C290 dipilih sebagai input dari sistem dikarenakan koneksi dengan komputer menggunakkan *interface* USB yang sudah menjadi standar dan resolusi yang memadai untuk mendapatkan citra wajah pengguna. Pada Gambar 4.3 ditunjukkan webcam logitech C290 yang akan digunakkan dalam penelitian ini.

Gambar 4.3 Webcam Logitech C920

Sumber: (https://www.allaboutcircuits.com/news/teardown-tuesday-logitech-hd-pro-webcam-c920/)

Kamera Logitech C920 juga dipilih karena memiliki spesifikasi yang jauh lebih baik dibandingkan kompetitornya di atas kertas. Dapat dilihat pada Tabel 4.3 berisi dari data yang dikumpulkan dari *website* resmi Logitech. Kamera ini memiliki dFoV atau *diagonal field of view* yang lebih lebar, hal ini sangat berguna ketika menangkap citra pengguna dari dekat. Kemudian resolusi yang lebih besar juga membuat gambar menjadi lebih jernih dibandingkan kamera yang lain.

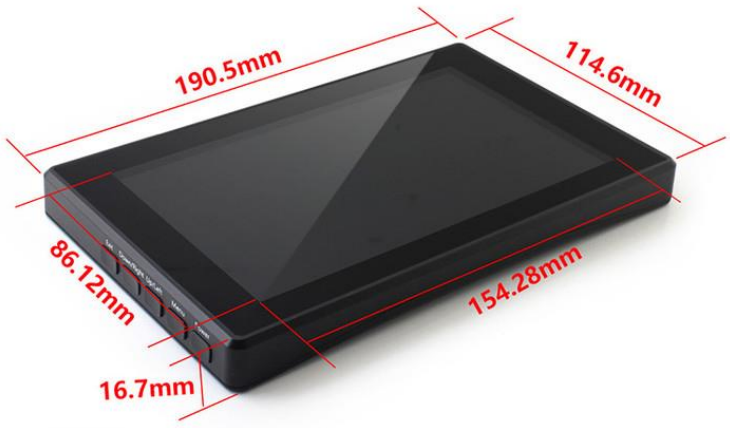
Tabel 4.3 Perbandingan Antara Kamera

	Kamera		
	C920	C310	C270
Resolusi	1080p/30fps	720p/30fps	720p/30fps
Tipe Fokus	Autofocus	fixed	fixed
Tipe Lensa	Kaca	Plastik	Plastik
dFoV	78°	60°	55°

Sumber: Diadaptasi dari (https://www.logitech.com/)

3. Layar

Berdasarkan kebutuhan fungsional ketiga dan keempat dibutuhkan adanya layar untuk menunjukkan antarmuka grafis kepada pengguna. Layar adalah suatu perangkat keras yang berguna untuk memberikan output kepada pengguna berupa gambar yang dapat dilihat langsung. Layar juga dapat dipergunakkan untuk menampilkan antarmuka untuk pengguna berinteraksi dengan suatu sistem. Dengan kemajuan teknologi saat ini layar dengan ukuran kecil dan memiliki resolusi tinggi sangat mudah ditemukan dengan harga yang tidak terlalu tinggi, beberapa layar juga memiliki kemampuan input seperti *tourch screen*. Pada penelitian ini penulis memilih layar dari Waveshare sebesar 7 inchi dengan resolusi 1024 x 600 dan kemampuan touch screen. Layar ini dipilih karena resolusi yang mencukupi untuk jarak pandang 30-35cm berdasarkan jarak pandangan optimal layar 7 inchi dan hanya memerlukan sumber listrik dari USB. Pada Gambar 4.4 ditunjukkan layar yang kemudian akan digunakkan untuk penelitian ini beserta dimensi dari layar tersebut.



Gambar 4.4 Dimensi layar yang akan digunakkan

Sumber: (https://www.tokopedia.com/kiosrobot/monitor-raspberry-pi-7-inch-inci-lcd-hdmi-ips-touch-screen-full-case)

4.5.2 Spesifikasi Perangkat Lunak

1. Pytorch

Sesuai dengan kebutuhan fungsional ketiga dan kelima dibutuhkan adanya *framework* untuk melakukan proses *inference*. Digunakkan *framework* Pytorch karena adanya *support* untuk akselerasi menggunakkan CUDA dan dapat berjalan pada bahasa pemrograman python. Pytorch adalah framework machine learning open-source yang dikembangkan berbasis library torch. Framework ini digunakkan utamanya pada aplikasi computer vision dan natural language processing. Pytorch dikembangkan sebagian besar oleh Meta AI, perusahaan riset di bidang AI

milik facebook. Pytorch memfokuskan pengembangannya menggunakkan bahasa python, namun juga memiliki implementasi di bahasa low-level seperti C++ dengan performa yang lebih baik (Paszke et al., 2019).

Pytorch sudah menjadi framework pilihan untuk mengembangkan beberapa aplikasi deep learning seperti autopilot milik pengembang mobil elektrik tesla. Pytorch memiliki banyak modul untuk memudahkan pengembangan artificial intellegent seperti modul nn yang memudahkan pengguna untuk mendefinisikan grafik komputasional dan gradien. Modul nn membantu pengguna untuk membuat neural network sesuai dengan spesifikasi yang diinginkan oleh pengguna.

Dengan menggunakkan pytorch akan memudahkan proses pengembangan sistem. Dengan adanya *support* untuk akselerasi proses *inference* menggunakkan GPU terutama dengan CUDA juga akan mempercepat waktu komputasi yang diperlukan oleh algoritme CNN untuk bekerja.

2. NumPy

Sebagai requirement dari *framework* Pytorch dibutuhkan *library* NumPy. Numpy adalah *library* yang dikembangkang untuk bahasa pemrograman python, dengan tujuan utama untuk menambahkan fungsionalitas operasi *array* multidimensional, matriks, dan komputasi matematika tingkat tinggi lainnya. Numpy pertama dikembangkan pada 1995 dengan nama Numeric, dan pada 2005 namanya berubah menjadi Numpy dan fokus awalnya untuk menarik komunitas saintis ke komputer menjadi seperti sekarang ini. Pada penelitian *library* numpy membantu penulis untuk melakukan operasi operasi pada citra digital yang biasanya direpresentasikan menggunakkan matriks. Operasi dengan menggunakkan algoritme *deep-learning* seperti CNN juga menggunakkan Numpy untuk memproses data dan mengoperasikan matriks matriks pada CNN.

3. Dlib

Untuk memenuhi kebutuha fungsional kedua maka diperlukan adanya Dlib untuk menjalankan deteksi wajah dan penempatan titik *facial landmark.* Dlib adalah *library open-source cross platform* yang dikembangkan dengan bahasa pemrograman C++. Dlib adalah *library* dengan kegunaan yang umum, beberapa komponen yang ada di dlib dapat digunakkan untuk keperluan data struktur, koneksi jaringan, hingga *machine learning*. Saat ini Dlib lebih memfokuskan pada pembuatan *library* untuk membantu *statistical machine learning.*

4. OpenCV

Untuk mendapatkan citra sesuai dengan kebutuhan fungsional serta melakukan *preprocessing* kepada citra sebelum dilakukan *inference* menggunakkan CNN. OpenCV atau open-source computer vision library adalah kumpulan fungsi-fungsi programming yang diperuntukan secara

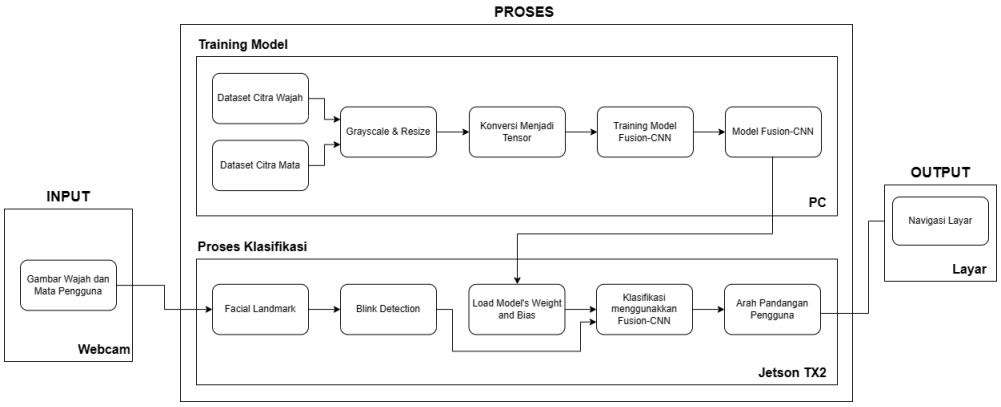
khusus untuk komputasi computer vision secara real-time. OpenCV awalnya dikembangkan oleh Intel, library ini gratis untuk digunakkan dengan lisensi open-source apache 2. Semenjak 2011 library OpenCV mendukung akselerasi menggunakkan GPU atau Graphics Processing Unit untuk membantu pada pemrosesan real-time.

OpenCV ditulis dengan menggunakkan bahasa pemrograman C++ dan dikembangkan dengan tujuan implementasi utama pada bahasa C++. Terdapat binding untuk beberapa bahasa pemrograman high level seperti Python, Java, dan MATLAB. Walaupun awalnya OpenCV dikembangkan untuk aplikasi CPU intensive, namun saat ini akselerasi menggunakkan CUDA untuk mengakselerasi kinerja algoritme. Akselerasi juga dapat dilakukan menggunakkan GPU berbasis Open-CL seperti milik AMD dan Intel.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

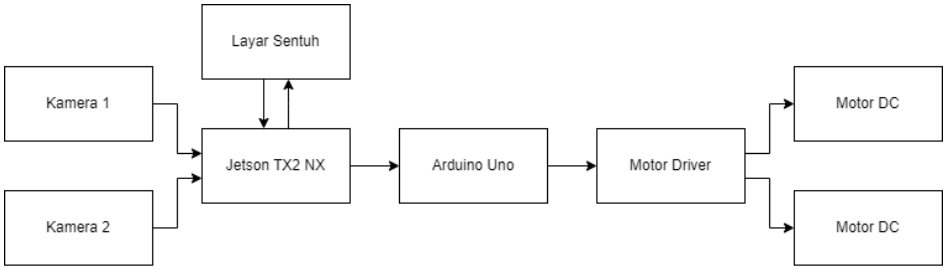
5.1 Perancangan Sistem

Dalam perancangan sistem akan dibahas mengenai perancangan yang dilakukan untuk memastikan sistem berjalan dengan baik dan sesuai dengan tujuan sistem. Diagram alir dan diagram blok akan digunakkan untuk menggambarkan perancangan sistem yang kemudian akan dilakukan, perancangan ini mencakup perancangan perangkat keras atau *hardware* dan perancangan perangkat lunak atau *software*. Blok diagram pada Gambar 5.1 menunjukkan keseluruhan proses



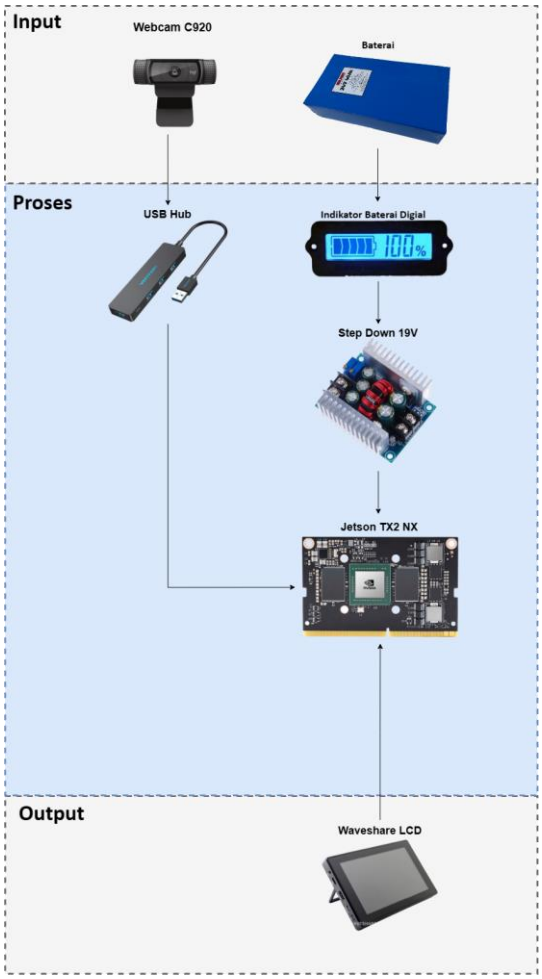
Gambar 5.1 Perancangan Sistem

Pada Gambar 5.1 dijelaskan input yang digunakkan pada sistem didapatkan dari perangkat kamera yang akan menangkap citra. Citra yang ditangkap merupakan citra digital dengan warna RGB yang berisi keseluruhan kepala pengguna. Baterai juga digunakkan sebagai sumber daya bagi Jetson TX2 yang kemudian akan digunakkan sebagai dapur pacu dari sistem. Pada blok diagram bagian proses dibagi menjadi 2 yaitu *training model* yang dikerjakan menggunakkan PC dan proses klasifikasi arah pandang yang akan dikerjakan pada Jetson TX2. Proses *training model* akan menghasilkan model CNN dengan *weight* dan *bias* yang sudah terkonfigurasi. Model inilah yang kemudian akan digunakkan untuk klasifikasi arah pandangan pada Jeton TX2. Proses *training model* hanya perlu diulang sekali saja hingga mendapatkan akurasi dan *loss* yang diinginkan dari model yang diperoleh setelah training. Kemudian pada proses klasifikasi arah pandang *input* dari kamera akan dilakukan *cropping* dengan facial landmark dan dimasukkan kedalam CNN yang sudah dilatih sebelumnya. Terakhir adalah pada bagian *output* dimana hasil klasifikasi akan digambarkan pada antarmuka pengguna di layar.



Gambar 5.2 Blok Diagram Sistem Kursi Roda Pintar

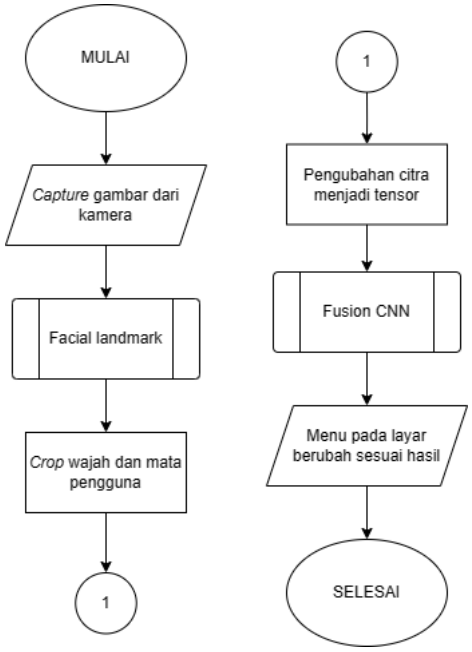
Pada Gambar 5.2 ditunjukkan blok diagram sistem kursi roda pintar yang menggambarkan rancangan perangkat keras yang akan digunakkan. Sub-sistem yang di rancang pada penelitian ini akan menerima input berupa gambar RGB yang ditangkap oleh Kamera 1. Gambar tersebut akan diproses oleh Jetson TX2 NX menggunakan metode *pre-preprocessing* dan di klasifikasikan dengan algortima yang akan dibahas pada Gambar 5.4. Kemudian hasil dari proses klasifikasi tersebut akan ditunjukkan pada layar berupa bagian yang dilihat oleh pengguna beserta dengan menu lainnya yang tersedia pada sistem. Dari blok diagram yang sudah maka dirancang skematik seperti pada Gambar 5.3. Dari skematik yang diberikan koneksi antara komponen dijelaskan pada Tabel 5.1.



Gambar 5.3 Skematik Sistem

Tabel 5.1 Koneksi antara komponen

Komponen	Jenis Hubungan	Port
USB Hub	USB	Port USB 1 pada Jetson TX2
Kamera Logitech C920	USB	Port USB 1 pada USB hub
Step Down 19V	Power	Port power pada Jetson TX2
Voltmeter Digital	Power	Pin input pada Step Down
Baterai	Power	Pin Input pada Voltmeter
Waveshare LCD	HDMI	Port HDMI pada Jetson TX2



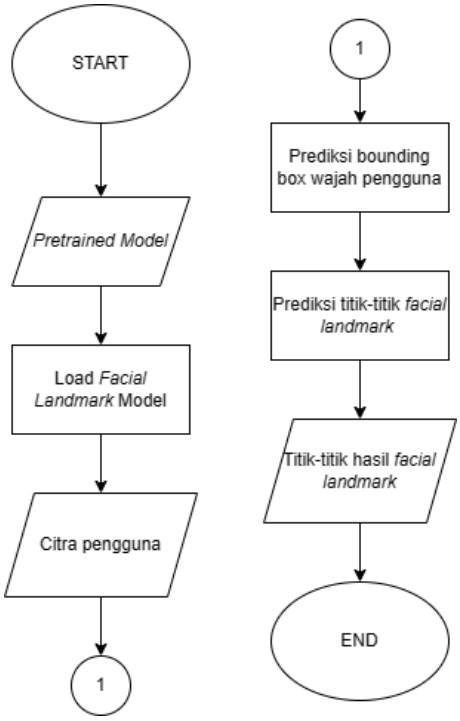
Gambar 5.4 Diagram Alir Program Utama

Pada Gambar 5.4 dijelaskan lebih lanjut mengenai algoritme yang digunakkan untuk melakukan *preprocessing* dan klasifikasi pada gambar yang ditangkap pada Jetson TX2 NX. Gambar yang sudah ditangkap akan diubah menjadi *grayscale* dan diaplikasikan algoritme *facial landmark* untuk menemukan titik-titik pada wajah yang kemudian akan digunakkan untuk proses *cropping* pada wajah dan mata pengguna. Gambar yang sudah di *crop* akan dilakukan proses *resize* sehingga menghasilkan citra mata berukuran 100x50 dan citra wajah berukuran 100x100. Kedua citra tersebut kemudian akan diubah menjadi tensor yang kemudian menjadi input bagi algoritme *Fusion CNN* yang akan mengklasifikasikan pandangan mata pengguna berdasarkan menu yang ada pada layar.

5.1.1 Perancangan *Facial Landmark*

Berdasarkan kebutuhan fungsional pertama yaitu sistem dapat menjalankan algoritme *facial landmark* untuk menemukan wajah pengguna, maka sistem dirancang dengan menggunakkan pengimplementasian algortima *facial landmark*

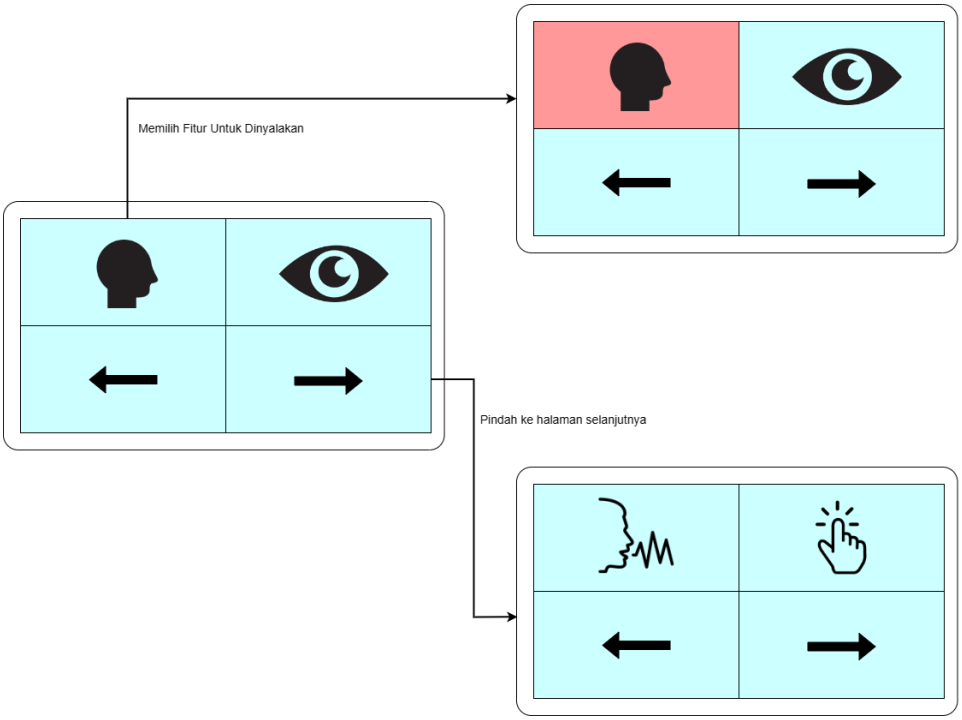
dari *library* dlib dan *pretrained* model yang disediakan oleh dlib. Pengimplementasian ini menggunakan 68 titik *facial landmark* yang tersebar pada fitur muka yang dikenali oleh algoritme. Untuk memproses algoritme ini digunakkan juga Jetson TX2NX dan kapabilitas akselerasi CUDA yang dimiliki perangkat tersebut untuk mendapatkan waktu pemrosesan yang singkat. *Flowchart* yang menggambarkan proses *facial landmark* dapat dilihat pada Gambar 5.5.



Gambar 5.5 *Flowchart Facial Landmark*

5.1.2 Perancangan GUI

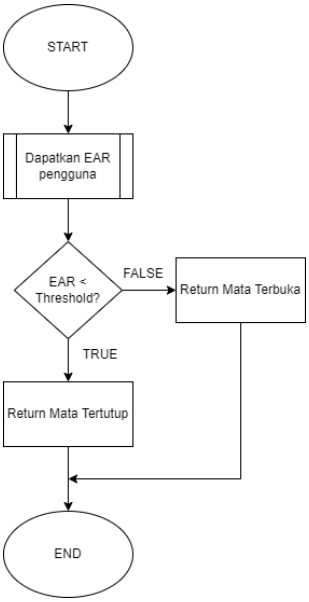
Berdasarkan kebutuhan fungsional kedua yaitu Sistem dapat menunjukkan antarmuka grafis untuk pengguna kursi roda pintar dan memberi tahu pengguna hasil deteksi arah pandangan melalui antarmuka grafis maka dirancang antarmuka grafis yang memiliki 4 tombol atau menu besar yang dapat dengan mudah menunjukkan menu pada pengguna. Pemilihan ukuran ini juga mempertimbangkan keterlihatan menu bagi pengguna. Menu juga akan digambarkan menggunakkan ikon yang merepresentasikan menu yang dapat dipilih. Untuk menu (fitur) yang saat ini sedang aktif akan ditandakan dengan tombol berwarna merah dan untuk arah pandangan pengguna akan ditandakan dengan tombol berwarna kuning. Gambar desain dari GUI dapat dilihat pada Gambar 5.6.



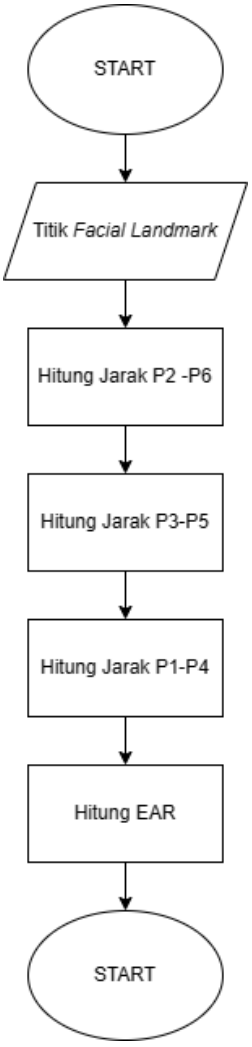
Gambar 5.6 Rancangan GUI yang akan dibuat

5.1.3 Perancangan *Blink Detection*

Berdasarkan kebutuhan fungsional ketiga yaitu Sistem dapat mendeteksi apabila pengguna dengan sengaja mengedipkan mata untuk memilih menu yang ditunjukkan pada layar maka dirancang juga algoritme untuk mengenali kedipan mata pengguna. Algoritme yang dipakai menggunakkan EAR atau *Eye Aspect Ratio* untuk mengenali apabila mata pengguna menutup ataupun terbuka. Kemudian sistem akan menentukan pilihan pengguna berdasarkan arah pandangan terakhir pengguna dan waktu antara pengguna menutup mata dan membuka mata. Ditentukan *threshold* durasi mata pengguna menutup untuk memilih menu adalah 1 detik. *Flowchart* perancangan algoritme dapat dilihat pada Gambar 5.7. Dan *flowchart* perancangan algoritme EAR dapat dilihat pada Gambar 5.8.



Gambar 5.7 *Flowchart blink detection*



Gambar 5.8 *Flowchart* Sub-Program EAR

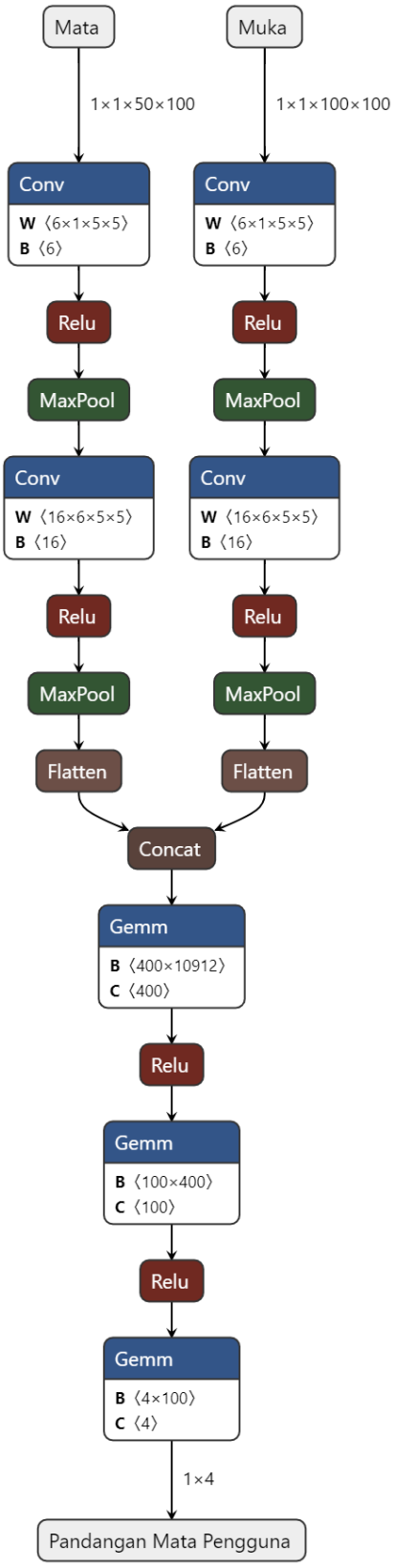
5.1.4 Perancangan *Fusion CNN*

Berdasarkan kebutuhan fungsional keempat yaitu Sistem dapat melakukan klasifikasi citra mata dan wajah untuk menentukan arah pandangan pengguna maka digunakkan algoritme *Fusion CNN* untuk melakukan pemrosesan terhadap citra wajah dan mata yang didapatkan sistem. Algoritme ini akan digunakkan untuk melakukan klasifikasi arah pandangan pengguna terhadap layar.

Gambaran mengenai arsitektur *Fusion CNN* dapat dilihat pada GAMBAR. Pada bagian atas adalah input yang berupa citra mata dan muka pengguna dengan ukuran masing masing 100 x 50 dan 100 x 100. Kemudian citra tersebut akan diproses menggunakkan blok konvolusi dan *Max Pooling* secara terpisah. Ada 2 blok yang disediakan untuk masing-masing citra, hasil dari proses ini adalah *feature map* yang kemudian akan di *flatten* atau di ubah bentuknya dari matrix 2D menjadi matrix 1D. Kedua matriks yang didapatkan dari proses sebelumnya kemudian akan di *concatenate* atau disatukan dengan menggabungkan kedua matriks tersebut sehingga bagian depan matriks adalah *feature map* mata pengguna dan selanjutnya adalah *feature map* wajah pengguna. Hasil *concatenation* tadi akan dijadikan input kepada GEMM atau *general matrix multiplication*, proses ini biasa juga disebut dengan sebutan *fully connected* atau *dense layer*. Proses GEMM dilakukan untuk melakukan klasifikasi inputnya menjadi keempat kelas yang sudah ditentukan sebelumnya yaitu kiri aras, kanan atas, kiri bawah, kanan bawah.

$$ReLU(x) = \max(0, a) \tag{5.1}$$

Setiap *layer* GEMM akan menggunakkan fungsi aktivasi ReLU untuk mengurangi *computation load* dimana ReLU hanya perlu menghitung nilai yang lebih besar antara 0 dan *a* dimana *a* adalah masukan dari fungsi ReLU, hal ini ditunjukkan lebih jelas dari persamaan (5.1) yang menggambarkan keluaran dari ReLU. Pada akhir dari *Fusion CNN* terdapat 1 *layer* GEMM tanpa fungsi aktivasi atau linear dimana output dari tiap *neuron* GEMM tidak akan di masukan kedalam fungsi aktivasinya. Pada Gambar 5.9 dapat dilihat arsitektur lengkap dari *Fusion CNN* yang divisualisasikan menggunakkan netron.



Gambar 5.9 Arsitektur *Fusion CNN*

5.1.5 Perancangan Sistem Dapat Menangkap Citra Wajah dan Mata

Berdasarkan kebutuhan fungsional yang kelima yaitu sistem dapat menangkap citra wajah dan mata pegguna maka kamera akan dirancang untuk menghadap ke arah pengguna. Kamera akan ditempatkan tepat di atas layar dengan arah sejajar dengan layar berjarak 30-50cm dari wajah pengguna sehingga kamera dapat dengan baik melihat pengguna dengan baik. Penempatan ini juga mempertimbangkan akurasi pada algoritme *Fusion CNN* dan *facial landmark* dimana wajah pengguna dapat terdeteksi dengan baik apabila berapa dekat dengan kamera dan wajah pengguna ada di tengah *frame* dari kamera. Gambar desain dari *hardware* sistem dapat dilihat pada Gambar 5.10.

Gambar 5.10 Desain *hardware* sistem

5.2 Implementasi

Setlah dilakukan perancangan pada sub-bab 5.1 maka dapat dilakukan implementasi sistem secara keseluruhan. Pengimplementasian sistem ini mencakup pengimplementasian secara *hardware* dan *software*.

5.2.1 Implementasi *Hardware*

Pengimplementasian pada sisi hardware akan dilakukan dengan menggunakkan perangkat Jetson TX2NX, layar sentuh 7 inci, dan *webcam* Logitech C920. Alasan dari pemilihan perangkat yang digunakkan dalam pengimplementasian sistem ini tertera pada bab 4, sub-bab 4.5. Pada Gambar 5.11 ditunjukkan ketika perangkat sudah terpasang pada sistem kursi roda pintar. Pada gambar dikiri adalah tampak sistem dari belakang dimana layar menghadap ke pengguna dan sedang melakukan *inference*, sementara gambar di kanan menunjukkan kursi dari tampak depan.

Gambar 5.11 Implementasi Hardware

5.2.2 Implementasi *Software*

Implementasi *software* akan menjelaskan mengenai pengimplementasian setiap bagian dari algoritme dan perangkat lunak yang akan berjalan pada sistem. Bagian ini akan mencakup *preprocessing,* proses klasifikasi dengan *Fusion CNN*, proses deteksi kedipan, dan GUI yang sudah diimplementasikan.

5.2.2.1 Implementasi Facial Landmark

Pada proses ini diimplementasikan kode untuk melakukan proses *facial landmark, cropping,* dan *resize* pada citra yang didapatkan dari pengguna. Pada proses *facial landmark* digunakkan *library dlib* dan model *pretrained* untuk memperoleh hasil titik-titik penanda pada wajah pengguna.

Algoritme 5.1: *facial landmark*

```
1   def getRect():
2       _, image = cap.read()
3       while image is None:
4           image = cap.read()
5       gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
6       rects = detector(gray, 0)
7       return (rects, gray, image)
    …
43  def getEyes(rects, gray, image, isShape = False):
44      image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
45      for (i, rect) in enumerate(rects):
46          shape = predictor(gray, rect)
47          shape = face_utils.shape_to_np(shape)
```

Kode dapat dideskripsikan sebagai berikut,

- Kode baris 2-4 melakukan pengambilan gambar menggunakkan kamera yang sebelumnya sudah di-*set*

- Kode baris 5 melakukan pengubahan gambar yang diambil menjadi *grayscale*
- Kode baris 6 digunakkan untuk mendapatkan *bounding box* dari wajah yang akan dideteksi oleh sistem
- Kode baris 45-47 akan mencari titik-titik *facial landmark* menggunakkan *pretrained model* sebagai *predictor*-nya

Pada kode program diatas program akan melakukan penangkapan citra menggunakkan kamera dengan fungsi cap.read(). Kemudian citra yang ditangkap akan dikonversi menjadi *grayscale* dan dilakukan deteksi wajah menggunakkan *detector* yang ada pada dlib menghasilkan variabel *rects* yang berisi koordinat wajah pada citra. Kemudian *rects* akan dimasukkan kedalam *predictor* yang menggunakkan model *pretrained* untuk menemukan titik-titik *facial landmark* pada wajah pengguna yang akan disimpan dalam bentuk matriks 2D dari koordinat titik-titik *facial landmark*.

Algoritme 5.2: *cropping* dan *resize*

```
 1  def getEyes(rects, gray, image, isShape = False):
 2      image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
 3      for (i, rect) in enumerate(rects):
 4          shape = predictor(gray, rect)
 5          shape = face_utils.shape_to_np(shape)
 6          if isShape:
 7              return shape[36: 42]
 8          image = image[shape[38][1]-10:shape[42][1]+10,
                          shape[37][0]-20:shape[40][0]+20]
 9          try:
10              image = cv2.resize(image, (100,50))
11          except:
12              return -1
13
```

Kode dapat dideskripsikan sebagai berikut,

- Kode baris 8 akan dilakukan cropping menggunakkan titik-titik yang sudah didapatkan dari *facial landmark*.
- Kode baris 9-12 akan mencoba untuk melakukan resize pada gambar. Apabila gambar tidak dapat di resize, maka akan mengembalikan nilai -1.

Kemudian titik-titik dari *facial landmark* tadi akan digunakkan sebagai acuan untuk melakukan *cropping* pada kode nomor 8. Titik pada posisi 38 dan 42 akan digunakkan untuk ukuran horizontal untuk citra mata dan posisi 37 dan 40 untuk ukuran vertikal. Sementara untuk wajah akan digunakan variabel *rect* yang didapatkan sebelumnya. Untuk resize akan dilakukan dengan kode nomor 10 dimana citra mata akan di *resize* menjadi 100 x 50 dan citra wajah menjadi 100 x 100.

5.2.2.2 Implementasi Fusion CNN

Untuk melakukan klasifikasi dengan algoritme *Fusion CNN* maka arsitekturnya harus didefinisikan terlebih dahulu. Arsitektur *Fusion CNN* yang diimplementasikan didasarkan pada perancangan di sub-bab 5.1.5. Dengan *framework* PyTorch maka mendefinisikan suatu arsitektur *Neural Network* dapat menggunakkan klas turunan dari klas torch.nn.Module. Pengkodean arsitektur *Fusion CNN* pada PyTorch ditunjukkan pada kode dibawah.

Algoritme 5.3: Pendefinisian *Fusion CNN*

```
1  class DualModel(nn.Module):
2      def __init__(self) -> None:
3          super().__init__()
4          self.layer1 = nn.Sequential(
5              nn.Conv2d(1, 6, kernel_size=5, stride=1, padding=0),
6              nn.BatchNorm2d(6),
7              nn.ReLU(),
8              nn.MaxPool2d(kernel_size = 2, stride = 2))
9          self.layer2 = nn.Sequential(
10             nn.Conv2d(6, 16, kernel_size=5, stride=1, padding=0),
11             nn.BatchNorm2d(16),
12             nn.ReLU(),
13             nn.MaxPool2d(kernel_size = 2, stride = 2))
14         self.layer12 = nn.Sequential(
15             nn.Conv2d(1, 6, kernel_size=5, stride=1, padding=0),
16             nn.BatchNorm2d(6),
17             nn.ReLU(),
18             nn.MaxPool2d(kernel_size = 2, stride = 2))
19         self.layer22 = nn.Sequential(
20             nn.Conv2d(6, 16, kernel_size=5, stride=1, padding=0),
21             nn.BatchNorm2d(16),
22             nn.ReLU(),
23             nn.MaxPool2d(kernel_size = 2, stride = 2))
24         self.flatten = nn.Flatten()
25         self.fc = nn.Linear(10912, 400)
26         self.relu = nn.ReLU()
27         self.fc1 = nn.Linear(400, 100)
28         self.relu1 = nn.ReLU()
29         self.fc2 = nn.Linear(100, 5)
```

Kode diatas dapat dideskripsikan sebagai berikut,

- Kode baris 4-8 mendefinisikan blok CNN *layer* pertama untuk mata
- Kode baris 9-13 mendefinisikan blok CNN *layer* pertama untuk wajah
- Kode baris 14-18 mendefinisikan blok CNN *layer* kedua untuk mata
- Kode baris 19-23 mendefinisikan blok CNN *layer* kedua untuk wajah

- Kode baris 24 mendefinisikan *layer flatten*
- Kode baris 25-29 mendefinisikan *fully connected layer* atau *dense layer*

Pada kode diatas variabel layer1, layer2, layer12, dan layer22 mendefinisikan *layer* konvolusional yang akan digunakkan dimana layer1 dan layer2 digunakkan untuk citra mata dan layer12 dan layer22 digunakkan untuk citra wajah. Kemudian operasi *flatten* untuk tiap citra didefinisikan hanya sekali dalam variabel flatten. Untuk operasi GMM yang dilihat pada sub-bab 5.1.5 diimplementasikan dalam variabel fc, fc1, dan fc2 dengan fungsi aktivasi ReLU yang tersimpan pada variabel relu dan relu2.

Algoritme 5.4: *Inference* dengan *Fusion CNN*

```
1 output = model(mata, muka)
2 prediction = torch.max(output, 1)
```

Kode diatas dapat dideskripsikan sebagai berikut,

- Kode baris 1 memanggil model untuk melakukan *inference* citra mata dan muka sebagai input
- Kode baris 2 mencari prediksi dengan nilai paling tinggi.

Untuk proses *inference* akan dilakukan dengan menjalankan kode dibawah. Dimana output akan mendapatkan keluaran langsung dari *Fusion CNN* yang memiliki 4 nilai, yaitu hasil akhir perhitungan dari keempat neuron pada *layer* terakhir. Variabel mata dan muka adalah input yang berupa hasil *preprocessing* citra pengguna yang sebelumnya diubah terlebih dahulu menjadi tensor. Keluaran pada output kemudian akan dijadikan kelas yang diprediksikan berdasarkan nilai terbesar pada *axis* 1 di variabel output menggunakkan fungsi torch.max().

5.2.2.3 Implementasi Blink Detection

Dalam pengimplementasiannya *blink detection* atau deteksi kedipan akan menggunakkan titik-titik *facial landmark* yang diperoleh dari kode pada sub-bab 5.2.2.1, algoritme 5.1. Titik-titik ini akan dijadikan acuan untuk mendapatkan EAR dari citra mata pengguna.

Algoritme 5.5: *Blink Detection*

```
1 def isBlinking(eye, thresh):
2     return True if eye_aspect_ratio(eye) < thresh - 0.3 else False
3
4 def eye_aspect_ratio(eye):
5     p2_minus_p6 = dist.euclidean(eye[1], eye[5])
6     p3_minus_p5 = dist.euclidean(eye[2], eye[4])
7     p1_minus_p4 = dist.euclidean(eye[0], eye[3])
8     ear = (p2_minus_p6 + p3_minus_p5) / (2.0 * p1_minus_p4)
9     return ear
```

Kode diatas dapat dideskripsikan sebagai berikut,

- Kode baris 2 akan melakukan pengecekan apabila EAR dari pengguna kurang dari *threshold* yang ditentukan sebelumnya
- Kdoe baris 5-9 akan melakukan kalkulasi jarak antara titik-titik mata pada *facial landmark* untuk menemukan EAR

Titik-titik mata yang sebelumnya sudah didapatkan akan dilakukan perhitungan EAR oleh algoritme 5.5. Algoritme ini akan menghitung jarak eucledian dari titik 1 dan 5 (kode nomor 5), jarak eucledian dari titik 2 dan 4 (kode nomor 6), dan jarak eucledian dari titik 0 dan 3 (kode nomor 7). Kemudian jarak titik 1-5 dan titik 2-4 akan dijumlahkan dan dibagi dengan 2 kali jarak 0-3 untuk mendapatkan *aspect ratio* dari mata atau EAR. Nilai ini yang kemudian akan digunakkan untuk menentukan apabila seseorang berkedip berdasarkan *threshold* yang sebelumnya sudah ditentukan dengan mengambil rata-rata EAR mata pengguna ketika terbuka dikurangi 0.3. konstanta 0.3 digunakkan untuk meningkatkan akurasi *blink detection* sehingga apabila mata pengguna menyipit (tidak tertutup) masih dapat dikategorikan sebagai mata yang terbuka.

5.2.2.4 Implementasi Graphical User Interface

Pendefinisian *graphical user interface* atau GUI pada *framework* pysimpleGUI menggunakkan matriks yang sudah didefinisikan terlebih dahulu. Untuk mendefinisikan tombol dan elemen lain yang akan ditampilkang digunakkan klas dari *library* PySimpleGUIQt seperti klas button untuk tombol dan text untuk tulisan. Pada tiap pendefinisiannya digunakkan beberapa parameter seperti *path* ke gambar yang akan ditunjukkan (untuk ikon pada tombol), key sebagai *identifier* ketika melakukan *event handling*, dan ukuran dari elemen. Pendefinisian GUI yang digunakkan pada sistem ada pada kode dibawah

Algoritme 5.6: Pendefinisian GUI

```
1  import PySimpleGUIQt as sg
2
3  pageSatu = [
4      [
5          sg.Button(image_filename= 'kepala.png',
6                    enable_events=True, key="-1-",
7                    size_px=(540,360),
8                    button_color = ("white", "blue")),
9          sg.VSeperator(),
10         sg.Button(image_filename= 'mata.png',
11                   enable_events=True,
12                   key="-2-", size_px=(540,360),
13                   button_color = ("white", "blue"))
14     ]
15 ]
16 pageDua = [
17     [
```

```
18          sg.Button(image_filename= '',
19                    enable_events=True,
20                    key="-3-",
21                    size_px=(540,360),
22                    button_color = ("white", "blue")),
23          sg.VSeperator(),
24          sg.Button(image_filename= 'touch.png',
25                    enable_events=True,
26                    key="-4-",
27                    size_px=(540,360),
28                    button_color = ("white", "blue"))
29      ],
30  ]
31  layout = [
32      [
33          sg.Column(pageSatu,
34                    key='-pageSatu-',
35                    visible=True),
36          sg.Column(pageDua,
37                    key='-pageDua-',
38                    visible= False)
39      ],
40      [
41          sg.HSeperator()
42      ],
43      [
44          sg.Button(button_text="KIRI",
45                    enable_events=True,
46                    key="-KIRI BAWAH-",
47                    size_px=(540,360),
48                    button_color = ("white", "blue")),
49          sg.VSeperator(),
50          sg.Button(button_text="KANAN",
51                    enable_events=True,
52                    key="-KANAN BAWAH-",
53                    size_px=(540,360),
54                    button_color = ("white", "blue"))
55      ],
56  ]
```

Kode diatas dapat dideskripsikan sebagai berikut,

- Kode baris 3-15 mendefinisikan halaman pertama pada GUI dengan tombol pada kode baris 5-8 dan kode baris 10-13.

- Kode baris 16-30 mendefinisikan halaman kedua dari GUI dengan tombol pada kode baris 18-22 dan kode baris 24-28
- Kdoe baris 31-56 mendefinisikan *layout* dari tampilan yang akan ditunjukkan kepada pengguna.

BAB 6 PENGUJIAN

Pada bab ini akan dibahas lebih lanjut mengenai pengujian yang telah dilakukan pada penelitian ini. Pembahasan pengujian pada penelitian ini akan meliputi hasil pengujian yang sudah dilakukan dan analisa mengenai hasil pengujian tersebut. Akan dijelaskan juga mengenai prosedur pengujian yang dilakukan.

6.1 Hasil Pengujian

Sub-bab hasil pengujian akan membahas lebih lanjut mengenai prosedur pengujian yang dilakukan dan hasil yang didapatkan. Parameter pengujian yang dilakukan didasarkan pada kebutuhan fungsional yang sudah dijelaskan pada bab sebelumnya.

6.1.1 Pengujian GUI

Pengujian GUI akan menguji keberhasilan pengimplementasian GUI pada sistem. Parameter yang akan diuji pada pengujian ini adalah fungsionalitas GUI dalam menjalankan fitur yang diinginkan dan integrasi GUI dengan algoritme *Fusion CNN*. Integrasi yang dimaksud adalah GUI dapat menampilkan hasil pendeteksian arah pandangan pengguna. Arah pandangan pengguna akan ditampilkan dengan warna kuning sementara fitur yang berjalan akan berwarna merah. Pada Gambar 6.2 ditunjukkan pengujian GUI pada sistem. Pada Tabel 6.1 ditunjukkan juga hasil pengujian GUI.

Pengujian dilakukan dengan langkah-langkah berikut,

1. Subjek pengujian duduk di kursi roda pintar dan menyesuaikan layar hingga dirasa nyaman untuk pengguna.
2. Subjek akan diminta untuk melihat ke arah yang sudah ditentukan oleh penguji.
3. Respon sistem akan dinilai setelah 1 detik subjek menatap ke arah yang diminta.
4. Apabila respon sistem tidak sesuai maka hasilnya akan ditulis SALAH, apabila respon sistem sesuai maka hasilnya akan ditulis BENAR.

Gambar 6.1 GUI Menunjukkan Arah Pandangan Pengguna

Tabel 6.1 Hasil Pengujian GUI

Pengujian GUI		
Arah Pandangan (Ground Thruth)	**Hasil Deteksi**	**Output Pada Layar**
Kanan Atas	0	Kanan Atas
Kanan Atas	0	Kanan Atas
Kanan Atas	0	Kanan Atas
Kanan Atas	0	Kanan Atas
Kiri Atas	1	Kiri Atas
Kiri Atas	1	Kiri Atas
Kiri Atas	3	Kiri Bawah
Kiri Atas	1	Kiri Atas
Kanan Bawah	2	Kanan Bawah
Kanan Bawah	2	Kanan Bawah
Kanan Bawah	0	Kanan Atas
Kanan Bawah	2	Kanan Bawah
Kiri Bawah	3	Kiri Bawah
Kiri Bawah	3	Kiri Bawah
Kiri Bawah	3	Kiri Bawah
Kiri Bawah	3	Kiri Bawah

6.1.2 Pengujian Blink Detection

Pengujian algoritme *blink detection* akan menguji efektifitas algoritme dalam mendeteksi apabila pengguna mengedipkan matanya dengan sengaja untuk memilih menu yang ada pada layar. Untuk membedakan kedipan yang disengaja dan tidak disengaja dibedakan atas waktu antara mata menutup dan terbuka. Batas waktu yang dibutuhkan untuk menentukan bahwa kedipan disengaja dijelaskan pada kebutuhan fungsional adalah sebesar 1 detik. hasil pengujian yang didapatkan dituliskan pada Tabel 6.2.

Pengujian dilakukan dengan langkah-langkah berikut,

1. Subjek pengujian duduk di kursi roda pintar dan menyesuaikan layar hingga dirasa nyaman untuk pengguna.
2. Subjek akan diminta untuk berkedip dengan durasi lebih dari atau sama dengan 1 detik.
3. Respon sistem akan dinilai setelah subjek selesai mengedip.
4. Apabila respon sistem subjek mengedip maka dituliskan BENAR, apabila respon sistem subjek tidak mengedip maka dituliskan SALAH.

Tabel 6.2 Hasil Pengujian *Blink Detection*

Pengujian *Blink Detection*	
Pengujian Ke-	**Deteksi**
1	Benar
2	Benar
3	Benar
4	Benar
5	Benar
6	Benar
7	Benar
8	Benar
9	Salah
10	Benar

6.1.3 Pengujian *Facial Landmark*

Pengujian *time constraint* akan menguji waktu yang dibutuhkan oleh sistem untuk menjalankan algoritme *facial landmark* untuk menemukan wajah pengguna. Pengujian dilakukan dengan menjalankan algoritme *facial landmark* pada sistem dan waktu dihitung sejak algoritme dijalankan hingga titik-titik *facial landmark* didapatkan. Untuk pengujian *time constraint* dilakukan 10 kali pengujian untuk mendapatkan hasil rata rata waktu komputasi algoritme *facial landmark* dan menguji kestabilan sistem melalui perhitungan standar deviasi hasil pengujian. Dari pengujian yang sudah dilakukan didapatkan hasil yang ditunjukkan pada Tabel 6.3

Pengujian dilakukan dengan langkah-langkah berikut,

1. Subjek pengujian duduk di kursi roda pintar dan menyesuaikan layar hingga dirasa nyaman untuk pengguna.
2. Sistem akan menjalankan algoritme *facial landmark* untuk menemukan wajah dan mendapatkan titik-titik *facial landmark*.
3. Waktu komputasi dihitung dari saat sistem memulai proses pencarian wajah hingga sistem mendapatkan titik-titik *facial landmark*.

Tabel 6.3 Hasil Pengujian *Time Constraint Facial Landmark*

Time Constraint *Facial Landmark*	
Pengujian Ke-	**Waktu Komputasi**
1	0.1497
2	0.1313
3	0.1317
4	0.139
5	0.1311
6	0.132
7	0.1296
8	0.1344
9	0.1242
10	0.126
Rata- Rata	**0.1329**
Std Dev	**0.007179755**

6.1.4 Pengujian Fusion CNN

Pada bagian ini akan diuji performa dari algoritme *Fusion CNN*. Beberapa parameter yang akan diujikan dalam pengujian ini adalah pengaruh jumlah *epoch* terhadap hasil akurasi dan *loss function* ketika proses training, akurasi dari model yang telah dibuat, waktu komputasi, dan pengujian penggunaan daya listrik serta daya komputasi pada saat proses *inference*.

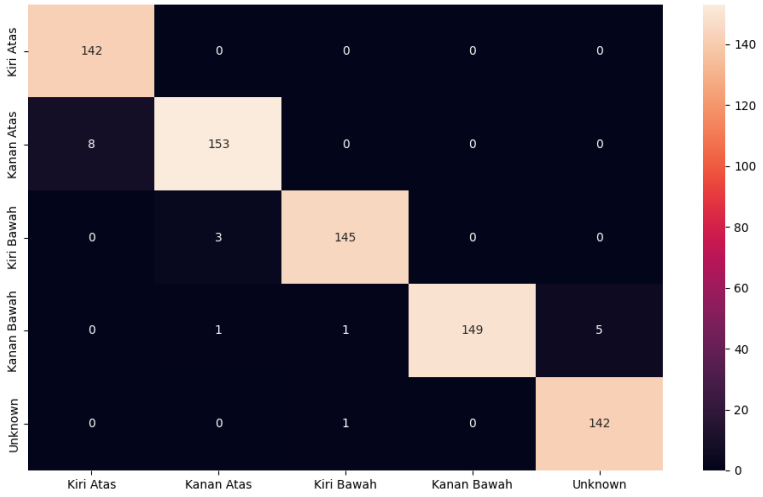
Pengujian pada proses training akan melihat pengaruh jumlah *epoch* terhadap nilai akurasi dan *loss function* yang didapatkan. Proses training dilakukan dengan menggunakkan *train loop* standard yang sudah didefinisikan oleh *framework* Pytorch dengan *batch size* sebesar 2100 step. Pengujian *training* dilakukan sebanyak 30 epoch atau hingga tidak ada perubahan signifikan didapatkan dari *loss function* dan akurasi test model. Hasil dari pengujian ini di berikan pada Tabel 6.4.

Tabel 6.4 Hasil Pengujian *Training* Model

epoch	loss	accuracy
1	0.011543034	92.93333
2	0.01947239	95.06667
3	0.003208967	95.73333
4	0.003886647	95.33333
5	0.003026074	95.46667
6	0.000589674	96.4

7	0.00065901	96.4
8	0.000238629	96
9	0.00026461	96.26667
10	5.29E-05	95.73333
11	0.000366382	96
12	0.000382231	96.26667
13	5.72E-05	96.4
14	0.000112766	96
15	7.58E-05	95.73333
16	5.96E-05	96.13333
17	2.92E-05	95.33333
18	0.000166998	96
19	4.58E-05	96
20	1.86E-05	94.93333
21	6.29E-05	95.73333
22	3.27E-05	95.46667
23	4.65E-05	95.06667
24	4.21E-05	95.06667
25	1.73E-05	96
26	1.76E-05	95.33333
27	8.34E-06	95.46667
28	4.86E-05	95.33333
29	2.38E-06	95.6
30	7.03E-06	95.6

Model yang didapatkan dari proses *training* sebelumnya akan dilakukan pengujian dengan *dataset* yang sudah sebelumnya dipisahkan. Pengujian ini akan menghitung akurasi, presisi, *recall*, dan *specitivity.* Pengujian akan dilakukan 10 kali untuk memberikan gambaran mengenai stabilitas model dan memberikan rata-rata tiap parameter. Hasil dari pengujian ini ditunjukkan pada *confusion matrix* pada Gambar 6.2 *Confusion Matrix* Hasil Pengujian Training.



Gambar 6.2 *Confusion Matrix* Hasil Pengujian Training

Pada fase *deployment* sistem perlu bekerja dengan batasan waktu 150ms per prediksi untuk mendapatkan performa *realtime.* Untuk itu diperlukan adanya pengujian waktu komputasi dimana pengujian akan dilakukan pada perangkat keras yang akan digunakkan pada waktu *deployment*. Algoritme *Fusion CNN* akan diuji dengan menggunakkan akselrasi CUDA dan tanpa akselerasi CUDA. Hasil pengujian dapat dilihat pada Tabel 6.5.

Tabel 6.5 Hasil Pengujian *Deployment*

Pengujian Penggunaan CUDA		
Pengujian Ke-	**Tanpa CUDA (CPU) (Detik)**	**Dengan CUDA (Detik)**
1	0.1971	0.1954
2	0.2154	0.1583
3	0.1798	0.1491
4	0.2062	0.1628
5	0.208	0.1557
6	0.1909	0.15322
7	0.2043	0.1611
8	0.2069	0.156
9	0.2017	0.1532
10	0.2072	0.1623
Rata- Rata	**0.20175**	**0.160712**
Std Dev	**0.010160189**	**0.012956461**

Dengan parameter yang mendapatkan hasil terbaik pada pengujia sebelumnya akan dilakukan pengujian penggunaan daya komputasi dan daya listrik sistem.

daya komputasi akan diukur dengan menggunakkan system monitor yang disediakan dari OS Nvidia Jetson TX2NX dan pengujian daya listrik akan menggunakkan *smart plug* yang dapat menguji penggunaan daya listrik. Hasil pengujian daya komputasi ditunjukkan di dalam Tabel 6.6 dan daya listrik ditunjukkan pada Tabel 6.7

Tabel 6.6 Hasil Pengujian Daya Komputasi

Pengujian Penggunaan Daya Komputasi		
System Resource	**Idle**	***Fusion CNN***
Overall CPU	5%	26%
RAM	1.4 GiB	3.2 GiB

Tabel 6.7 Hasil Pengujian Daya Listrik

Pengujian Penggunaan Daya Listrik		
Pengujian Ke-	**Idle (Watt)**	***Fusion CNN* (Watt)**
1	8.40	11.70
2	9.00	11.80
3	8.90	12.00
4	9.10	12.20
5	8.90	12.20
6	8.90	12.60
7	9.30	11.50
8	8.90	11.80
9	9.10	12.40
10	9.30	11.80
Rata- Rata	**8.98**	**12.00**

6.1.5 Pengujian Penangkapan Citra Mata dan Muka

Pada pengujian ini akan diuji fungsionalitas sistem dalam kemampuannya untuk menangkap citra mata dan muka pengguna. Akurasi sistem akan diuji ketika melakukan *inference* terhadap ketinggian mata pengguna. Pengujian dilakukan dengan memberikan arahan kepada partisipan untuk melihat ke salah satu tombol pada layar dan melihat respon sistem. Hasil pengujiannya dapat dilihat pada Tabel 6.8. Pada Gambar 6.3 diperlihatkan contoh penangkapan citra mata dan muka pengguna, pada gambar diatas adalah citra mata dan muka pengguna ketika sistem dapat menemukan muka melalui facial landmark, sementara gambar dibawah merupakan hasil yang didapatkan ketika sistem tidak dapat menemukan muka pengguna.

Tabel 6.8 Hasil Pengujian Penangkapan Citra

Pengujian Perangkapan Citra Mata dan Muka		
Posisi	**Subjek**	**Hasil**
Tinggi	1	BENAR
		BENAR
		BENAR
		BENAR
		SALAH
		BENAR
		BENAR
		BENAR
		SALAH
		BENAR
	2	BENAR
		SALAH
		BENAR
		SALAH
		BENAR
		BENAR
		BENAR
		BENAR
		SALAH
		BENAR
Rendah	3	BENAR
		BENAR
		BENAR
		BENAR
		SALAH
		BENAR
		BENAR
		SALAH
		BENAR
		BENAR
	4	SALAH
		BENAR
		SALAH
		BENAR
		SALAH
		BENAR
		BENAR
		BENAR

		SALAH
		BENAR

Gambar 6.3 Contoh citra mata dan wajah

6.2 Analisis Hasil Pengujian

Pada sub-bab ini akan dibahas lebih lanjut mengenai hasil pengujian yang telah dilakukan pada sub-bab sebelumnya. Analisa dari hasil pengujian ini juga ikut serta menjawab pertanyaan yang disampaikan pada rumusan masalah sebelumnya.

6.2.1 Analisis Pengujian Kebutuhan Fungsional

Pada pengujian ini didapatkan algoritme *facial landmark* dapat memproses citra pengguna untuk memberikan *facial landmark* dalam waktu rata rata 0.13 detik. Algoritme facial landmark juga memberikan standar deviasi sebesar 0.007 yang menandakan performa yang stabil dalam memberikan hasil *facial landmark*. Nilai ini dinilai sudah sangat memuaskan untuk memberikan hasil terbaik pada saat estimasi arah pandangan pengguna

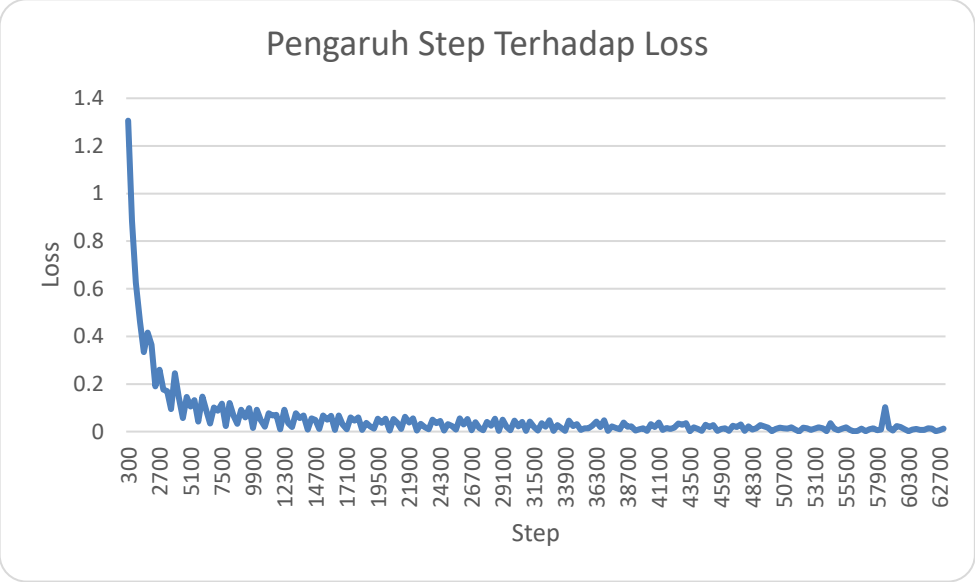
Kemudian pengujian selanjutnya didapatkan bahwa algoritme *blink detection* dapat melakukan deteksi pada kedipan mata pengguna dengan akurasi sebesar 90%. Hasil ini didapatkan setelah melakukan pengujian 10 kali, dengan kondisi pada pencahayaan yang baik di dalam ruangan. Nilai akurasi ini dinilai sudah sangat baik dalam mendeteksi kedipan pengguna dikarenakan pada penggunaannya dibutuhkan kedipan dengan durasi 1 detik dari pengguna untuk mengaktifkan fitur yang diinginkan, durasi kedipan yang panjang akan mempermudah sistem untuk mendeteksi kedipan pengguna.

Terakhir pada pengujian GUI yang diuji berdasarkan arah pandangan pengguna. Didapatkan bahwa GUI dapat memberikan hasil arah pandangan pengguna dengan akurasi dengan kisaran nilai 75% hingga 100%. Ketika hasil estimasi arah pandangan tidak sesuai dengan *ground thruth* GUI tetap

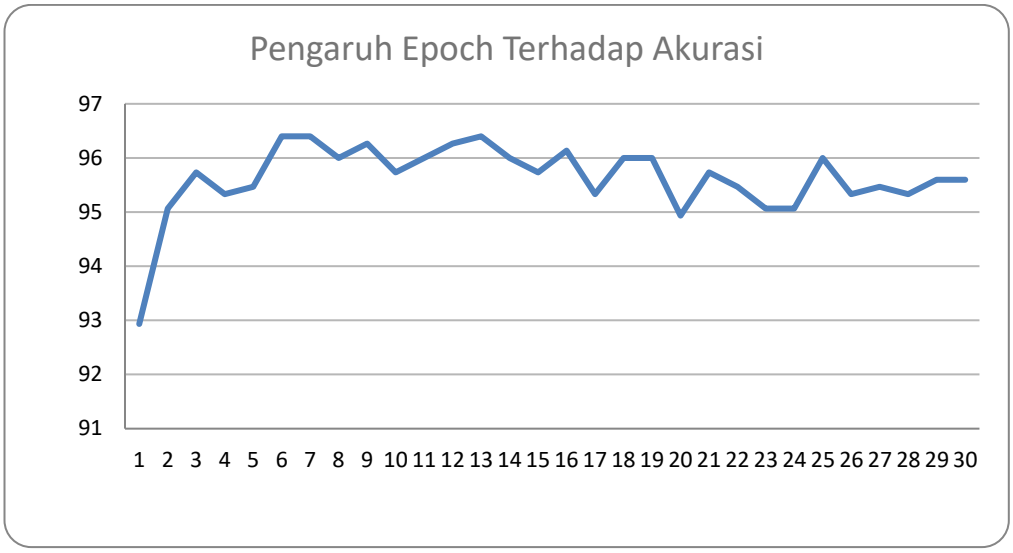
menunjukkan hasil yang diberikan oleh arah pandangan yang digunakkan sebagai inputnya. Kesalahan yang kerap terjadi adalah ketika pengguna melihat ke salah satu sisi (kanan atau kiri), namun GUI memberikan hasil estimasi (dalam bentuk tombol berwarna kuning) ke arah yang sama namun pada bagian (atas atau bawah) yang salah. Hal ini dapat disebabkan karena citra yang diterima terlalu mirip antara kiri atas dengan kiri bawah ataupun kanan atas dengan kanan bawah.

6.2.2 Analisis Pengujian Pengaruh Epoch Terhadap Performa Model

Dari hasil pengujian yang sudah dilakukan dapat dibuat grafik hubungan antara jumlah step dan performa model yang dihitung berdasarkan akurasi serta *loss* dari model. Grafik yang menggambarkan relasi step dengan loss ditunjukkan pada Gambar 6.4 dan grafik relasi antara epoch dan akurasi ditunjukkan pada Gambar 6.5. Akurasi hanya dihitung berdasarkan epoch dikarenakan dibutuhkan adanya proses *back-propagation* sebelum model dapat dievaluasi, sementara loss dapat dievaluasi pada saat *training* dengan satuan yang lebih detail yaitu step. Setiap epoch yang dilakukan saat *training* terdiri dari 2100 step.



Gambar 6.4 Grafik Relasi Antara Step Dengan Loss



Gambar 6.5 Grafik Relasi Antara Epoch Dengan Akurasi

Dari data yang didapatkan pada proses training dapat dilakukan observasi bahwa *loss* dari model sudah mulai melandai pada step 12.900. Pelandaian dari *loss* ini menandakan bahwa model sudah tidak mempelajari informasi baru mengenai objek yang dilakukan klasifikasi. Oleh karena itu *training* dengan *step* lebih dari 12.900 atau 6 epoch tidaklah efektif dinilai dari hasil loss.

Kesimpulan yang sama juga dapat ditarik apabila mengamati pengaruh epoch terhadap akurasi dari model. Setelah epoch ke 6 model tidak mengalami kenaikan yang signifikan dalam performa akurasi menggunakkan data testing. Sebaliknya, epoch diatas 6 menghasilkan model dengan akurasi yang cenderung menurun ke angka 95% dibandingkan dengan model dengan epoch 6 yang memiliki akurasi sebesar 96.4%.

Dari hasil pengujian proses *training* dapat disimpulkan bahwa jumlah epoch yang menghasilkan perforrma yang optimal berdasarkan *loss* dan akurasi adalah 6 epoch. Dengan mempertimbangkan hasil pengujian ini maka seluruh pengujian yang dilakukan selanjutnya akan menggunakkan model yang dilakukan *training* dengan 6 epoch dan 2.100 step.

6.2.3 Analisis Pengujian Akurasi, Presisi, Recall, dan Specitivity Model

Untuk melakukan analisis pengujian performa model secara mendetail akan digunakkan parameter akurasi, presisi, recall, dan specitivity berdasarkan *confusion matrix* yang didapatkan pada pengujian sebelumnya. Untuk mempermudah perhitungan maka klasifikasi yang sebelumnya berupa *multiclass classification* akan diubah bentuknya menjadi *binary classification* dengan memfokuskan kepada salah satu kelas.

Untuk menghitung keempat parameter performa ini maka dibutuhkan *confusion matrix* baru untuk setiap kelas yang ada pada model. Oleh karena itu dibuat 5 *confusion matrix* baru berdarkan hasil yang didapatkan pada Gambar 6.2

dengan kelas yang akan di evaluasi sebagai kelas positif dan kelas yang lain sebagai kelas negatifnya. Kelima *confusion matrix* tersebut dapat dilihat pada Gambar 6.6.

	0	other
0	142	0
other	8	604

	2	other
2	154	3
other	2	595

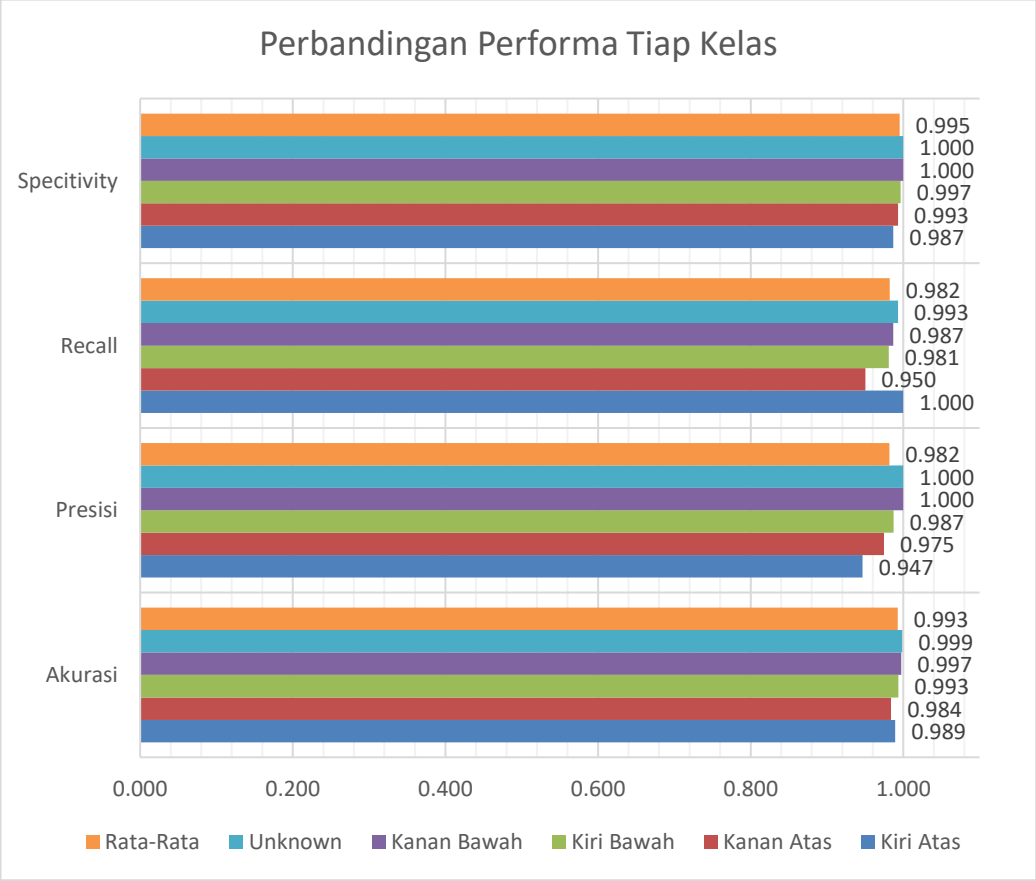
	4	other
4	142	1
other	0	611

	1	other
1	153	8
other	4	589

	3	other
3	149	2
other	0	603

Gambar 6.6 *Confusion Matrix* Yang Sudah Dipisahkan

Dari *confusion matrix* biner inilah dapat dihitung akurasi, presisi, recall, dan specitivity dari masing masing kelas. Nilai ini kemudian akan dirata-ratakan untuk menentukan performa keseluruhan dari model yang sudah dibuat. Hasil perhitungan dengan presisi tiga tempat desimal disajikan pada grafik di Gambar 6.7.



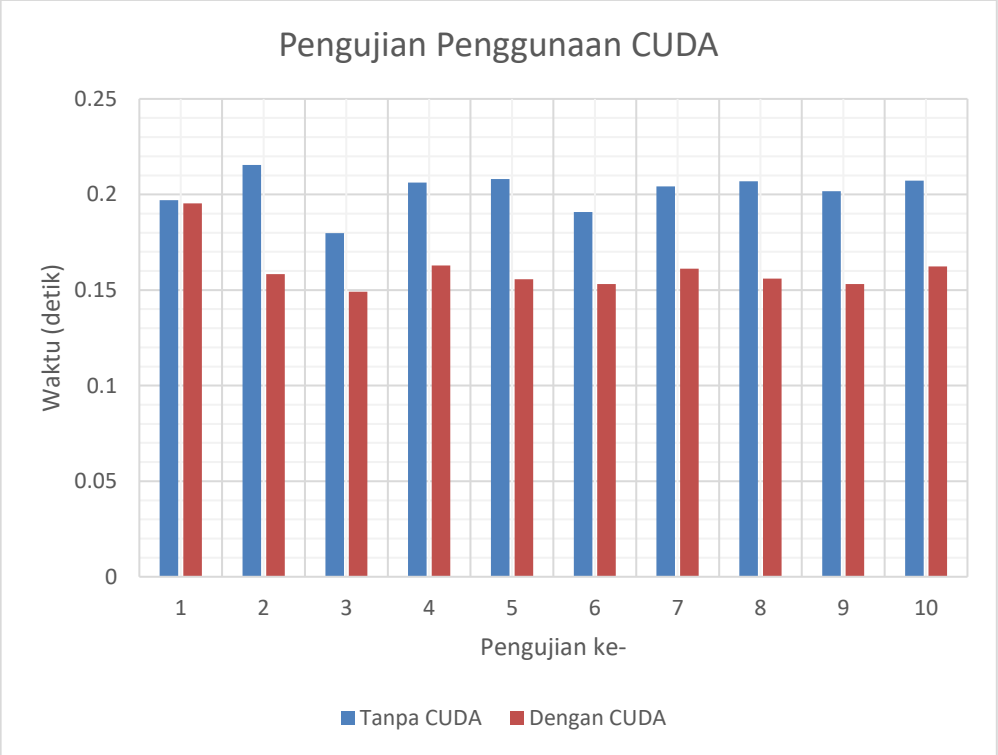
Gambar 6.7 Perbandingan Performa Tiap Kelas

Dari data ini dapat disimpulkan bahwa masing-masing kelas memiliki performa yang sangat baik, bahkan cenderung mengarah ke *overfitting*. Hal ini dapat disebabkan oleh dataset yang digunakkan kurang variatif pada proses *training*. Namun performa model secara keseluruhan dapat dinilai sangat memuaskan berdasarkan kepada parameter yang diuji.

6.2.4 Analisis Pengujian Akurasi Terhadap Tinggi Pengguna

Pada pengujian ini digunakkan rata-rata akurasi pada pengguna yang memiliki tinggi badan yang variatif. Dari pengujian ini didapatkan bahwa tinggi pengguna tidak memiliki pengaruh yang signifikan pada akurasi yang dihasilkan. Baik pengguna dengan tinggi badan rendah maupun tinggi mendapatkan akurasi yang berkisar dari 70% hingga 80%.

6.2.5 Analisis Pengujian Pengaruh CUDA Terhadap Performa



Gambar 6.8 Perbandingan Pengujian Waktu Komputasi

Algoritme *Fusion CNN* menggunakkan *framework* PyTorch dapat menggunakkan akselerasi CUDA untuk mempercepat waktu *inference*. Hal ini juga dapat dilihat dari tabel yang disajikan pada Tabel 6.5. Dapat diobservasi bahwa terjadi penurunan 20% dari waktu komputasi yang dibutuhkan untuk mendapatkan satu prediksi dan standar deviasi yang tetap sama kecilnya yaitu di kisaran 0.01 menandakan bahwa sistem dapat menjalankan algoritme ini dengan stabil pada waktu rata-rata 0.16 detik. Grafik perbandingan antara waktu komputasi dengan menggunakkan CUDA dan tanpa menggunakkan CUDA juga disajikan pada Gambar 6.8.

6.2.6 Analisis Pengujian Penggunaan Daya Komputasi

Pada pengujian ini digunakkan rata rata dari persentase penggunaan CPU dan penggunaan memori saat program utama berjalan dan ketika sistem dalam keadaan *idle*. Dari pengujian ini didapatkan bahwa ada kenaikan penggunaan CPU 5 kali lipat dan kenaikan penggunaan memori 2 kali lipat ketika sistem

menjalankan program utama. Namun ada kejanggalan yang dapat diteliti ketika pengujian yaitu nilai penggunaan *CPU core* 2 dan 3 yang diam di angka 0%, hal ini dapat menandakan adanya permasalahan optimasi sistem sehingga program utama hanya dapat berjalan pada 4 *CPU core* dari 6 yang tersedia.

6.2.7 Analisis Pengujian Penggunaan Daya Listrik

Pada pengujian ini digunakkan rata-rata penggunaan daya listrik pada saat sistem menjalankan program utama dengan semua komponen sudah terpasang ke sistem. dari pengujian ini didapatkan adanya rata-rata kenaikan sebesar 25%. Nilai rata-rata penggunaan daya listrik pada sistem adalah 12 Watt dibandingkan dengan ketika sistem *idle* yaitu 8.98 Watt. Apabila diasumsikan sistem terus menjalankan program utama maka dengan baterai 9600Wh yang terpasang pada sistem, sistem dapat menjalankan program utama selama 80 jam. Hasil yang didapatkan ini berarti pengguna tidak perlu khawatir mengenai penggunaan daya listrik ketika menggunakkan sistem.

BAB 7 PENUTUP

Pada bab ini akan dijelaskan kesimpulan dari pengujian yang sudah dilakukan pada bab 6. Kesimpulan yang diberikan akan menjawab pertanyaan-pertanyaan yang disampaikan pada rumusan masalah penelitian ini. Akan dibahas juga mengenai saran untuk penelitian kedepan beserta kemungkinan solusi untuk permasalahan baru yang ditemui pada pengujian penelitian ini.

7.1 Kesimpulan

Dari penelitian yang telah dilakukan sebelumnya dapat ditarik beberapa kesimpulan berdasarkan rumusan masalah yang sebelumnya telah ditentukan. Kesimpulan yang didapatkan adalah,

1. Berdasarkan rumusan masalah pertama dimana kebutuhan fungsional dari sistem diuji, didapatkan bahwa kebutuhan fungsional sistem pertama yaitu GUI mendapatkan akurasi yang tinggi yaitu di kisaran 75% hingga 100% yang dirasa sudah sangat mencukupi. Kemudian algoritme *blink detection* juga mendapatkan akurasi yang sangat baik hingga 90% untuk mendeteksi kedipan mata pengguna. Dan yang terakhir adalah algoritme *facial landmark* dapat berjalan dengan waktu komputasi 0.13 detik dan standar deviasi 0.007 detik.
2. Berdasarkan rumusan masalah yang kedua dapat disimpulkan bahwa dengan kenaikan *epoch* terjadi juga kenaikan pada akurasi dan penurunan hasil dari *loss function*. Namun perubahan ini mulai melandai setelah melewati epoch ke 6 dimana nilai *loss* dari model tidak mengalami perubahan signifikan dan cenderung berosilasi pada nilai yang sama. Begitu juga dengan akurasi model dimana didapatkan akurasi optimal pada epoch ke-6 dan setelah itu tidak terjadi perubahan yang signifikan dan cenderung menurun. Sehingga dapat disimpulkan bahwa *epoch* untuk mendapatkan performa model yang optimal adalah 6.
3. Berdasarkan rumusan masalah ketiga ditemukan bahwa akurasi model dengan parameter *training* optimal memiliki nilai akurasi rata-rata 0.993 dengan akurasi per kelas ter-rendah 0.984. Presisi rata-rata dari model didapatkan sebesar 0.982 dengan nilai presisi per-kelas terendah adalah 0.947. Nilai *recall* rata-rata dari model sebesar 0.982 dengan nilai per-kelas terendah 0.95. Dan nilai *specitivity* rata-rata model sebesar 0.995 dengan nilai per-kelas terendah 0.987. Dari hasil ini model yang dibuat dapat dinilai mengalami *overfitting* yang kemungkinan besar disebabkan oleh dataset yang kurang variatif, namun *overfitting* yang terjadi tidak terlalu parah sehingga model masih dapat digunakkan untuk mendeteksi pada saat *deployment.*
4. Berdasarkan rumusan masalah keempat didapatkan bahwa dengan menggunakkan akselerasi dengan CUDA waktu *inference* atau waktu komputasi dari algoritme berkurang hingga 20% dibandingkan dengan waktu komputasi tanpa menggunakkan akselerasi CUDA. Waktu komputasi yang dihasilkan pada proses pengujian juga memiliki standar deviasi sebesar 0.01

yang menandakan bahwa sistem dapat menjalankan algoritme dengan stabil pada waktu 0.16 detik. Hasil ini dinilai sudah mencukupi untuk sistem dapat memprediksi arah pandangan pengguna dengan batasan waktu 200ms.

5. Berdasarkan rumusan masalah kelima setelah diuji ada kenaikan penggunaan daya komputasi sistem sebesar 80% dibandingkan ketika sistem sedang dalam keadaan *idle* atau tidak melakukan apa apa, namun rata-rata penggunaan CPU sistem belum mencapai 30% sehingga sistem masih dapat menjalankan beberapa *background task* dengan baik. Diamati juga adanya kenaikan penggunaan memori pada sistem hingga 56% hingga 3.2GiB dari 4GiB yang dimiliki sistem. Dari hasil ini dinyatakan bahwa sistem dapat menjalankan algoritme dengan baik dan masih memiliki *overhead* untuk menjalankan program lain untuk digunakkan pengguna.
6. Berdasarkan rumusan masalah keenam setelah sistem diuji terdapat kenaikan penggunaan daya listrik rata-rata menjadi 12 Watt atau kenaikan sebesar 25% dibandingkan saat sistem pada keadaan *idle*. Dengan nilai ini diestimasikan baterai yang digunakkan dapat memberikan daya untuk rentang waktu hingga 80 jam penggunaan. Hal ini dinilai sangat memuaskan dikarenakan sistem utama diharapkan dapat digunakkan secara *portable*.
7. Berdasarkan rumusan masalah ketujuh ditemukan bahwa GUI sistem dapat memberikan akurasi 70-80% ketika melakukan estimasi arah pandangan pengguna. Dengan hasil pengujian ini juga ditemukan bahwa pengguna dengan tinggi badan yang bermacam macam dapat menggunakkan sistem dengan baik dan masih dapat memberikan hasil yang cukup baik.

7.2 Saran

Dari penelitian ini didapatkan beberapa permasalahan yang didapati ketika pengujian yang diharapkan dapat diselesaikan pada penelitian-penelitian selanjutnya. Beberapa permasalahan yang ditemui antara lain adalah model yang mengalami *overfitting*. Hal ini dapat disebabkan oleh kurangnya deversifikasi data pada data *training*. Pada penelitian selanjutnya diharapkan data yang dikumpulkan dapat lebih variatif dengan subjek yang berbeda dan lingkungan pencahayaan yang berbeda.

Saran lain yang dapat dilakukan pada penelitian selanjutnya adalah optimasi kode agar berjalan lebih baik lagi di sistem NVIDIA Jetson. Pada penelitian ini didapatkan bahwa hanya 4 dari 6 *core* dapat digunakkan saat *inference,* namun performa lebih baik seharusnya dapat dicapai apablida dapat menggunakkan semua *core* yang tersedia.

DAFTAR REFERENSI

Aaron, J., Wait, E., DeSantis, M., & Chew, T. L. (2019). Practical Considerations in Particle and Object Tracking and Analysis. In *Current Protocols in Cell Biology* (Vol. 83, Issue 1). John Wiley and Sons Inc. https://doi.org/10.1002/cpcb.88

Aihara, S., Shibata, R., Mizukami, R., Sakai, T., & Shionoya, A. (2022). Deep Learning-Based Myoelectric Potential Estimation Method for Wheelchair Operation. *Sensors*, *22*(4). https://doi.org/10.3390/s22041615

Akinyelu, A. A., & Blignaut, P. (2022). Convolutional Neural Network-Based Technique for Gaze Estimation on Mobile Devices. *Frontiers in Artificial Intelligence*, *4*. https://doi.org/10.3389/frai.2021.796825

Anam, K., & Saleh, A. (2020). Voice Controlled Wheelchair for Disabled Patients based on CNN and LSTM; Voice Controlled Wheelchair for Disabled Patients based on CNN and LSTM. In *2020 4th International Conference on Informatics and Computational Sciences (ICICoS)*. https://doi.org/10.1109/ICICoS51170.2020.9299007

Araujo, J. M., Zhang, G., Hansen, J. P. P., & Puthusserypady, S. (2020, June 2). Exploring Eye-Gaze Wheelchair Control. *Eye Tracking Research and Applications Symposium (ETRA)*. https://doi.org/10.1145/3379157.3388933

Armstrong, W., Borg, J., Krizack, M., Lindsley, A., Mines, K., Pearlman, J., Reisinger, K., & Sheldon, S. (2008). *Guidelines on the provision of in less resourced settings*. http://apps.who.int/iris/bitstream/handle/10665/43960/9789241547482_eng.pdf;jsessionid=B6B7AA1DD349BAA500C70A79E16EABF1?sequence=1

Cheng, Y., Wang, H., Bao, Y., & Lu, F. (2021). *Appearance-based Gaze Estimation With Deep Learning: A Review and Benchmark*. http://arxiv.org/abs/2104.12668

Fard, A. P., & Mahoor, M. H. (2021). *Facial Landmark Points Detection Using Knowledge Distillation-Based Neural Networks*. http://arxiv.org/abs/2111.07047

Faris, I., & Utaminingrum, F. (2022). Rancang Bangun Sistem Deteksi Gerakan Mata untuk Pemilihan Enam Menu Display menggunakan Circular Hough Transform berdasarkan Facial Landmark berbasis NVIDIA Jetson Nano. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *6*(6), 2799–2804. http://j-ptiik.ub.ac.id

Fergus, K. B., Zambeli-Ljepović, A., Hampson, L. A., Copp, H. L., & Nagata, J. M. (2022). Health care utilization in young adults with childhood physical disabilities: a nationally representative prospective cohort study. *BMC Pediatrics*, *22*(1). https://doi.org/10.1186/s12887-022-03563-0

Fischer, T., Chang, J., & Demiris, Y. (2018). RT-GENE: Real-Time Eye Gaze Estimation in Natural Environments. *European Conference on Computer Vision (ECCV)*, 334–352. www.imperial.ac.uk/PersonalRobotics.

Health Organization, W. (2011). *WORLD REPORT ON DISABILITY*. http://www.who.int/about/

Hecht, H., Siebrand, S., & Thönes, S. (2020). Quantifying the Wollaston Illusion. *Perception*, *49*(5), 588–599. https://doi.org/10.1177/0301006620915421

Hu, J., Chen, Z., Yang, M., Zhang, R., & Cui, Y. (2018). A multiscale fusion convolutional neural network for plant leaf recognition. *IEEE Signal Processing Letters*, *25*(6), 853–857. https://doi.org/10.1109/LSP.2018.2809688

Huang, S. C., Pareek, A., Seyyedi, S., Banerjee, I., & Lungren, M. P. (2020). Fusion of medical imaging and electronic health records using deep learning: a systematic review and implementation guidelines. In *npj Digital Medicine* (Vol. 3, Issue 1). Nature Research. https://doi.org/10.1038/s41746-020-00341-z

Juhong, A., Treebupachatsakul, T., & Pintavirooj, C. (2018). Smart eye-tracking system. *2018 International Workshop on Advanced Image Technology, IWAIT 2018*, 1–4. https://doi.org/10.1109/IWAIT.2018.8369701

Kuwahara, A., Nishikawa, K., Hirakawa, R., Kawano, H., & Nakatoh, Y. (2022). Eye fatigue estimation using blink detection based on Eye Aspect Ratio Mapping(EARM). *Cognitive Robotics*, *2*, 50–59. https://doi.org/10.1016/j.cogr.2022.01.003

Lavinia, Y., Vo, H. H., & Verma, A. (2017). Fusion based deep CNN for improved large-scale image action recognition. *Proceedings - 2016 IEEE International Symposium on Multimedia, ISM 2016*, 609–614. https://doi.org/10.1109/ISM.2016.84

Lecun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, *1*(4), 541–551.

Li, B., Fu, H., Wen, D., & Lo, W. (2018). Etracker: A mobile gaze-tracking system with near-eye display based on a combined gaze-tracking algorithm. *Sensors (Switzerland)*, *18*(5). https://doi.org/10.3390/s18051626

Li, H., Sun, J., Xu, Z., & Chen, L. (2017). Multimodal 2D+3D Facial Expression Recognition with Deep Fusion Convolutional Neural Network. *IEEE Transactions on Multimedia*, *19*(12), 2816–2831. https://doi.org/10.1109/TMM.2017.2713408

Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 1–21. https://doi.org/10.1109/tnnls.2021.3084827

McClain-Nhlapo, C. (2022, April 14). *Disability Inclusion Overview*. https://www.worldbank.org/en/topic/disability

Mohan, S., & Phirke, M. (2020). Eye Gaze Estimation Invisible and IR Spectrum for Driver Monitoring System. *Signal & Image Processing : An International Journal*, *11*(5), 1–20. https://doi.org/10.5121/sipij.2020.11501

Mowrer, H., Ruch, T. C., & Miller, N. E. (1935). *THE CORNEO-RETINAL POTENTIAL DIFFERENCE AS THE BASIS OF THE GALVANOMETRIC METHOD OF RECORDING EYE MOVEMENTS I*. www.physiology.org/journal/ajplegacy

Naqvi, R. A., Arsalan, M., Batchuluun, G., Yoon, H. S., & Park, K. R. (2018). Deep learning-based gaze detection system for automobile drivers using a NIR camera sensor. *Sensors (Switzerland)*, *18*(2). https://doi.org/10.3390/s18020456

Pangestu, G., & Utaminingrum, F. (2020). Electric Wheelchair Control Mechanism Using Eye-mark Key Point Detection. *International Journal of Intelligent Engineering and Systems*, *13*(2), 228–238. https://doi.org/10.22266/ijies2020.0430.22

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury Google, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Xamla, A. K., Yang, E., Devito, Z., Raison Nabla, M., Tejani, A., Chilamkurthy, S., Ai, Q., Steiner, B., … Chintala, S. (2019). *PyTorch: An Imperative Style, High-Performance Deep Learning Library*.

Rockland, R. H., & Reisman, S. (1998). Voice Activated Wheelchair Controller. *Bioengineering, Proceedings of the Northeast Conference*, 128–129. https://doi.org/10.1109/nebc.1998.664900

Soukupová, T., & Cec, J. (2016). Real-Time Eye Blink Detection using Facial Landmarks. *21st Computer Vision Winter Workshop*.

Süzen, A. A., Duman, B., & Şen, B. (2020). Benchmark Analysis of Jetson TX2, Jetson Nano and Raspberry PI using Deep-CNN. *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*.

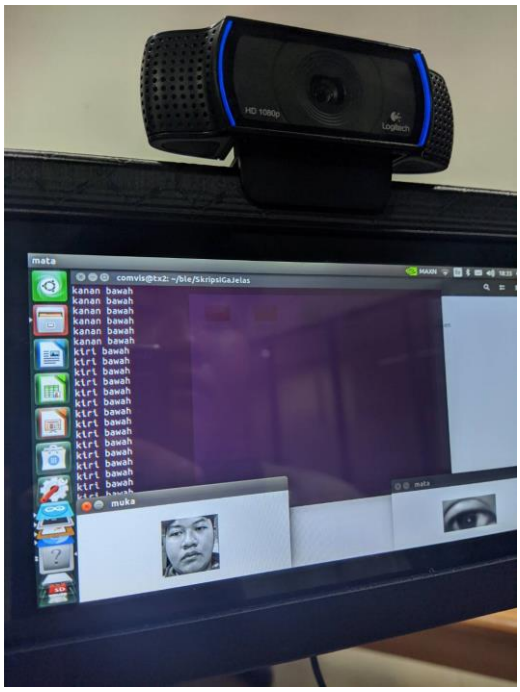
Utaminingrum, F., Purwanto, A. D., Masruri, M. R. R., Ogata, K., & Somawirata, I. K. (2021). Eye movement and blink detection for selecting menu on-screen display using probability analysis based on facial landmark. *International Journal of Innovative Computing, Information and Control*, *17*(4), 1287–1303. https://doi.org/10.24507/ijicic.17.04.1287

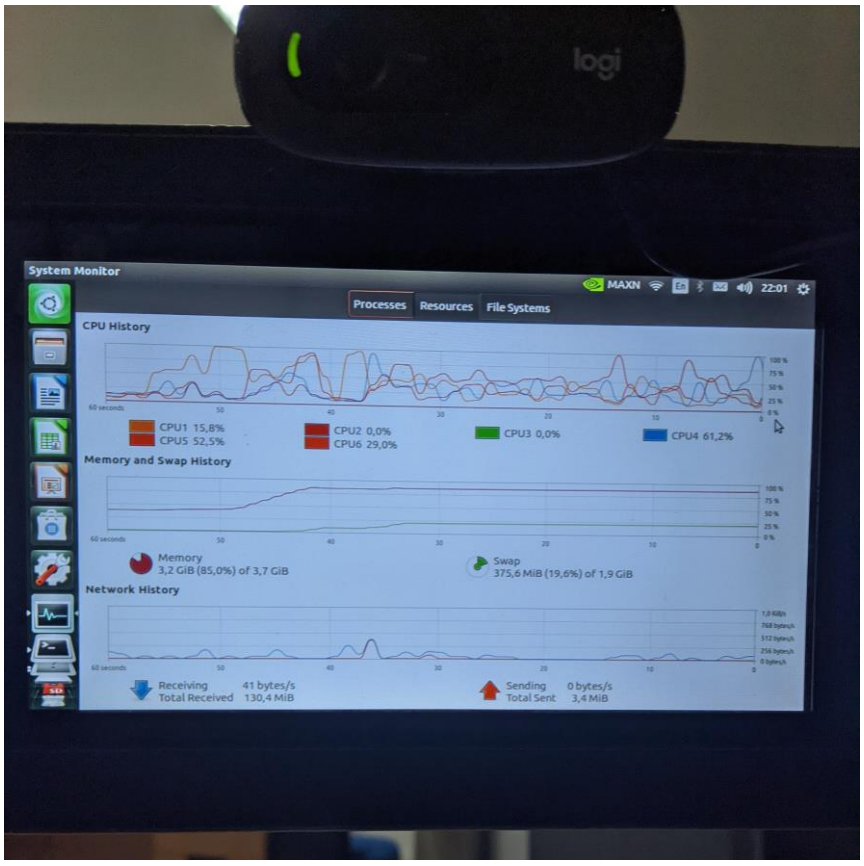
Wu, Y., & Ji, Q. (2019). Facial Landmark Detection: A Literature Survey. *International Journal of Computer Vision*, *127*(2), 115–142. https://doi.org/10.1007/s11263-018-1097-z

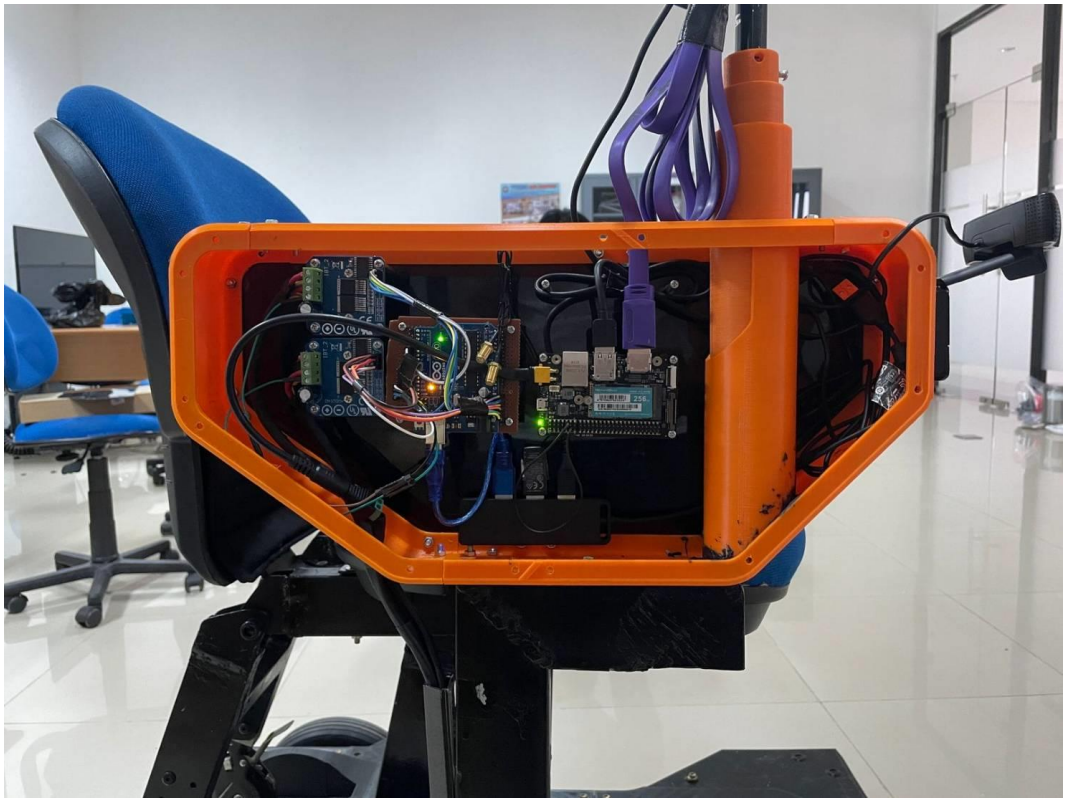
Yoon, H. S., Baek, N. R., Truong, N. Q., & Park, K. R. (2019). Driver Gaze Detection Based on Deep Residual Networks Using the Combined Single Image of Dual

Near-Infrared Cameras. *IEEE Access*, *7*, 93448–93461. https://doi.org/10.1109/ACCESS.2019.2928339

LAMPIRAN A DOKUMENTASI ALAT







LAMPIRAN B KODE PROGRAM

Kode program dapat di unduh pada repository github dibawah:

https://github.com/Blessius321/SkripsiS1.git