ALGORITMA PENDETEKSI OBJEK SEKITAR BERBASIS *DEEP LEARNING* UNTUK PENYANDANG DISABILITAS TUNANETRA

SKRIPSI

Diajukan sebagai Salah Satu Syarat untuk Mendapatkan
Gelar Sarjana Komputer (S.kom) Program Studi Informatika

AKRAM
10584111920

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH MAKASSAR
2024

ALGORITMA PENDETEKSI OBJEK SEKITAR BERBASIS *DEEP LEARNING* UNTUK PENYANDANG DISABILITAS TUNANETRA

Diajukan sebagai Salah Satu Syarat untuk Mendapatkan
Gelar Sarjana Komputer (S.kom) Program Studi Informatika

Disusun Dan Diajukan Oleh :

AKRAM

10584111920

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH MAKASSAR
2024

UNIVERSITAS MUHAMMADIYAH MAKASSAR
FAKULTAS TEKNIK
GEDUNG MENARA IQRA LT. 3
Jl. Sultan Alauddin No. 259 Telp. (0411) 866 972 Fax (0411) 865 588 Makassar 90221
Website: www.unismuh.ac.id, e_mail: unismuh@gmail.com
Website: http://teknik.unismuh.makassar.ac.id

بسم الله الرحمن الرحيم

PENGESAHAN

Skripsi atas nama Akram dengan nomor induk Mahasiswa 105 84 11119 20, dinyatakan diterima dan disahkan oleh Panitia Ujian Tugas Akhir/Skripsi sesuai dengan Surat Keputusan Dekan Fakultas Teknik Universitas Muhammadiyah Makassar Nomor : 0010/SK-Y/55202/091004/2024, sebagai salah satu syarat guna memperoleh gelar Sarjana Komputer pada Program Studi Informatika Fakultas Teknik Universitas Muhammadiyah Makassar pada hari Jumat tanggal 30 Agustus 2024.

Makassar, 25 Safar 1445 H / 30 Agustus 2024 M

Panitia Ujian :

1. Pengawas Umum
   a. Rektor Universitas Muhammadiyah Makassar
      Dr. Ir. H. Abd. Rakhim Nanda, ST., MT., IPU. : .....
   b. Dekan Fakultas Teknik Universitas Hasanuddin
      Prof. Dr. Eng. Muhammad Isran Ramli, ST., MT. : .....
2. Penguji
   a. Ketua : Dr. Ir. Zahir Zainuddin, M.Sc. : .....
   b. Sekertaris : Titin Wahyuni, S.Pd.,M.T. : .....
3. Anggota : 1. Muhyiddin A. M. Hayat, S.Kom., MT. : .....
   2. Lukman, S.Kom., MT. : .....
   3. Desi Anggreani, S.Kom., MT. : .....

Mengetahui :

Pembimbing I

Fahrim Irhamna Rahman S.Kom., MT.

Pembimbing II

Rizki Yusliana Bakti ST., MT.

Dekan

Dr. Ir. Hj. Nurnawaty, ST., MT., IPM.
NBM. 795 108

UNIVERSITAS MUHAMMADIYAH MAKASSAR
FAKULTAS TEKNIK
GEDUNG MENARA IQRA LT. 3
Jl. Sultan Alauddin No. 259 Telp. (0411) 866 972 Fax (0411) 865 588 Makassar 90221
Website: www.unismuh.ac.id, e_mail: unismuh@gmail.com
Website: http://teknik.unismuh.makassar.ac.id

بسم الله الرحمن الرحيم

HALAMAN PENGESAHAN

Tugas Akhir ini diajukan untuk memenuhi syarat ujian guna memperoleh gelar Sarjana Komputer (S.Kom) Program Studi Informatika Fakultas Teknik Universitas Muhammadiyah Makassar.

Judul Skripsi : **ALGORITMA PENDETEKSI OBJEK SEKITAR BERBASIS DEEP LEARNING UNTUK PENYANDANG DISABILITAS TUNANETRA**

Nama : Akram

Stambuk : 105 84 11119 20

Makassar, 30 Agustus 2024

Telah Diperiksa dan Disetujui
Oleh Dosen Pembimbing:

Pembimbing I	Pembimbing II
Fahrim Irhamna Rahman S.Kom., MT.	**Rizki Yusliana Bakti ST., MT.**

Mengetahui,
Ketua Program Studi Informatika

Muhyiddin A.M. Hayat, S.Kom., MT.
NBM. 1504577

MOTTO DAN PERSEMBAHAN

Motto

"Hidup bukan saling mendahului, bermimpilah sendiri-sendiri"

Persembahan

Bismillahirrahmanirrahim. Skripsi ini saya persembahkan dengan penuh rasa syukur kepada: Allah SWT, yang dengan rahmat dan pertolongan-Nya, mempermudah segala urusan sehingga saya dapat menyelesaikan skripsi ini dengan baik. Kepada kedua orang tua tercinta, Bapak (Alm.) Laris dan Ibu Syamsuryati, yang senantiasa memanjatkan doa-doa penuh kasih dan menjadi sumber motivasi saya dalam menyelesaikan tugas ini. Terima kasih telah mengantarkan saya hingga titik ini, dan dengan restu serta dukungan kalian, saya berhasil menjadi sarjana pertama dalam keluarga kita. Untuk diri saya sendiri, Akram, saya ucapkan selamat atas segala usaha dan perjuangan yang telah ditempuh hingga sejauh ini. Meskipun menghadapi berbagai tantangan, saya bangga tidak pernah memilih untuk menyerah, seberat apapun proses penyusunan skripsi ini. Saya juga menyampaikan rasa terima kasih yang mendalam kepada semua pihak yang telah mendukung keberhasilan ini. Terutama kepada Kepala Program Studi Informatika, para dosen, serta seluruh staf dan civitas akademika Fakultas Teknik Universitas Muhammadiyah Makassar. Dukungan kalian sangat berarti dalam pencapaian ini. Kepada rekan-rekan seperjuangan di Prodi Informatika Angkatan 2020, terima kasih atas kebersamaan dan dukungannya. Bantuan dan semangat kalian begitu berarti, sehingga kita dapat menyelesaikan studi ini bersama-sama. Tak lupa, kepada keluarga, kerabat, sahabat, dan semua pihak yang telah membantu, meskipun tidak dapat saya sebutkan satu per satu, saya haturkan terima kasih yang sebesar-besarnya. Dukungan, semangat, dan doa kalian merupakan bagian penting dalam perjalanan ini. Akhir kata, semoga skripsi ini dapat bermanfaat dan memberikan wawasan yang berguna bagi orang lain. Aamiin..

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabaraktuh

Dengan penuh rasa syukur ke hadirat Allah Subhanallahu Wa Ta'ala, atas nikmat iman, Islam, dan kesehatan yang senantiasa terlimpahkan. sehingga penulis dapat menyelesaikan Laporan Tugas Akhir yang berjudul "**ALGORITMA PENDETEKSI OBJEK SEKITAR BERBASIS *DEEP LEARNING* UNTUK PENYANDANG DISABILITAS TUNANETRA**". Shalawat serta salam teriring kepada junjungan Nabi Muhammad SAW, pembawa rahmat bagi semesta alam. yang telah membawa kita dari Zaman jahiliah menuju Zaman yang serba modern seperti saat ini.

Ucapan terima kasih yang tidak terhingga penulis sampaikan kepada semua pihak yang telah membantu dan memberikan dukungan dalam penyusunan Laporan Tugas Akhir ini, khususnya:

1. Kepada kedua orang tua tercinta saya, Bapak **Alm Drs. Laris** dan Ibu **Syamsuryati**, persembahan terima kasih yang tak terhingga atas segala pengorbanan, kasih sayang, dan bimbingan, yang telah diberikan.
2. Ibu **Dr.Ir.Hj Nurnawati, S.T., M.T., I.P.M**, selaku Dekan Fakultas Teknik.
3. Bapak **Muh. Syafaat S Kuba, S.T., M.T**, selaku Wakil Dekan Fakultas Teknik.
4. Bapak **Muhyiddin AM Hayat S.Kom., M.T**, selaku Ketua Prodi Informatika.
5. Bapak **Fahrim Irhamna Rachman S.Kom., M.T**, selaku Dosen Pembimbing 1 proposal.
6. Ibu **Rizki Yusliana Bakti S.T.,M.T** selaku Dosen Pembimbing 2 Proposal.
7. Dosen dan Staf Fakultas Teknik Universitas Muhammadiyah Makassar.
8. Senior-senior saya sejurusan terkhusus Kak **Andi Agung Dwi Arya, S.Kom**, atas bantuan dan bimbingannya selama saya berkuliah.
9. Teman-teman Khususnya Angkatan 2020 Fakultas Teknik, Universitas Muhammadiyah Makassar, terima kasih atas dukungan dan doanya.

10. Teman-teman kelas D angkatan 2020 Program Studi Informatika Universitas Muhammadiyah Makassar

Penulis menyadari bahwa Laporan Tugas Akhir ini masih jauh dari kesempurnaan. Oleh karena itu, kritik dan saran yang membangun sangat penulis harapkan demi penyempurnaan laporan ini di masa depan. Harapan penulis, semoga Laporan Tugas Akhir ini dapat memberikan manfaat bagi penyandang disabilitas tunanetra dalam meningkatkan kemandirian dan kualitas hidup mereka. Akhir kata, penulis mohon maaf atas segala kekurangan dan kekhilafan yang terdapat dalam Laporan Tugas Akhir ini.

Billahi fisabililhaq, fastabiqul khairat.
Wassalamualaikum Wr.Wb.

Makassar, Agustus 2024

Akram

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

DAFTAR ISTILAH

Machine Learning	*Machine Learning* adalah bagian dari kecerdasan buatan yang memungkinkan komputer belajar dari data tanpa pemrograman eksplisit. Algoritma menemukan pola dalam data untuk membuat prediksi atau keputusan.
Computer Vision	*Computer Vision* adalah bidang kecerdasan buatan yang memungkinkan komputer memahami dan menafsirkan gambar dan video. Ini digunakan untuk tugas seperti pengenalan objek, deteksi wajah, OCR, dan analisis gerakan, dengan aplikasi di keamanan, kendaraan otonom, dan medis.
Deep Learning	*Deep Learning* adalah cabang dari *machine learning* yang menggunakan jaringan saraf tiruan dengan banyak lapisan (*deep neural networks*) untuk memodelkan dan memproses data yang kompleks, seperti gambar dan suara.
You Only Look Once (YOLO)	Adalah algoritma deteksi objek *real-time* yang cepat dan akurat, yang mendeteksi dan mengklasifikasikan objek dalam satu langkah.
Object Detection	*Object Detection* adalah teknologi komputer yang digunakan untuk menemukan dan mengidentifikasi objek dalam gambar atau video.
Text to Speech	*Text to Speech* (TTS) adalah teknologi yang mengubah teks tertulis menjadi suara ucapan yang dihasilkan secara otomatis oleh komputer.

Roboflow	*Roboflow* adalah *platform* yang memudahkan pengembangan dan pengelolaan model *computer vision* dengan menyediakan alat untuk anotasi data, augmentasi, dan pelatihan model.
Preprocessing	Adalah langkah awal dalam *machine learning* di mana data diolah dan dibersihkan untuk memastikan kualitas dan format yang sesuai sebelum digunakan dalam model. Ini termasuk menangani nilai yang hilang, normalisasi, dan transformasi data.
Graphics Processing Unit (GPU)	Adalah prosesor khusus yang dirancang untuk mempercepat pemrosesan grafis dan tugas komputasi paralel, sering digunakan dalam rendering video, game, dan aplikasi *machine learning*.
Flowchart	*Flowchart* adalah diagram yang menunjukkan langkah-langkah atau alur dari suatu proses secara visual menggunakan simbol-simbol seperti kotak, panah, dan lainnya.
Validasi	*Validasi* adalah proses untuk memastikan bahwa model atau sistem bekerja dengan baik dan menghasilkan output yang akurat berdasarkan data yang tidak pernah dilihat sebelumnya.

BAB I
PENDAHULUAN

A. Latar Belakang

Di antara kita terdapat saudara-saudara kita yang mengalami keterbatasan penglihatan, dikenal sebagai difabel tunanetra. Kondisi ini dapat bervariasi, dari *low vision* hingga *totally blind*. Berdasarkan data Kementerian Kesehatan RI, 1,5% dari populasi Indonesia tergolong tunanetra. Dengan asumsi jumlah penduduk Indonesia saat ini mencapai 250 juta jiwa, maka diperkirakan terdapat minimal 3,75 juta tunanetra di Indonesia, termasuk mereka yang mengalami low vision (Kadiva Dwilia Rosadi Putri, 2023). Keterbatasan penglihatan ini menjadi hambatan dalam berbagai aktivitas sehari-hari, seperti menjelajahi ruangan, mengenali benda, dan mengidentifikasi orang. Hal ini dapat menimbulkan rasa terisolasi, frustrasi, dan ketergantungan pada orang lain.

Kekurangan sistem yang efektif untuk mendeteksi dan mengenali objek di sekitar bagi penyandang adalah salah satu tantangan utama yang dihadapi oleh orang dengan keterbatasan penglihatan. Untuk aktivitas sehari-hari seperti belanja, memasak, atau bahkan hanya bergerak, hal ini sangat penting. Solusi saat ini, seperti tongkat, anjing pemandu, atau bantuan orang lain, seringkali memiliki keterbatasan dalam mendeteksi objek dan memberikan informasi lingkungan yang akurat.

Kemajuan teknologi saat ini telah sangat maju sehingga beberapa teknologi dapat membantu individu dengan keterbatasan penglihatan atau tunanetra. Salah satu teknologi yang dapat digunakan adalah pendeteksian objek yang dilakukan oleh kamera ponsel. Sistem pendeteksian objek diharapkan dapat menemukan objek dalam kehidupan nyata dengan menggunakan model objek yang telah diketahui sebelumnya (Randy Moh Yusup et al., 2024).

Deteksi objek cabang dari computer vision, memungkinkan komputer untuk mengenali dan memahami objek dalam gambar atau video. Teknologi ini mereplikasi kemampuan manusia dalam melihat dan memahami dunia di sekitarnya, memungkinkan komputer untuk menemukan dan mengidentifikasi

objek-objek tertentu (Sarmah et al., 2023). Penerapan deteksi objek memiliki dampak signifikan bagi penyandang tunanetra, meningkatkan mobilitas dan kemandirian mereka. Sistem ini membantu navigasi dengan memberikan informasi tentang objek di sekitar, seperti orang, kendaraan, dan hambatan. Kemampuan ini memungkinkan mereka untuk bergerak dengan lebih aman dan percaya diri, bahkan di lingkungan yang tidak dikenal. Tujuan utama deteksi objek adalah mengenali semua objek yang umum ditemui dalam kehidupan sehari-hari, seperti benda-benda yang digunakan sehari-hari, kendaraan, dan rambu jalan. Hal ini membantu penyandang tunanetra berinteraksi dengan lingkungan dengan lebih efektif dan mandiri.

Deep learning, subbagian dari *machine learning*, memainkan peran penting dalam pengembangan sistem deteksi objek yang andal dan efisien. Dengan memanfaatkan kemampuan *deep learning*, memungkinkan untuk membangun sistem yang dapat mendeteksi dan mengenali objek dengan akurasi tinggi, memberikan rasa kemandirian dan kepercayaan diri yang lebih besar bagi penyandang tunanetra (Randy Moh Yusup et al., 2024). Salah satu algoritma *deep learning* yang populer untuk deteksi objek adalah *You Only Look Once (YOLO)*.

YOLO merupakan sebuah algoritma yang dihasilkan oleh Joseph Redmon untuk melakukan deteksi objek. YOLO mampu untuk melakukan object detection secara *real time*. Jika dibandingkan dengan sistem deteksi objek *real time* yang lain, YOLO memiliki *mAP* dan *FPS* yang lebih tinggi (Liunanda et al., 2020).

Penelitian ini bermaksud untuk membangun sebuah sistem deteksi objek yang efisien serta sangat penting untuk meningkatkan kualitas hidup dan kemandirian orang dengan keterbatasan penglihatan. Dengan menggunakan kemampuan *deep learning* seperti YOLO, sistem ini dapat membantu orang dengan keterbatasan penglihatan berinteraksi dengan lingkungannya dengan lebih mandiri.

Dalam upaya memberikan solusi yang inovatif dan signifikan bagi penyandang tunanetra, penelitian ini bertujuan untuk mengembangkan sistem deteksi objek berbasis *deep learning* yang efisien dan akurat. Dengan memanfaatkan algoritma YOLO sistem ini diharapkan dapat mendeteksi dan mengenali objek dalam *realtime*, memberikan informasi lingkungan yang relevan kepada pengguna.

Penerapan teknologi ini diharapkan dapat meningkatkan mobilitas, kemandirian, dan kualitas hidup penyandang tunanetra dengan memungkinkan mereka untuk berinteraksi dengan lingkungan sekitar secara lebih efektif dan percaya diri. Melalui penelitian ini, kami berharap dapat memberikan kontribusi nyata dalam mendukung inklusi dan kemandirian penyandang tunanetra, menjadikan dunia lebih ramah dan aksesibel bagi semua.

B. Rumusan Masalah

Berdasarkan keadaan latar belakang di atas, peneliti dapat merumuskan masalah berikut. merumuskan masalah sebagai berikut:

1. Bagaimana algoritma YOLO diimplementasikan pada sistem pendeteksi objek untuk membantu penyandang tunanetra?
2. Bagaimana mengetahui tingkat akurasi sistem pendeteksi objek yang dibangun?

C. Tujuan Penelitian

Berdasarkan rumusan masalah yang telah dipaparkan sebelumnya, penelitian ini bertujuan sebagai berikut:

1. Untuk mengetahui proses implementasi algoritma YOLO pada sistem pendeteksi objek untuk membantu penyandang tunanetra
2. Untuk mengetahui tingkat akurasi dari sistem pendeteksi objek yang dibangun

D. Manfaat Penelitian

Berdasarkan latar belakang, rumusan masalah, dan tujuan penelitian di atas, keuntungan dari penelitian ini dapat dibagi menjadi beberapa aspek sebagai berikut:

1. Aspek Teoritis
    a. Meningkatkan pengetahuan, terutama dalam bidang komputer dan informatika
    b. Mempelajari cara menggunakan metode YOLO untuk membuat sistem pendeteksi objek

2. Aspek Praktis
    a. Bagi Peneliti
        1) Dapat membuat publikasi ilmiah yang dapat meningkatkan reputasi peneliti dalam komunitas akademis.
        2) Akan meningkatkan pemahaman peneliti tentang proses kerja *Machine Learning* dan bentuk implementasinya

    b. Bagi Universitas
        1) Memberikan peningkatan pengetahuan dan pemahaman tentang *Machine Learning* ini merupakan aset bagi universitas untuk mendukung pendidikan, penelitian, dan layanan masyarakat.
        2) Sebagai referensi untuk penelitian yang lebih lanjut mengenai *Machine Learning*

E. Ruang Lingkup Penelitian

Dari rumusan masalah diatas, dapat dirumuskan beberapa ruang lingkup penelitian sebagai berikut:

1. Pemfokusan pada algoritma YOLO untuk membangun sistem pendeteksi objek yang khusus ditujukan bagi penyandang tunanetra.
2. Pengembangan sistem akan difokuskan pada pengenalan objek yang umum ditemui dalam kehidupan sehari-hari yang relevan bagi penyandang tunanetra terkhusus saat di jalan seperti: Orang, Mobil, Motor, Sepeda, Tongkat, Tangga, Lubang, Tiang, Guiding Block Garis, Guiding Block Titik.
3. Penelitian ini tidak akan mencakup pengujian terhadap penggunaan objek-objek yang tidak relevan atau tidak umum ditemui dalam kehidupan sehari-hari bagi penyandang tunanetra.
4. Penelitian tidak akan mempertimbangkan implementasi teknologi lain selain YOLO untuk deteksi objek, seperti deteksi berbasis suara atau sensor lainnya.

5. Penelitian ini tidak akan memasukkan aspek implementasi perangkat keras (*hardware*), namun akan berfokus pada pengembangan dan evaluasi algoritma perangkat lunak (*software*) yang digunakan dalam sistem pendeteksi objek.
6. Pada penelitian ini peneliti menggunakan kamera laptop Asus TUF *Gaming* FX505DT dengan jarak terdeteksi sejauh 5 meter.

F. Sistematika Penulisan

Untuk memberikan gambaran umum tentang penulisan ini, sistematika penulisan diterapkan yang diterapkan sebagai berikut:

BAB I PENDAHULUAN

Bab ini membahas secara singkat dan jelas latar belakang penulisan penelitian, rumusan masalah, tujuan dan manfaat dari penelitian, ruanglingkup penelitian, dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini membahas tentang teori-teori yang melandasi penulis untuk menyusun sebuah skripsi.

BAB III METODE PENELITIAN

Bab ini membahas tentang metode penelitian dan instrumen yang digunakan untuk pembuatan sistem

BAB IV HASIL DAN PEMBAHASAN

Bab ini menjelaskan hasil dari penelitian yang telah dilakukan sebelumnya, serta memaparkan hasil penelitian dan pengujian yang dilakukan.

BAB V PENUTUP

Bab ini berisi kesimpulan dari penelitian yang telah dilakukan serta memberikan saran untuk penelitian selanjutnya.

BAB II
TINJAUAN PUSTAKA

A. Landasan Teori

1. Tunanetra

Tunanetra adalah kondisi mata yang tidak berfungsi normal yang disebabkan oleh kerusakan, kekurangan, atau ketidakfungsian indra penglihatan (mata). Kondisi ini mencakup berbagai tingkat keparahan gangguan penglihatan, mulai dari penglihatan yang sangat terbatas hingga kebutaan total. Tunanetra dapat disebabkan oleh berbagai faktor, termasuk kelainan genetik yang mempengaruhi perkembangan mata, penyakit mata seperti glaukoma atau katarak yang merusak struktur mata, infeksi yang menyerang mata atau saraf optik, cedera fisik yang merusak jaringan mata, serta kondisi medis lainnya yang mempengaruhi sistem saraf atau pembuluh darah di mata. Istilah "tunanetra" sendiri berasal dari bahasa Jawa, terdiri dari dua kata: "tuna" yang berarti rusak, hilang, terhambat, terganggu, atau tidak memiliki, dan "netra" yang berarti mata. Gabungan dari kedua kata ini menggambarkan keadaan di mana seseorang mengalami gangguan signifikan dalam fungsi penglihatannya, sehingga mempengaruhi kemampuan mereka untuk berinteraksi dengan dunia sekitarnya secara visual (Pendidikan et al., 2020).

Gangguan penglihatan yang dialami oleh tunanetra tidak selalu berarti kebutaan total. Tunanetra umumnya dipahami sebagai orang yang tidak melihat sama sekali atau buta total. Namun, realitasnya lebih kompleks dan bervariasi. Sebagian besar tunanetra memiliki sisa penglihatan, yang berarti mereka masih memiliki kemampuan penglihatan meski sangat terbatas. Sisa penglihatan ini memungkinkan mereka untuk mendeteksi cahaya, bayangan, atau bentuk yang samar. Data menunjukkan bahwa hanya sekitar 10% dari populasi tunanetra yang mengalami kebutaan total, di mana mereka sama sekali tidak dapat melihat apa pun, bahkan cahaya terang. Bagi mereka yang memiliki sisa penglihatan, kemampuan ini bisa sangat bervariasi. Beberapa tunanetra mungkin hanya mampu membedakan antara terang dan gelap, yang masih memberikan mereka orientasi dasar dalam ruang. Sementara itu, yang lain mungkin masih dapat melihat bentuk atau warna dalam

tingkat keterbatasan tertentu, yang bisa membantu mereka dalam aktivitas sehari-hari. Oleh karena itu, istilah tunanetra total (Totally Blind) digunakan untuk merujuk kepada mereka yang benar-benar tidak memiliki kemampuan untuk membedakan antara terang dan gelap, menandakan hilangnya seluruh fungsi penglihatan mereka (Junaedy & Lukman, 2023).

2. *Artificial Intelligence* (AI)

AI atau dalam bahasa Indonesia dikenal sebagai Kecerdasan Buatan. Cabang ilmu ini berkonsentrasi pada pengembangan sistem dan mesin cerdas yang dapat meniru kemampuan kognitif manusia, seperti belajar, memahami, dan menyelesaikan masalah. Tujuan utama kecerdasan buatan adalah menciptakan mesin yang dapat melakukan tugas-tugas yang biasanya membutuhkan kecerdasan manusia, seperti pengenalan pola, pengambilan keputusan, dan interaksi dengan lingkungan sekitar.

Sistem AI memanfaatkan algoritma canggih dan model matematika untuk memungkinkan komputer dan perangkat lainnya mempelajari data, mengidentifikasi pola, dan mengambil keputusan secara cerdas. Konsep fundamental dalam AI meliputi pembelajaran mesin (*machine learning*), jaringan saraf tiruan (*neural networks*), dan pemrosesan bahasa alami (*natural language processing*). Penerapan AI telah merevolusi berbagai bidang, termasuk pengenalan suara, pengenalan wajah, kendaraan otonom, dan bidang medis (Eriana, 2023).

3. *Machine Learning*

Machine learning bagaikan perpaduan ilmu dan seni dalam pemrograman komputer, memungkinkan mesin untuk belajar dan berkembang dari data. Bidang ini terbagi menjadi empat kategori utama, yaitu:

Supervised Learning: Belajar dari data berlabel, seperti guru yang melatih murid. Contohnya: klasifikasi email spam.

1) *Unsupervised Learning*: Menemukan pola tersembunyi dalam data tanpa label, seperti arkeolog yang menggali situs kuno.

2) *Semi-Supervised Learning*: Gabungan supervised learning dan unsupervised learning.
3) *Reinforcement Learning*: Belajar dengan sistem reward dan penalties, seperti bermain game.

Tidak mungkin bagi model *Machine learning* untuk mengolah data dari berbagai sumber secara instan. Istilah "*Garbage In—Garbage Out*" menunjukkan bahwa jika input yang dimasukkan buruk, maka hasil pembelajaran mesin akan buruk juga. Untuk melakukan analisis teks dengan pengajaran mesin, hal-hal berikut harus dilakukan:

1) *Data Cleaning/ Preparation*

   Sebelum membangun model *machine learning*, data *training* perlu data *cleaning* terlebih dahulu. Tahapan yang termasuk dalam proses ini adalah Memastikan Konsistensi Format: Format data, skala data, duplikasi data, *missing value*, dan *skewness* (kemencangan yang menyebabkan distribusi data tidak seimbang) diperiksa dan diperbaiki.
2) *Data Preprocessing*

   Tahap ini melibatkan berbagai proses, dimulai dengan *case folding* (mengubah semua huruf menjadi huruf kecil). Kemudian, *tokenizing* dilakukan untuk menghapus angka, kata sambung, tanda baca, dan memecah kalimat menjadi token (kata-kata penyusunnya). Selanjutnya, kata-kata *stopword* seperti "yang", "dengan", dan kata penghubung lainnya dihapus.
3) *Data Normalization*

   Tahap ini penting untuk menstandariskan teks ke dalam format baku, baik dalam bahasa Inggris maupun Indonesia. Kata-kata slang atau gaul diubah menjadi padanan resminya. Hal ini bertujuan untuk meminimalisir bias dan meningkatkan akurasi model *machine learning* yang akan dibangun.
4) *Data Classifying*

   Klasifikasi adalah suatu teknik untuk menentukan kelas atau kategori data berdasarkan atribut yang diberikan. Teknik ini termasuk dalam kategori *supervised learning*, di mana model *machine learning* dilatih dengan

menggunakan data yang sudah memiliki label kelasnya. Tujuan dari model klasifikasi adalah untuk menentukan kelas baru berdasarkan atribut yang dimilikinya.

Dengan pengolahan data yang tepat dan pemilihan algoritma yang sesuai, *machine learning* dapat membuka berbagai peluang baru di berbagai bidang, mulai dari kesehatan dan keuangan hingga manufaktur dan ritel. Masa depan di mana mesin dapat membantu manusia dalam berbagai aspek kehidupan bukan lagi khayalan, tetapi sebuah kenyataan yang menanti untuk diwujudkan (Astuti, 2021).

4. *Computer Vision*

Computer vision, sebuah cabang dari kecerdasan buatan dan *machine learning*, berfokus pada pengembangan kemampuan komputer untuk "melihat" seperti manusia. Pada level abstrak, tujuannya adalah menggunakan data gambar yang diamati untuk memahami dunia di sekitarnya.

Bidang ini memanfaatkan metode khusus dan algoritma pembelajaran umum untuk mencapai tujuannya. *Computer vision* bertujuan untuk memahami isi gambar digital, replikasi kemampuan penglihatan manusia melalui pelatihan dengan data yang tepat. Pemahaman isi gambar ini mencakup ekstraksi informasi seperti objek, deskripsi teks, dan model tiga dimensi (Astuti, 2021).

5. Deteksi Objek

Deteksi objek (*Object Detection*) biasanya menggunakan *mesin learning* atau *deep learning* untuk menghasilkan hasil yang signifikan. Deteksi objek adalah teknik visi komputer untuk menemukan contoh objek dalam gambar atau video. Berbeda dengan komputer, yang membutuhkan komputasi yang kompleks, manusia dapat mengenali dan menemukan objek dalam beberapa saat ketika mereka melihat gambar atau video.

Tujuan deteksi objek adalah untuk menggunakan komputer untuk meniru kecerdasan manusia dalam melihat objek. Deteksi objek bekerja dengan dua proses: mengklasifikasikan jenis objek dan kemudian menggambar kotak di sekitarnya. Ini menempatkan objek dalam gambar dan menggambar kotak pembatas di sekitarnya.

Senario klasifikasi gambar dan skenario deteksi objek serupa. Secara umum, skenario klasifikasi adalah mengkategorikan gambar ke dalam kategori tertentu, sedangkan skenario deteksi objek adalah mengidentifikasi lokasi objek dalam gambar dan menghitung jumlah instancenya (Prisky Ratna Aningtiyas, 2020).

6. *Deep Learning*

Deep Learning adalah abang kecerdasan buatan canggih yang terinspirasi dari struktur otak manusia, menggunakan jaringan saraf tiruan berlapis-lapis untuk mencapai performa luar biasa dalam tugas-tugas seperti deteksi objek, pengenalan suara, dan terjemahan bahasa.

Bayangkan sebuah komputer yang mampu mempelajari pola kompleks dari gambar, memahami percakapan manusia, dan bahkan menerjemahkan bahasa dengan tingkat akurasi yang tinggi. Itulah yang ditawarkan oleh Deep Learning, sebuah terobosan dalam kecerdasan buatan (AI) yang meniru cara kerja otak manusia. *Deep Learning* memanfaatkan jaringan saraf tiruan (*artificial neural network*) yang tersusun atas berlapis-lapis neuron, seperti neuron pada otak manusia. Neuron-neuron ini saling terhubung dan berkomunikasi satu sama lain, memungkinkan jaringan untuk belajar dan beradaptasi dengan informasi baru (Raup et al., 2022).

7. PyTorch

PyTorch, bagaikan pisau *Swiss Army* di dunia *deep learning*, hadir sebagai salah satu pustaka (*library*) terpopuler dalam bahasa pemrograman python. Dirancang khusus untuk komputasi *deep learning*, PyTorch membuka gerbang bagi penggunanya untuk menyelami analisis mendalam pada objek digital seperti gambar (2D) dan video (3D). Kekuatannya terletak pada kemampuannya untuk digabungkan dengan pustaka *torchvision*, yang memperluas cakupan analisis dan membuka peluang baru dalam memahami dan memanipulasi data visual.

Di jantung PyTorch *lies the concept of tensors*, struktur data multidimensional yang dioptimalkan untuk komputasi GPU. Tensor menjadi landasan bagi pembangunan model *deep learning* yang kompleks, memungkinkan pengguna

untuk memanipulasi data dengan presisi dan efisiensi tinggi. Kemampuan ini menjadikannya pilihan ideal bagi para peneliti dan praktisi yang ingin menjelajahi potensi *deep learning* dalam berbagai aplikasi, seperti pengenalan gambar, klasifikasi objek, dan pemrosesan video.

Salah satu teknik populer untuk analisis gambar yang digemari para pengguna PyTorch adalah CNN. Terinspirasi oleh struktur neuron kortikal, CNN mampu mengenali pola dan fitur kompleks dalam gambar dengan cara yang mirip dengan bagaimana otak manusia memproses informasi visual. *Torchvision*, sebagai pelengkap PyTorch, menyediakan berbagai arsitektur CNN yang siap pakai, seperti ResNet dan VGG, yang telah terbukti efektif dalam berbagai tugas pengenalan gambar. Hal ini menjadikan torchvision sebagai alat yang sangat andal untuk membangun model CNN yang kuat dan akurat.(Nur et al., 2023)

8. YOLO

Metode YOLO adalah salah satu cara untuk membuat sistem pengenalan objek. YOLO adalah nama aplikasi yang dapat diakses melalui DARKNET, yang merupakan neural network open source. Cara kerja dari YOLO adalah dengan melihat gambar secara keseluruhan sekali, melewati neural network sekali, dan kemudian menemukan objek yang ada, sebab itu dinamakan YOLO (Randy Moh Yusup et al., 2024).

Bayangkan sebuah sistem AI yang mampu mendeteksi objek dalam gambar dengan kecepatan luar biasa, hanya dengan sekali pandang. Itulah yang ditawarkan oleh YOLO, sebuah jaringan deteksi objek revolusioner yang diciptakan oleh Joseph Redmon di tahun 2016. Berbeda dengan algoritma deteksi objek tradisional yang kompleks, YOLO bekerja dengan cara yang jauh lebih sederhana. Jaringan saraf tiruan CNN menjadi satu-satunya komponen utama YOLO. CNN ini bertugas memprediksi beberapa kotak (*bouding box*) dan probabilitas kelas (kelas objek) untuk setiap kotak (Liunanda et al., 2020).

YOLO memulai aksinya dengan membagi gambar input menjadi grid dengan ukuran SxS. Setiap sel grid ini kemudian diolah oleh CNN untuk menghasilkan *bouding box* dan prediksi kelasnya.Hebatnya lagi, YOLO tidak hanya memprediksi

satu kotak untuk setiap sel grid. Setiap sel grid diprediksikan memiliki B *bouding box* dengan *confidence score* masing-masing. *Confidence score* ini menunjukkan tingkat keyakinan model bahwa objek di dalam kotak tersebut memang sesuai dengan kelas yang diprediksikan. YOLO menghitung *confidence score* dengan rumus Pr(*Object*) * IOUtruth (*Intersection of Union*). Pr(*Object*) mewakili probabilitas terdapat objek dalam kotak, sedangkan IOUtruth mewakili tingkat tumpang tindih antara kotak prediksi dengan kotak sebenarnya (ground truth) dari objek.

Setiap *bouding box* memiliki lima prediksi: x, y, w, h. Koordinat (x, y) mewakili pusat kotak relatif terhadap batas grid cell. Lebar (*width*) dan tinggi (*height*) adalah prediksi relatif terhadap keseluruhan gambar. Setiap grid cell memprediksi probabilitas kelas bersyarat C, Pr(*Class* | *Object*). Probabilitas ini diasosiasikan dengan grid cell yang mengandung objek (Liunanda et al., 2020).

Gambar 1. Arsitektur YOLO (Randy Moh Yusup et al., 2024).

9. YOLOv8

YOLOv8, dirilis oleh Ultralytics pada Januari 2023 (tim yang sama yang merilis YOLOv5 di tahun 2020), dapat dijalankan melalui antarmuka baris perintah (CLI) atau diinstal sebagai paket PIP. YOLOv8 dilengkapi dengan integrasi untuk pelabelan, pelatihan, dan implementasi. Beberapa perubahan telah dilakukan untuk meningkatkan akurasi model, termasuk peningkatan selama proses pelatihan.

YOLOv8 melakukan augmentasi gambar secara online selama pelatihan. Pada setiap era, model melihat variasi gambar yang sedikit berbeda dari yang disediakan.

YOLOv8 menggunakan augmentasi mosaik selama pelatihan, tetapi karena ditemukan dapat menurunkan akurasi jika digunakan selama seluruh proses, augmentasi ini dinonaktifkan untuk sepuluh era terakhir. Augmentasi mosaik adalah teknik di mana 4 gambar pelatihan digabungkan untuk membentuk gambar baru, membantu model berlatih dan belajar lebih efisien. YOLOv8 lebih efisien daripada versi sebelumnya karena menggunakan peta fitur yang lebih besar dan jaringan konvolusi yang lebih efektif (Tamang et al., 2023).

Algoritma YOLOv8 terdiri dari jaringan saraf konvolusi CNN dengan 24 lapisan untuk mengekstraksi fitur, serta 2 lapisan fully connected untuk memprediksi probabilitas dan koordinat objek dalam gambar. Setelah program Python dijalankan dan diproses oleh Algoritma YOLOv8, monitor yang terhubung ke PC akan menampilkan hasil deteksi objek. Sistem penelitian ini mengintegrasikan ESP32CAM dengan Algoritma YOLOv8 untuk deteksi objek *real-time* yang akurat dan efisien.

Algoritma YOLOv8 memecah gambar menjadi grid untuk mendeteksi objek di dalamnya. Lapisan CNN mengekstraksi fitur penting, sementara lapisan *fully connected* menghitung probabilitas dan koordinat kotak pembatas objek. Hasil deteksi ditampilkan dengan kotak pembatas dan label jenis objek serta tingkat kepercayaan.

Integrasi ESP32CAM dengan Algoritma YOLOv8 memungkinkan deteksi objek *real-time* dengan biaya rendah dan efisiensi tinggi. ESP32CAM, modul kamera murah, bekerja dengan YOLOv8 untuk deteksi objek cepat dan akurat. Penggunaan Python memudahkan pengembangan dan pemeliharaan sistem. Penelitian ini menunjukkan potensi besar dalam menggabungkan teknologi sederhana dengan algoritma canggih untuk aplikasi di berbagai bidang.(Yudha et al., 2024)

10. *Text to Speech* (TTS)

Konversi kata-kata menjadi suara adalah proses yang dikenal sebagai TTS. Setelah alat mengumpulkan teks pengguna, itu memprosesnya dan membuat kesimpulan logis tentangnya. Kemudian, teks ini dikirim ke blok berikutnya, di mana sinyal digital diproses. Banyak algoritma yang digunakan untuk mengubah teks menjadi format bicara. Sepanjang proses ini, sintesis pidato diperlukan. Dalam Python, ada beberapa API yang dapat digunakan untuk mengkonversi teks menjadi suara. API Google *Text to Speech*, juga dikenal sebagai gTTS, adalah alat yang mudah digunakan untuk mengkonversi teks menjadi suara yang dapat disimpan sebagai file mp3. API gTTS mendukung berbagai bahasa, seperti bahasa Inggris, Prancis, Jerman, Indonesia dan banyak lagi (Sarmah et al., 2023).

Menurut Samsudin & Putra, TTS adalah alat yang dapat mengubah teks menjadi suara yang dapat didengar. Berdasarkan teori mereka, TTS dapat menjadi media pembelajaran berbasis audio yang membantu pembicara dalam pengucapan yang lebih jelas. Namun, perlu diingat bahwa suara yang dihasilkan TTS adalah hasil manipulasi mesin komputer di ranah digital Keuntungan penggunaan TTS untuk membantu pembicara adalah kemampuannya dalam membaca teks menjadi suara yang terdengar seperti pengucapan manusia asli (Manu et al., 2020).

Berdasar definisi TTS dari pakar di atas, dapat disimpulkan bahwa TTS merupakan sebuah sistem program yang secara otomatis menghasilkan suara dari teks apa pun melalui konversi grafem ke fonem, dengan memanfaatkan sampel suara yang mirip dengan suara asli sebagai suara sintetis. Sistem ini dapat digunakan secara online maupun melalui *installer* (Manu et al., 2020).

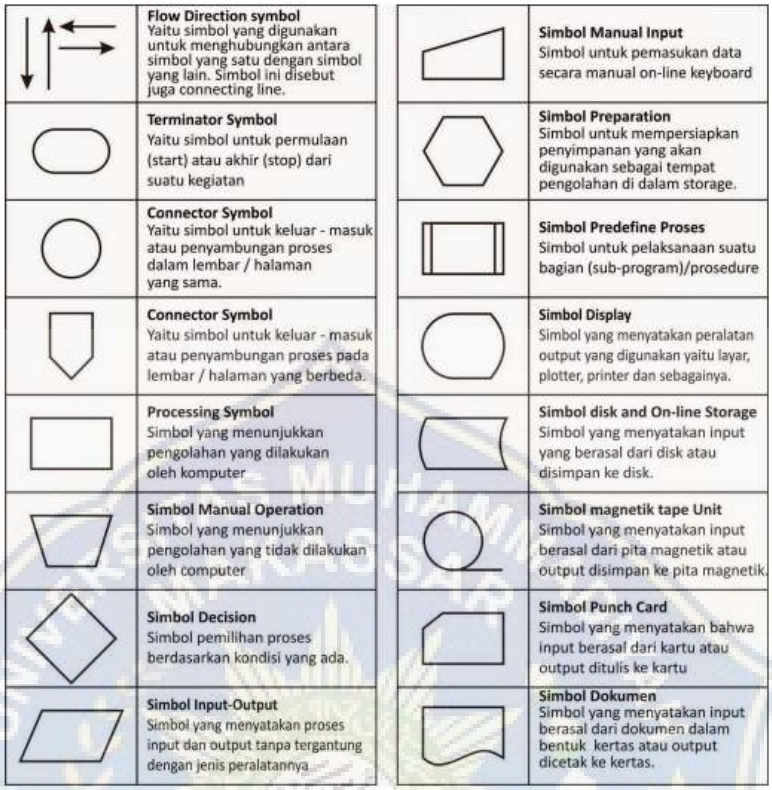
11. Python

Python adalah bahasa pemrograman yang dirancang untuk mudah dipahami dan digunakan. Hal ini membuatnya ideal bagi pemula maupun programmer berpengalaman. Python juga menawarkan banyak fitur dan pustaka standar yang lengkap, sehingga dapat digunakan untuk berbagai macam proyek.

Utamanya, Python mendukung berbagai paradigma pemrograman, termasuk pemrograman berorientasi objek, pemrograman imperatif, dan pemrograman fungsional. Python dapat digunakan sebagai bahasa pemrograman dinamis dengan manajemen memori otomatis, seperti halnya bahasa pemrograman dinamis lainnya. Namun, python biasanya digunakan sebagai bahasa skrip, mencakup aplikasi yang biasanya tidak dilakukan dengan bahasa skrip. Python dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan berjalan di berbagai *platform* sistem operasi saat ini (Fauzi et al., 2020).

12. *Flowchart*

Seorang analis sistem menggunakan *flowchart*, yang juga disebut diagram alir, sebagai bukti untuk menjelaskan algoritma atau langkah-langkah instruksi yang berurutan yang ada dalam sistem. *flowchart* yang menunjukkan gambaran logis sistem yang akan dibangun kepada programmer. Ini memungkinkan programmer untuk menyelesaikan masalah yang mungkin muncul saat membangun sistem. Pada dasarnya, *flowchart* menggunakan simbol untuk menunjukkan suatu proses tertentu. Garis penghubung juga digunakan untuk menunjukkan hubungan antara proses. *flowchart* membantu menjelaskan setiap urutan proses lebih jelas dan memudahkan penambahan proses baru. Setelah proses selesai, programmer akan menerjemahkan desain logis tersebut ke dalam program yang ditulis dalam berbagai bahasa pemrograman yang telah disepakati (Rosaly, 2019).



Gambar 2. Simbol-simbol *Flowchart* (Fauzi et al., 2020).

B. Penelitian Terkait

1. Pendeteksi Objek Menggunakan *OpenCV* dan Metode YOLOv4-tINY Untuk Membantu Tunanetra (Randy Moh Yusup et al., 2024).

   Pada penelitian yang dilakukan oleh Randy Moh Yusup, Aldof Faris Anugrah, Dinda Desmonda Muslimah , Sri Mentari Widya Ningrum Permana, Shindi Yuliani penelitian tersebut bertujuan untuk membangun sebuah sistem deteksi objek yang handal dan efisien menggunakan *OpenCV* dan YOLOv4-Tiny. Peneliti mengevaluasi kinerja sistem pada tolok ukur standar dan mengeksplorasi potensinya untuk aplikasi *real-time*. Penelitian ini diharapkan dapat berkontribusi pada kemajuan dalam bidang visi komputer dan berbagai penerapannya.

   Hasil dari penelitian tersebut menunjukkan bahwa YOLOv4-Tiny menawarkan keseimbangan antara akurasi dan kecepatan. Meskipun

akurasinya (*mAP* 20-30%) lebih rendah dibandingkan dengan model YOLOv4 penuh dan beberapa model lain, YOLOv4-Tiny tetap mampu mendeteksi berbagai objek (orang, kendaraan, hewan, dll.) dengan baik. Kemampuan deteksi objek kecilnya mungkin sedikit terbatas karena arsitekturnya yang diperkecil. Namun, kelebihan utama YOLOv4-Tiny adalah kecepatan inferensinya yang jauh lebih cepat dibandingkan dengan YOLOv4 penuh. Hal ini membuatnya ideal untuk aplikasi *real-time* pada perangkat dengan sumber daya terbatas, seperti sistem tertanam atau perangkat edge. Kecepatan inferensi yang tinggi memungkinkan YOLOv4-Tiny untuk memproses streaming video atau umpan kamera secara *real-time*. Meskipun akurasinya tidak setinggi model lain, YOLOv4-Tiny tetap menjadi pilihan yang menarik untuk aplikasi *real-time* yang membutuhkan kecepatan tinggi.

2. Pengembangan Aplikasi Pendeteksi Objek Untuk Tunanetra Menggunakan Operator Tepi Canny Dengan LIbrary *OpenCV* Berbasis Android (Pradana & Sula, 2023).

   Pada penelitian yang di lakukan Teguh Pradana, Eko Hajianto Sula penelitian tersebut bertujuan untuk mengembangkan aplikasi pendeteksi objek berbasis Android yang dapat digunakan oleh penyandang tunanetra dan anak usia dini. Aplikasi ini bertujuan untuk membantu pengguna dalam mengenali objek-objek di sekitar mereka dengan cara yang mudah dan interaktif.

   Hasil dari penelitian tersebut Penelitian ini menghasilkan aplikasi pendeteksi objek berbasis Android untuk membantu penyandang tunanetra dan anak usia dini. Halaman utama aplikasi menampilkan logo, nama, versi aplikasi, nama pembuat, dan tombol untuk dua mode pendeteksian: deteksi tepi objek dan deteksi objek terdekat. Mode deteksi tepi menggunakan kamera smartphone untuk menampilkan tepi objek dengan latar belakang hitam dan objek berwarna putih. Pengguna dapat menyimpan gambar hasil deteksi untuk pembelajaran anak usia dini. Mode deteksi objek terdekat

menampilkan hasil deteksi dengan label objek dan persentase akurasi, serta mengonversi label objek menjadi suara untuk membantu tunanetra. Penelitian ini menggunakan Operator Tepi Canny dan *OpenCV* untuk deteksi tepi, serta TensorFlow *Lite* untuk deteksi objek. Saran pengembangan meliputi penambahan data model untuk lebih banyak objek dan penyimpanan otomatis gambar hasil deteksi. Penelitian ini berhasil menciptakan aplikasi yang bermanfaat dan dapat ditingkatkan dengan fitur tambahan dan data yang lebih luas.

3. Deteksi dan Pengenalan Objek Dengan Model Machine Learning: Model Yolo (Aini et al., 2021).

Pada penelitian yang di lakukan Qurotul Aini, Ninda Lutfiani, Hendra Kusumah, Muhammad Suzaki Zahran penelitian tersebut bertujuan untuk menjelaskan dan memperkenalkan model algoritma YOLO (You Only Look Once) dalam konteks pengenalan dan pendeteksian objek. Penelitian ini dilakukan dengan tujuan untuk mengevaluasi kemampuan YOLO dalam mengenali dan mendeteksi objek pada gambar, serta membandingkannya dengan model algoritma yang telah ada sebelumnya. Selain itu, penelitian ini juga bertujuan untuk mengidentifikasi kelebihan dan kekurangan dari YOLO, termasuk masalah yang masih dihadapi oleh versi-versi terbaru seperti YOLO v3 dan YOLO v5. Metode literatur *review* digunakan dalam penelitian ini untuk memperoleh informasi yang dibutuhkan dengan membandingkan berbagai jurnal ilmiah yang relevan dalam ranah pengenalan dan pendeteksian objek. Dengan demikian, penelitian ini diharapkan dapat memberikan pemahaman yang lebih baik tentang model algoritma YOLO dan kontribusinya dalam bidang Computer Vision, serta menyoroti tantangan dan potensi pengembangan yang terkait dengan model tersebut.

Hasil dari penelitian tersebut perkembangan algoritma pengenalan objek YOLO, mulai dari versi awal hingga versi terbaru. YOLO diciptakan untuk menjawab kebutuhan akan model pengenalan objek yang cepat dan

akurat. Versi awal YOLO, seperti YOLO v1, memiliki kecepatan tinggi namun akurasinya masih rendah. Seiring perkembangannya, YOLO terus ditingkatkan dengan berbagai versi baru, seperti YOLO v2, YOLO9000, YOLO v3, dan YOLO v4. Setiap versi memiliki fokus pengembangan yang berbeda, seperti peningkatan recall dan lokalisasi, pengoptimasian komputasi paralel, dan lain sebagainya. Versi terbaru YOLO, yaitu YOLO v5, masih tergolong baru dan belum memiliki banyak informasi yang tersedia. YOLO v5 menggunakan *framework* PyTorch dan hadir dalam berbagai versi dengan spesifikasi dan kegunaannya masing-masing. Secara keseluruhan, YOLO merupakan algoritma pengenalan objek yang terus berkembang dengan berbagai kelebihan dan kekurangan di setiap versinya. YOLO v5, sebagai versi terbaru, masih perlu diteliti lebih lanjut untuk mengetahui potensinya secara lebih mendalam.

4. Aplikasi Penghitung Jarak dan Jumlah Orang Berbasis YOLO Sebagai Protokol Kesehatan Covid-19 (Indaryanto et al., 2021)

   Pada penelitian yang di lakukan Faizal Indaryanto, Anan Nugroho, Alfa Faridh Suni penelitian tersebut bertujuan untuk mengembangkan aplikasi social distancing detector yang bertujuan untuk mendeteksi jumlah dan jarak antar objek manusia pada suatu area. Penelitian ini merespon perlunya penanganan yang tepat terhadap penyebaran Covid-19 dengan mengimplementasikan konsep Social Distancing. Aplikasi ini dibangun menggunakan bahasa pemrograman Python dengan memanfaatkan *library* YOLOv3 yang terkenal karena tingkat akurasi deteksinya yang tinggi, mencapai di atas 90%. Pengujian metode dilakukan dengan menggunakan lima dataset pejalan kaki dari kamera pengawas jalan yang memiliki resolusi baik dan beragam objek manusia. Hasil pengujian menunjukkan bahwa aplikasi memiliki tingkat akurasi yang cukup tinggi, dengan rata-rata keberhasilan sebesar 90,04% berdasarkan analisis terhadap jumlah data percobaan berhasil dan data pengamatan untuk setiap citra. Dengan demikian, penelitian ini bertujuan untuk memberikan solusi teknologi yang

efektif dalam mendukung penerapan Social Distancing sebagai langkah pencegahan penyebaran Covid-19 di masyarakat.

Hasil dari penelitian tersebut adalah pengembangan sebuah aplikasi yang dapat digunakan untuk mendeteksi jumlah orang dan mengukur jarak antara individu. Berdasarkan hasil pengujian, aplikasi ini menunjukkan hasil akurasi yang baik, dengan rata-rata akurasi sebesar 90,04%. Hal ini menunjukkan bahwa tujuan dari penelitian ini telah tercapai, dan aplikasi ini memiliki potensi untuk diterapkan dalam lingkungan masyarakat guna membantu meredakan penyebaran Covid-19 yang semakin meningkat. Meskipun demikian, terdapat beberapa faktor yang dapat mempengaruhi akurasi dari aplikasi ini, seperti kejelasan input yang digunakan dan kontras antara objek manusia dengan latar belakang. Selain itu, penelitian ini mengalami keterbatasan dalam penggunaan dataset karena keterbatasan sumber daya yang tersedia. Namun demikian, aplikasi ini diharapkan dapat meminimalisir penyebaran Covid-19 dan menjadi landasan untuk penelitian lanjutan yang melibatkan kumpulan data yang lebih besar dan lebih baik, sehingga akurasi aplikasi dapat ditingkatkan. Dengan demikian, hasil dari penelitian ini memberikan kontribusi dalam upaya penanggulangan pandemi Covid-19 melalui pengembangan teknologi yang dapat membantu masyarakat dalam menjaga jarak sosial dan menerapkan protokol kesehatan yang tepat.

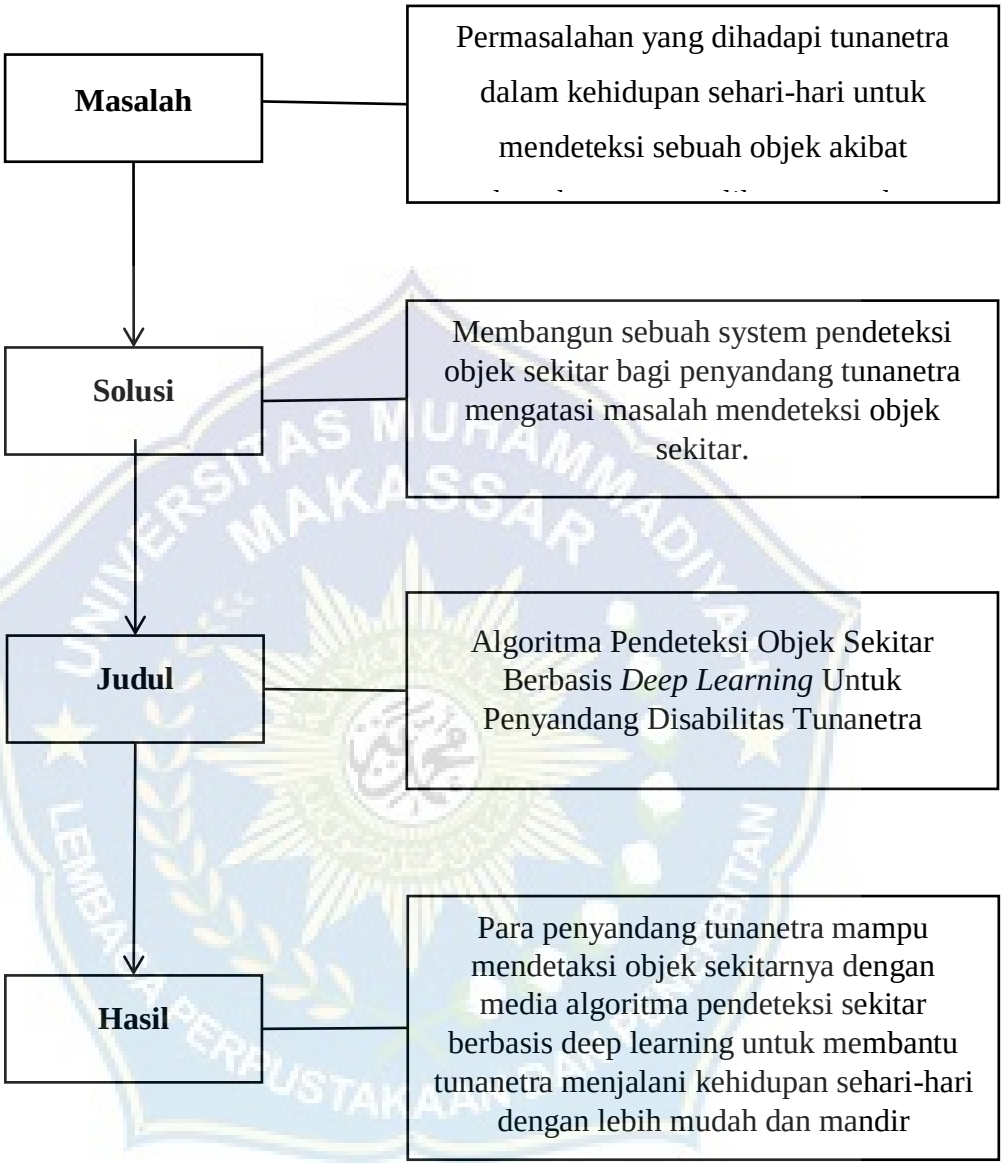
5. Pembuatan Aplikasi Deteksi Objek Menggunakan Memanfaatkan SSD *MobileNet* V2 Sebagai Model *TensorFlow Object Detection* API dengan Pra-Terlatih (Prisky Ratna Aningtiyas, 2020).

   Pada penelitian yang di lakukan Prisky Ratna Aningtiyas, Agus Sumin, Setia Wirawan . Penelitian ini bertujuan untuk menerapkan ilmu *Deep Learning* dalam mendeteksi objek seperti kamera, handphone, headphone, laptop, dan mouse melalui input gambar. Aplikasi ini diharapkan dapat mengidentifikasi dan mengukur akurasi deteksi objek tersebut. Selain itu, penelitian ini juga bertujuan untuk mengevaluasi

performa aplikasi dengan dataset yang dibagi menjadi *train set*, *validation set*, *dan test set*, dan untuk menentukan tingkat akurasi aplikasi yang dihasilkan, yang mencapai 93.02% pada *test* set.

Hasil dari penelitian tersebut berdasarkan penelitian yang telah dilakukan, berhasil dibuat aplikasi untuk mendeteksi objek dengan menggunakan *TensorFlow Object Detection API* dengan memanfaatkan SSD *Mobilenet* V2 sebagai model pra terlatih. Program dapat mendeteksi 5 kategori kelas objek, yaitu Camera, Handphone, Headphone, Laptop, dan Mouse. Aplikasi dapat menampilkan tingkat pengukuran akurasi masing-masing objek. Hasil pengujian yang dilakukan pada 50 test set dengan jumlah masing masing kelas objek adalah 10 gambar. Gambar objek Camera mendapatkan rata-rata presetase sebesar 99%, Handphone sebesar 89.1%, Head- phone sebesar 89.1%, Laptop sebesar 89.1%, dan Mouse sebesar 98.8%. sehingga diperoleh rata-rata akurasi keberhasilan pendeteksian aplikasi deteksi objek adalah 93.02%. Pengembangan diharapkan dapat membuat aplikasi dapat mendeteksi lebih dari satu objek dalam satu gambar. Selain itu, dapat dilakukan penambahan fitur *real time* image processing dan pengembangan aplikasi yang dapat dijalankan pada berbagai *platform* seperti pada android maupun *website*

C. Kerangka Pikir



Gambar 3. Kerangka Berfikir

BAB III
METODE PENELITIAN

A. Tempat dan Waktu Penelitian

1. Tempat Penelitian

Penelitian ini dilakukan dengan mengambil gambar menggunakan kamera handphone pada objek-objek sekitar dan mengambil data berupa gambar objek-objek yang tersedia pada internet guna memperbanyak data

2. Waktu Penelitian

Adapun waktu pelaksanaan penelitian ini di lakukan dalam kurung waktu 2 yakni pada bulan Mei – Juli 2024

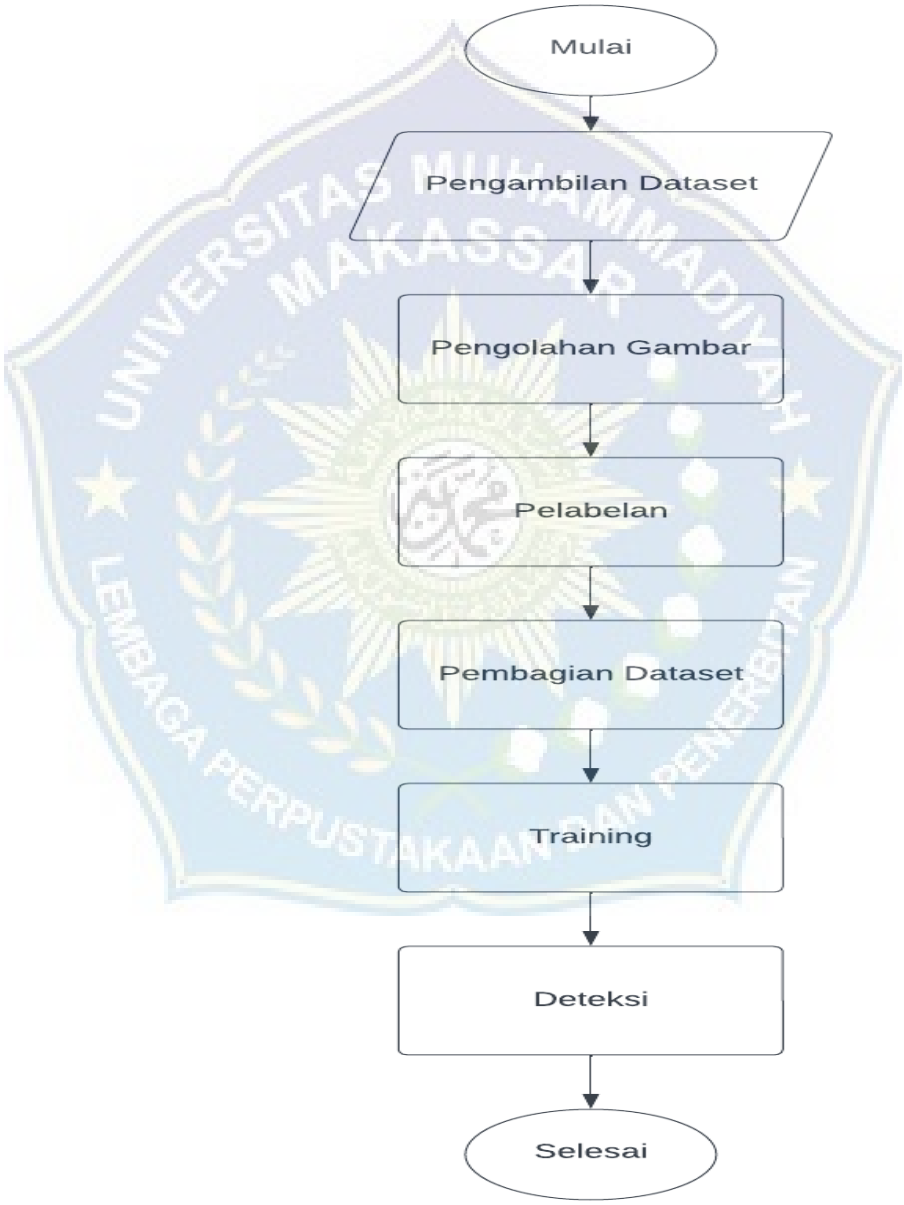
B. Alat dan Bahan

Adapun alat dan bahan yang akan digunakan dalam penelitian ini adalah sebagai berikut :

1. Kebutuhan *Hardware* (Perangkat Keras)
   a. Asus TUF Gaming FX505DD
   b. Infinix Smart 8
2. Kebutuhan *Software* (Perangkat Luna)
   a. VS Code
   b. Goggle Colab
   c. Python
   d. Roboflow (Untuk annotasi gambar)

C. Perancangan Sistem

Flowchart bagaikan peta untuk suatu program. Peta ini menunjukkan langkah demi langkah yang perlu dilakukan, seperti petunjuk arah garis dan panah menghubungkan langkah-langkah tersebut membuat prosesnya lebih mudah. Berikut adalah *flowchart* perancangan algoritma pendeteksi objek sekitar menggunakan *deep learning*



Gambar 4. Perancangan Sistem

1. Pengambilan Dataset

   Data diambil dalam bentuk gambar dengan menggunakan kamera pada handphone untuk mengambil gambar dari objek-objek yang telah di tentukan dan juga berupa gambar objek-objek yang tersedia pada internet guna mengumpulkan data. Nantinya dataset ini akan digunakan sebagi contoh untuk melatih sistem pendeteksi objek yang sedang dikerjakan

2. Pengolahan Gambar

   Pada tahap *preprocessing* data, akan dilakukan proses *resize* untuk mengubah ukuran gambar-gambar yang ada. Proses ini penting untuk memastikan bahwa semua gambar memiliki ukuran yang seragam dan sesuai dengan persyaratan input dari model yang akan dilatih. Selain itu, proses *resize* juga membantu dalam mengoptimalkan penggunaan memori dan sumber daya komputasi selama pelatihan model. Dengan demikian, langkah ini merupakan bagian krusial dalam mempersiapkan data gambar sebelum digunakan untuk pelatihan model *machine learning*.

3. Pelabelan Dataset

   Proses ini akan melabeli data yang telah diproses menggunakan Roboflow. Tujuan Roboflow adalah untuk memberikan identifikasi atau label pada setiap gambar objek dalam sistem deteksi objek yang dibuat.

4. Pembagian Dataset

   Setelah dataset diberi label, dataset akan dibagi menjadi tiga bagian: data *training*, data validasi, dan data testing. Model akan dilatih menggunakan data yang ada dalam bagian *training*. Bagian data validasi digunakan untuk mengevaluasi kinerja model dengan mengidentifikasi dan menguji data, sehingga memungkinkan penilaian akurasi model terhadap data baru yang belum pernah dilihat sebelumnya.

5. Training

   Pada tahap ini, data yang telah dibagi menjadi data *training* dan data testing akan dipanggil dan diproses. Data dalam bentuk API yang tersimpan dalam LabelImg akan diakses melalui aplikasi Gogle Colab. Setelah data tersedia, pengujian model dilakukan dengan menggunakan algoritma

YOLOv8. Akurasi akan digunakan sebagai parameter utama untuk mengevaluasi kinerja model. Jika hasil pengujian menunjukkan bahwa akurasi model masih rendah, maka proses pelatihan ulang akan dilakukan untuk meningkatkan akurasi. Pelatihan ulang ini bertujuan untuk mengoptimalkan model agar dapat menghasilkan prediksi yang lebih baik lagi.

6. Deteksi

Proses pendeteksian dimulai dengan memasukkan gambar yang telah terkumpul ke dalam sistem. Setelah gambar dimasukkan, sistem menginisiasi proses analisis menggunakan YOLOv8 untuk mengenali objek-objek. Selama proses pengujian, algoritma YOLOv8 digunakan untuk menandai objek-objek yang ditemukan dalam gambar dengan *bouding box* sebagai label. Ini memungkinkan model untuk mengklasifikasikan objek dengan tepat dan memisahkan mereka dari latar belakang. Output dari proses ini mencakup nilai akurasi, yang merupakan ukuran seberapa baik model dapat mengidentifikasi objek, serta ROI (*Region of Interest*) untuk menentukan area deteksi yang relevan.

D. Cara Kerja YOLO

Berikut adalah *flowchart* cara kerja YOLO yang mana melewati 6 proses yaitu *Preprocessing*, Membagi Gambar Menjadi Grid, Melewatkan Setiap Sel Grid Lewat Darknet, Prediksi *Bouding box* & Probabilitas Kelas, *Non-Max Suppression (NMS)*, Gambar *Bouding box* & Label Kelas

Gambar 5. Cara Kerja YOLO

1. Memasukkan Gambar

   Proses dimulai dengan gambar yang ingin Anda analisa untuk deteksi objek.

2. *Preprocessing*

Gambar melalui tahap *preprocessing* seperti *resize*, normalisasi, atau konversi ruang warna agar kompatibel dengan model YOLO.

3. Membagi Gambar Menjadi Grid

   Gambar dibagi menjadi grid dengan sel berukuran sama. Setiap sel akan bertanggung jawab memprediksi objek di dalam batasnya.

4. Melewatkan Setiap Sel Grid Lewat Darknet

   Setiap sel grid dan area sekitarnya dilewatkan melalui arsitektur jaringan saraf konvolusi (CNN) Darknet. Darknet adalah komponen inti YOLO yang melakukan deteksi objek.

5. Prediksi *Bouding box* & Probabilitas Kelas

   CNN Darknet memprediksi *bouding box* (koordinat untuk daerah persegi panjang di sekitar objek) dan probabilitas kelas (skor keyakinan untuk setiap kelas objek) untuk setiap sel grid.

6. *Non-Max Suppression (NMS)*

   NMS adalah teknik yang digunakan untuk menghilangkan *bouding box* yang berlebihan. Jika beberapa *bouding box* saling tumpang tindih secara signifikan untuk objek yang sama, NMS menyimpan yang memiliki skor keyakinan tertinggi dan membuang sisanya.

7. Gambar *Bouding box* & Label Kelas

   Berdasarkan prediksi *bouding box* akhir dan label kelas yang sesuai, algoritme menggambar persegi panjang di sekitar objek yang terdeteksi dan memberi label dengan nama kelas yang diprediksi.

8. Gambar Keluaran

   Gambar akhir dihasilkan dengan *bouding box* dan label kelas yang dilapisi pada gambar asli, menyoroti objek yang terdeteksi.

E. Teknik Pengujian Sistem

Sistem pengujian empiris dalam penelitian algoritma pendeteksi objek sekitar berbasis *deep learning* untuk penyandang disabilitas tunanetra dengan metode metode YOLOv8 merupakan proses yang krusial untuk memastikan keefektifan

dan keandalan sistem yang dikembangkan. Pengujian ini melibatkan serangkaian langkah dan metode yang dirancang untuk mengevaluasi berbagai aspek kinerja sistem, seperti akurasi, presisi, dan recall dalam mendeteksi objek di sekitar tunanetra

Berikut adalah beberapa langkah umum yang biasanya dilakukan dalam pengujian empiris:

1. Pengumpulan Dataset:

   Tahap awal pengembangan sistem melibatkan pengumpulan dataset yang berisi berbagai objek sehari-hari yang relevan bagi penyandang tunanetra. Dataset ini kemudian perlu dibagi menjadi beberapa subset, yaitu subset pelatihan, subset validasi, dan subset pengujian. Setiap gambar dalam dataset harus diberi label yang menunjukkan posisi dan nama objek yang ada di dalamnya.
2. Pembuatan Model

   Model deteksi objek *deep learning*, seperti YOLOv8, merupakan salah satu alat yang ampuh untuk mendeteksi objek dalam gambar dan video. Namun, model ini tidak selalu dapat langsung digunakan dalam penelitian, karena perlu disesuaikan dengan kebutuhan spesifik penelitian.
3. Pelatihan Model

   Model deteksi objek dilatih dengan menggunakan subset data dari dataset yang telah disiapkan. Proses ini melibatkan beberapa iterasi, di mana model secara bertahap belajar mengenali pola dan fitur yang terkait dengan objek-objek dalam gambar.
4. Validasi Model

   Setelah model *deep learning* selesai dilatih, tahapan selanjutnya adalah mengujinya dengan menggunakan data validasi yang terpisah dari dataset utama. Hal ini bertujuan untuk memastikan bahwa model tidak hanya mampu mempelajari data pelatihan dengan baik, tetapi juga dapat memberikan hasil yang akurat pada data baru yang belum pernah dilihat

sebelumnya. Proses validasi ini dapat dilakukan dengan mengukur berbagai metrik, seperti akurasi deteksi objek, presisi, recall, dan F1-score.

5. Evaluasi Kinerja

   Setelah melalui proses validasi, langkah terakhir dalam pengembangan model adalah pengujian kinerja pada subset data baru yang terpisah dari dataset pelatihan. Tujuannya adalah untuk mengetahui kemampuan model dalam mengenali objek-objek baru yang belum pernah dilihat sebelumnya.Evaluasi kinerja ini dapat mencakup berbagai metrik, seperti akurasi deteksi objek, waktu inferensi, dan metrik relevan lainnya. Hasil evaluasi ini kemudian digunakan untuk menilai performa model secara keseluruhan.

   Jika performa model belum sesuai dengan target, proses pengujian ini dapat diulang dengan melakukan penyesuaian pada model atau parameternya. Penyesuaian ini dapat berupa memodifikasi arsitektur model, mengubah algoritma pelatihan, atau menambahkan data pelatihan baru.Proses pengujian empiris yang sistematis dan terukur ini sangat penting untuk memastikan bahwa model deteksi objek yang dikembangkan memiliki performa yang optimal dan dapat diandalkan dalam aplikasi dunia nyata.

F. Teknik Analisis Data

Analisis data adalah langkah-langkah yang melibatkan pengumpulan, pengelompokan, dan seleksi data secara terstruktur dari berbagai sumber seperti wawancara, catatan lapangan, dan dokumentasi. Proses ini mencakup pengorganisasian data ke dalam kategori yang relevan, merangkumnya menjadi unit-unit yang dapat dipahami, melakukan sintesis informasi, mengidentifikasi pola yang muncul, menentukan data yang krusial untuk dianalisis lebih lanjut, dan membuat kesimpulan yang mudah dimengerti. Terdapat tiga teknik utama yang digunakan dalam analisis data:

1. Reduksi Data (*Data Reduction*)

Mereduksi data adalah proses mengurangi jumlah data yang relevan atau signifikan tanpa kehilangan informasi yang penting. Tujuan dari reduksi data adalah untuk membuat dataset yang lebih kecil tetapi masih mempertahankan representasi yang akurat dari data asli. Reduksi data membantu dalam mempercepat analisis data, mengurangi beban komputasi, dan meningkatkan efisiensi penyimpanan dan pengambilan data. Ini menjadi penting terutama ketika bekerja dengan dataset yang besar atau kompleks.

2. Penyajian Data (*Data Display*)

Penyajian data adalah suatu proses yang terstruktur dalam mengatur informasi dengan sistematis untuk memfasilitasi analisis mendalam serta mengambil kesimpulan yang tepat. Dalam proses ini, informasi disusun secara terstruktur dan didesain dengan cermat untuk memudahkan pemahaman dan interpretasi. Penyajian data tidak hanya sekadar menampilkan informasi, tetapi juga mempertimbangkan bagaimana cara terbaik untuk menyampaikan pesan secara efektif kepada pemirsa atau pengguna. Dengan penyajian data yang baik, informasi dapat disampaikan dengan jelas dan mudah dipahami, memungkinkan untuk pengambilan keputusan yang lebih baik dan tindakan yang tepat.

3. Penarikan Kesimpulan (*Conclusing Drawing Verivication*)

Penarikan kesimpulan yang diambil merupakan hasil awal dari analisis dan dapat mengalami transformasi seiring dengan penemuan bukti baru dalam tahap pengumpulan data yang berikutnya. Proses ini mencakup upaya untuk menemukan atau mengurai makna, pola, dan penjelasan yang tersembunyi dalam dataset. Penarikan kesimpulan menandai tahap penutup dalam analisis data, di mana informasi yang dikumpulkan dianalisis secara komprehensif untuk mendapatkan pemahaman yang lebih dalam.

BAB VI

HASIL DAN PEMBAHASAN

A. Pembuatan Model

1. Pengambilan Dataset

Pengumpulan dataset dilakukan dengan mengambil gambar menggunakan kamera HP serta mengunduh dari internet untuk memperkaya dataset. Dataset ini terdiri dari gambar objek yang umum ditemui dalam kehidupan sehari-hari dan relevan bagi penyandang tunanetra, seperti: orang, mobil, motor, sepeda, tongkat, tangga, lubang, tiang, guiding block garis, dan guiding block titik. Total dataset yang dikumpulkan berjumlah 2040 gambar.

Tabel 1. Jumlah Dataset

Nama Kelas	Jumlah Data
Orang	220
Mobil	220
Motor	220
Sepeda	220
Tongkat	220
Tangga	220
Lubang	220
Tiang	220
Guiding block garis, dan Guiding block titik.	280
Total	2040

a. Gambar Orang

Gambar 6. Contoh Gambar Orang

b. Gambar Mobil

Gambar 7. Contoh Gambar Mobil

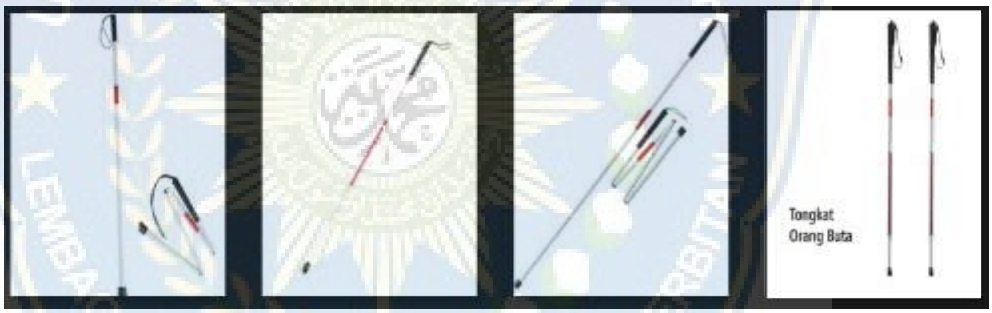
c. Gambar Motor

Gambar 8. Contoh Gambar Motor

d. Gambar Sepeda

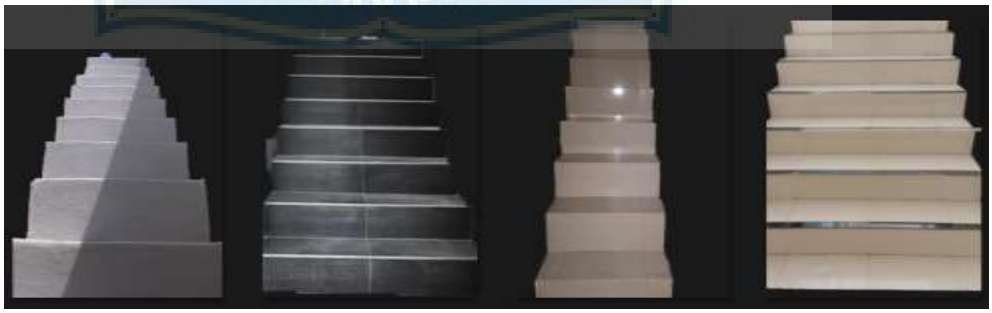
Gambar 9. Contoh Gambar Motor

e. Gambar Tongkat



Gambar 10. Contoh Gambar Tongkat

f. Gambar Tangga



Gambar 11. Contoh Gambar Tangga

g. Gambar Lubang

Gambar 12. Contoh Gambar Lubang

h. Gambar Tiang

Gambar 13. Contoh Gambar Tiang

i. Gambar Guiding block garis, dan Guiding block titik.

Gambar 14. Contoh Gambar Guiding block garis, dan Guiding block titik.

2. Pelabelan

a. Upload Dataset

. Gambar-gambar yang telah dikumpulkan kemudian dimasukkan ke dalam *platform* Roboflow sebelum dilakukan proses pelabelan. Roboflow digunakan untuk mengelola dan mempersiapkan dataset gambar, memastikan setiap gambar terorganisir dengan baik.

Gambar 15. Upload Data Pada Roboflow

b. Pembuatan *Class* Dataset

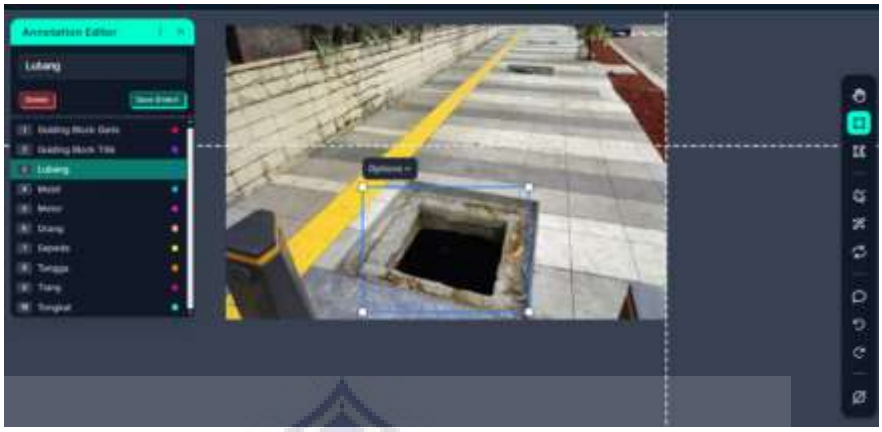
Tujuan dari pembuatan kelas ini adalah untuk menyederhanakan proses penentuan dataset gambar yang telah diberi *bouding box*. Kelas ini dirancang agar secara otomatis dapat mengelompokkan gambar ke dalam 10 kategori yang telah ditentukan sebelumnya, yaitu orang, mobil, motor, sepeda, tongkat, tangga, lubang, tiang, guiding block garis, dan guiding block titik. Dengan adanya kelas ini, proses penentuan dan pengelompokan dataset

menjadi lebih efisien dan terstruktur. Setiap gambar dalam dataset akan dianalisis dan diberi label sesuai dengan kategori yang paling tepat.

Gambar 16. Pembutan *Class* Dataset Pada Roboflow

c. Proses Pelabelan

Proses pelabelan gambar dilakukan menggunakan *platform* Roboflow dengan cara menambahkan *bouding box* atau bingkai di sekitar objek yang terdapat dalam gambar. Langkah pertama dalam proses ini adalah membuat *bouding box* secara manual, yaitu dengan menggambar kotak di sekitar setiap objek yang ingin diidentifikasi. Setelah *bouding box* dibuat, sistem Roboflow secara otomatis menampilkan kelas atau kategori objek yang telah ditentukan sebelumnya. Kategori-kategori ini telah diatur sebelumnya sesuai dengan jenis objek yang terdapat dalam dataset, sehingga setiap *bouding box* yang dibuat langsung diberi label sesuai dengan kategori yang relevan. Proses ini memastikan bahwa setiap objek dalam gambar diberi label yang akurat dan konsisten, memudahkan dalam tahap-tahap pengolahan data selanjutnya.

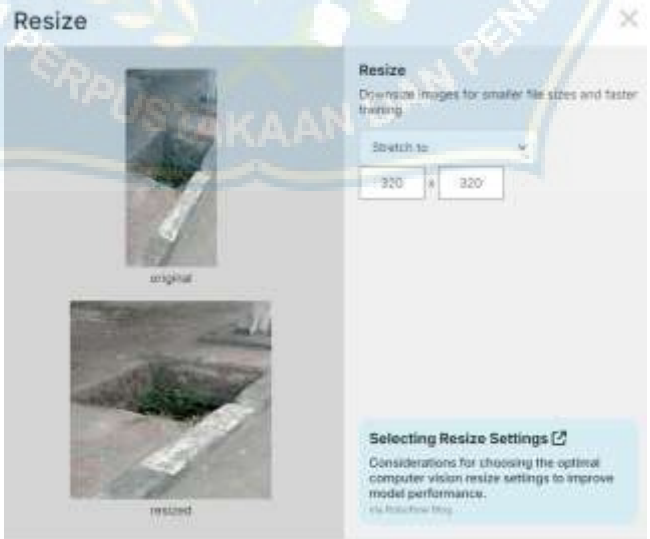


Gambar 17. Proses Pelebelan Dataset Pada Roboflow

3. Preprocessing Gambar

a. Pengolahan Gambar

Pada langkah pengolahan gambar dengan Roboflow, berbagai modifikasi dilakukan, seperti perubahan warna, bentuk, dan ukuran. Selain itu, teknik seperti *preprocessing* dan augmentasi juga digunakan. Resizing, metode yang digunakan, mengubah ukuran gambar menjadi 320 x 320 piksel. Tujuan dari resizing ini adalah untuk mempercepat keseluruhan proses pelatihan model dengan mengurangi beban kerja GPU.



Gambar 18. Proses Pengolahan Gambar (Resize)

Gambar 19. Proses Pengolahan Gambar (Flip)

Dalam pengolahan gambar, augmentasi adalah tahap di mana pola, posisi, dan elemen lain dari gambar aslinya diubah atau diubah. Tujuan utamanya adalah untuk meningkatkan jumlah data yang tersedia untuk pelatihan dan mengajarkan mesin untuk mengenali berbagai pola gambar. Peneliti menggunakan pembalikan gambar (*flip*) vertikal dan horizontal dalam augmentasi *bouding box*, yang menghasilkan variasi tambahan pada gambar aslinya untuk membantu model belajar mengenali objek.

b. Pembagian Data

Dalam penelitian ini, proses pengelompokan dataset dilakukan menggunakan alat Roboflow, yang bertujuan untuk membagi dataset menjadi tiga bagian: *training*, validasi, dan pengujian. Sebelum menerapkan augmentasi, dataset yang berjumlah total 2040 gambar akan dibagi dengan proporsi tertentu. Sebanyak 70% dari gambar-gambar ini akan digunakan untuk data pelatihan,

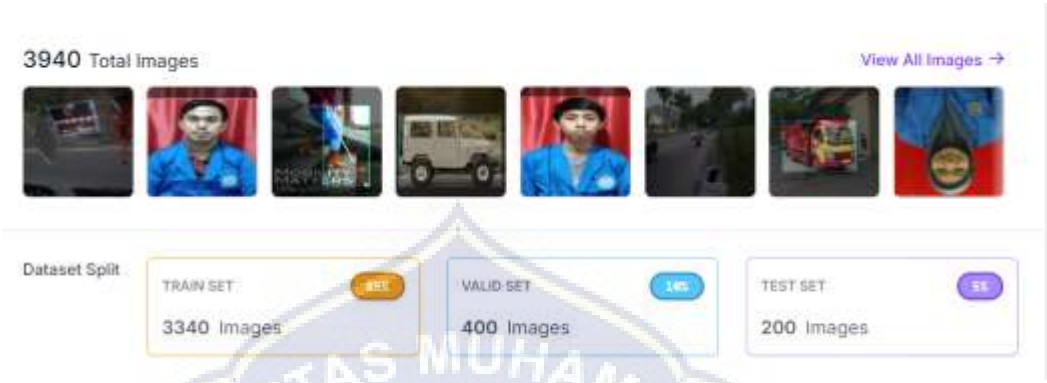
yaitu sebanyak 1440 gambar. Kemudian, 20% akan dialokasikan untuk data validasi, yang berjumlah 400 gambar. Sisanya, yaitu 10% atau 200 gambar, akan digunakan sebagai data pengujian. Proses ini memastikan bahwa model yang dikembangkan memiliki data yang cukup untuk dilatih, divalidasi, dan diuji secara menyeluruh.

Gambar 20. Proses Split Dataset Sebelum Augmentasi

Setelah melakukan augmentasi menggunakan *bouding box* dan menerapkan *flip* secara horizontal dan vertikal, data awal dimanipulasi untuk memperkaya dataset pelatihan. Augmentasi ini bertujuan untuk meningkatkan variasi dan jumlah data yang tersedia bagi model untuk dilatih. Setelah proses augmentasi selesai, dataset yang semula terdiri dari 2040 gambar bertambah menjadi 2514 gambar. Kemudian, dataset ini akan dibagi lagi dengan proporsi yang lebih besar untuk pelatihan: 85% dari total dataset akan digunakan sebagai data pelatihan, yang berjumlah 3354 gambar. Selain itu, 10% dari dataset akan digunakan untuk validasi, berjumlah 400 gambar, dan sisanya 5% atau 200 gambar akan digunakan sebagai data pengujian. Pembagian ini memastikan bahwa model memiliki data yang lebih banyak dan beragam untuk proses

pelatihan, validasi, dan pengujian, sehingga dapat meningkatkan kinerja dan akurasi model secara keseluruhan.

Gambar 21. Split Dataset Setelah Augmentasi



4. *Training*

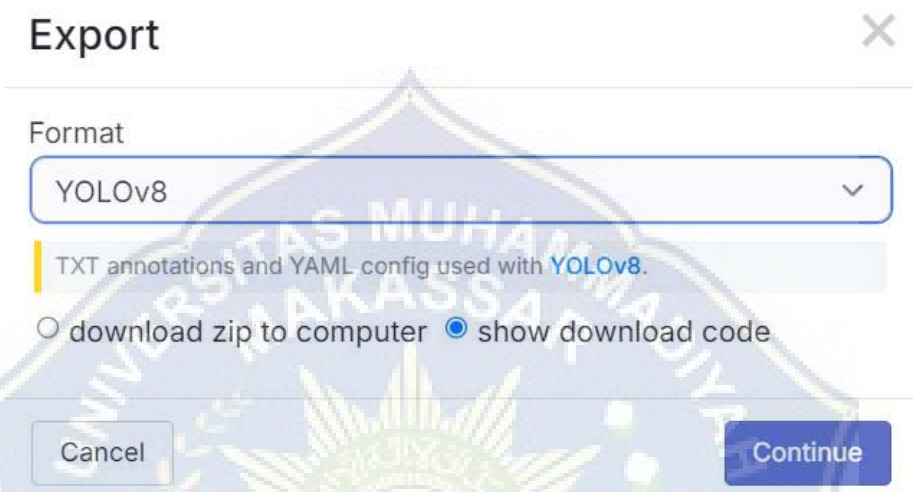
Data yang telah diolah sesuai dengan persyaratan spesifik akan diekspor untuk digunakan dalam proses pelatihan di Google Colab. Proses ekspor ini menghasilkan dataset dalam bentuk yang siap digunakan untuk pengembangan dan pelatihan model YOLOv8. Dengan dataset yang sudah dioptimalkan, API yang dihasilkan akan memungkinkan integrasi yang mulus dengan Google Colab, sehingga dapat dimanfaatkan untuk melatih model YOLOv8 secara efektif. Hasil akhir dari ekspor ini adalah menyediakan API yang berfungsi sebagai jembatan untuk memfasilitasi pelatihan model YOLOv8 di lingkungan Google Colab, memastikan bahwa data yang digunakan sudah sesuai dan siap untuk memaksimalkan kinerja model dalam mendeteksi objek.

a. *Export* Data Roboflow

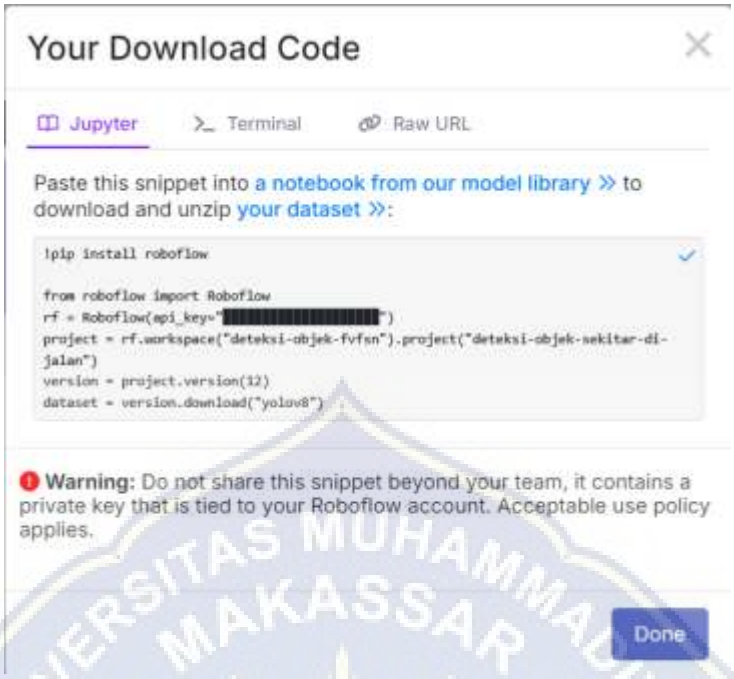
Setelah proses anotasi pada dataset selesai, langkah selanjutnya adalah mengekspor dataset tersebut ke dalam format yang sesuai dengan YOLOv8. Proses ekspor ini memastikan bahwa semua data yang telah dianotasi dapat digunakan dengan baik oleh YOLOv8 untuk tujuan pelatihan model.

Setelah dataset diekspor, Roboflow akan menghasilkan API khusus yang berisi dataset yang telah dianotasi tersebut. API ini dirancang untuk memudahkan akses dan penggunaan dataset dalam proses pelatihan model. Kemudian, API yang

disediakan oleh Roboflow ini akan digunakan di Google Colab untuk melatih model *machine learning* menggunakan YOLOv8. Dengan menggunakan Google Colab, kita dapat memanfaatkan sumber daya komputasi yang kuat dan efisien untuk menjalankan proses pelatihan model secara optimal.



Gambar 22. Proses *Export* Dataset Menjadi API



Gambar 23. Hasil Export Dataset Menjadi API

b. *Training* Googlecolab

Pelatihan model dilakukan dengan menggunakan API yang disediakan oleh Roboflow. API ini memungkinkan akses ke berbagai dataset dan alat yang dapat digunakan untuk pelatihan model. Proses pelatihan ini akan dilakukan di Google Colab, *platform* yang mendukung eksekusi kode Python di cloud. Dengan mengakses API dari Roboflow, kita dapat mengimpor dataset yang diperlukan dan menggunakan berbagai fungsi dan fitur yang disediakan untuk melatih model *machine learning* secara efektif. Pemanfaatan Google Colab juga memberikan keuntungan tambahan berupa akses ke sumber daya komputasi yang kuat dan alat kolaborasi yang memudahkan proses pengembangan dan evaluasi model.

```
import os
HOME = os.getcwd()
print(HOME)
# Pip install method (recommended)

!pip install ultralytics==8.0.196
```

```
from IPython import display
display.clear_output()

import ultralytics
ultralytics.checks()

from ultralytics import YOLO

from IPython.display import display, Image
```

Program diatas pengaturan direktori kerja, instalasi paket, pembersihan output, pengecekan, dan impor modul. Bagian pertama mengimpor modul `os` dan mendapatkan direktori kerja saat ini dengan menggunakan `os.getcwd()`, kemudian mencetak direktori tersebut. Ini memungkinkan Anda untuk mengetahui lokasi kerja di mana skrip ini dijalankan. Mengetahui direktori kerja saat ini dapat membantu dalam mengelola file dan direktori selama sesi Colab.

Bagian kedua dari kode ini menginstal paket `ultralytics` versi 8.0.196 menggunakan perintah `pip`. Perintah !`pip install ultralytics==8.0.196` dijalankan di dalam sel Colab, yang mendukung eksekusi perintah shell dengan awalan tanda seru (!). Paket `ultralytics` digunakan untuk menjalankan model YOLO, yang merupakan model deteksi objek populer. Setelah instalasi selesai, output dihapus menggunakan `display.clear_output()` dari modul IPython untuk menjaga kebersihan tampilan notebook.

Bagian ketiga dari kode ini melibatkan pengecekan dan impor modul yang diperlukan untuk menggunakan YOLO. Pertama, modul `ultralytics` diimpor dan menjalankan `ultralytics.checks()` untuk memastikan bahwa lingkungan dan dependensi sudah diatur dengan benar. Selanjutnya, modul `YOLO` diimpor dari `ultralytics`, yang menyediakan fungsionalitas untuk bekerja dengan model YOLO. Modul Image dari `IPython.display` juga diimpor untuk menampilkan gambar dalam notebook. Kombinasi dari modul-modul ini memungkinkan Anda untuk menjalankan deteksi objek menggunakan model YOLO dan menampilkan hasilnya langsung di notebook Google Colab.

```
!mkdir {HOME}/datasets
%cd {HOME}/datasets
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="Iqowf7szQMvCdAH8ibaX")
project = rf.workspace("deteksi-objek-
fvfsn").project("deteksi-objek-sekitar-di-jalan")
version = project.version(7)
dataset = version.download("yolov8")
```

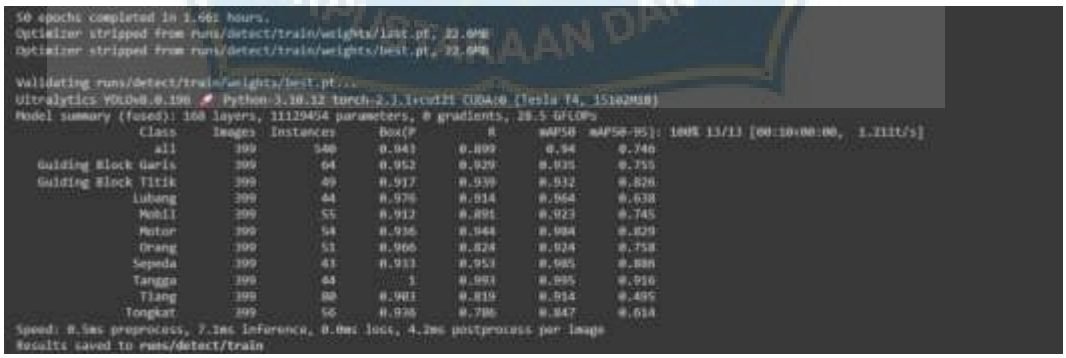
Program tersebut bertujuan untuk mempersiapkan direktori dan dataset untuk proyek deteksi objek menggunakan YOLOv8. Pertama, direktori baru bernama `datasets` dibuat di dalam direktori kerja saat ini dengan menggunakan perintah shell `mkdir`, dan kemudian direktori kerja diubah ke dalam direktori tersebut menggunakan magic command `%cd`. Langkah selanjutnya adalah menginstal paket `roboflow` melalui `pip`, yang memungkinkan penggunaan API Roboflow untuk mengelola dan mengunduh dataset. Setelah instalasi, kelas `Roboflow` diimpor, dan instance `Roboflow` dibuat dengan API key yang disediakan untuk mengautentikasi pengguna. Proyek spesifik dari workspace Roboflow diakses dengan menggunakan nama workspace `deteksi-objek-fvfsn` dan nama proyek `deteksi-objek-sekitar-di-jalan`. Versi proyek yang dipilih adalah versi ke-7, dan dataset yang sesuai diunduh dalam format yang kompatibel dengan YOLOv8 menggunakan metode `download`. Dataset ini kemudian disimpan dalam direktori `datasets`, mempersiapkan semua yang diperlukan untuk melanjutkan ke tahap pelatihan model atau inferensi dalam proyek deteksi objek di Google Colab.

API Roboflow akan dimanfaatkan untuk mengakses dataset yang diperlukan, dan setelah akses tersebut berhasil dilakukan, API ini akan digunakan untuk melatih model deteksi objek.

```
%%cd {HOME}
```

```
!yolo task=detect mode=train model=yolov8s.pt
data=/content/datasets/datasets/Deteksi-Objek-Sekitar-
Di-Jalan-7/data.yaml epochs=50 imgsz=800 plots=True
```

Program tersebut bertujuan untuk mengubah direktori kerja saat ini dan melatih model YOLO untuk tugas deteksi objek. Pertama, direktori kerja diubah ke direktori yang disimpan dalam variabel `HOME` menggunakan perintah `%cd {HOME}`, yang memastikan bahwa semua file dan folder yang diperlukan berada di jalur yang benar, memungkinkan akses yang tepat selama pelatihan model. Kemudian, pelatihan model YOLO dimulai dengan menjalankan perintah `!yolo task=detect mode=train model=yolov8s.pt data=/content/datasets/datasets/Deteksi-Objek-Sekitar-Di-Jalan-7/data.yaml epochs=50 imgsz=800 plots=True`. Perintah ini mengatur berbagai parameter penting: `task=detect` menetapkan tugas deteksi objek, `mode=train` menetapkan mode pelatihan, dan `model=yolov8s.pt` menunjukkan model YOLO pralatih versi 8 yang akan digunakan. Dataset untuk pelatihan ditentukan oleh path dalam `data`, sementara jumlah epoch pelatihan ditetapkan ke 50 untuk memastikan model dilatih secara menyeluruh. Ukuran gambar yang digunakan selama pelatihan diatur ke 800x800 piksel dengan `imgsz=800`, dan `plots=True` memastikan bahwa hasil pelatihan, termasuk grafik kerugian dan metrik evaluasi, akan ditampilkan. Dengan konfigurasi ini, model YOLO akan dilatih untuk mendeteksi objek dalam dataset yang ditentukan, meningkatkan akurasi deteksi melalui proses pelatihan yang komprehensif.



Gambar 24. Hasil Training

Setelah melakukan percobaan pelatihan di Google Colab dengan menggunakan 70% data untuk pelatihan, 20% data untuk validasi, dan 10% data untuk pengujian,

kami berhasil memperoleh akurasi sebesar 94%. Ketika proses pelatihan ini selesai, akan ada berkas yang dihasilkan dengan nama `best.pt`. Berkas `best.pt` ini akan digunakan untuk menguji model pada video yang belum pernah dilihat oleh model sebelumnya, sehingga dapat mengevaluasi kinerja model dalam mendeteksi objek pada data baru dan memastikan bahwa model bekerja dengan baik di luar dataset pelatihan.

c. *Validating*

Setelah latihan selesai, model yang dibuat akan divalidasi dengan menggunakan data yang belum pernah dilihat sebelumnya. Proses verifikasi ini melibatkan pengujian model pada berbagai dataset untuk memastikan bahwa model dapat mengenali objek dengan sangat akurat pada data baru. Hasil dari verifikasi ini sangat penting untuk mengevaluasi kinerja model dalam situasi dunia nyata dan untuk menentukan apakah model telah siap digunakan atau memerlukan penyesuaian lebih lanjut.

```
%cd {HOME}

!yolo task=detect mode=val
model={HOME}/runs/detect/train/weights/best.pt
data=/content/datasets/datasets/Deteksi-Objek-Sekitar-
Di-Jalan-7/data.yaml
```

Program ini bertujuan untuk memvalidasi model deteksi objek YOLO yang telah dilatih. Pertama, direktori kerja diubah ke direktori yang disimpan dalam variabel `HOME` dengan perintah `%cd {HOME}` untuk memastikan operasi file berikutnya dilakukan relatif terhadap direktori ini.

Kemudian, perintah `!yolo task=detect mode=val model={HOME}/runs/detect/train/weights/best.pt data=/content/datasets/datasets/Deteksi-Objek-Sekitar-Di-Jalan-7/data.yaml` dijalankan. Perintah ini menjalankan validasi model YOLO dengan model yang

telah dilatih (terletak di `{HOME}/runs/detect/train/weights/best.pt`) dan menggunakan data konfigurasi yang ditentukan dalam file YAML (berisi informasi tentang dataset validasi).

Langkah ini memastikan bahwa model yang telah dilatih dievaluasi pada dataset validasi untuk mengukur kinerjanya dan memastikan akurasi dalam mendeteksi objek pada data yang belum pernah dilihat sebelumnya.

```
Ultralytics YOLOv8.0.196 🚀 Python-3.10.12 torch-2.3.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 11129454 parameters, 0 gradients, 28.5 GFLOPs
val: Scanning /content/datasets/datasets/Deteksi-Objek-Sekitar-Di-Jalan-7/valid/labels.cache..
WARNING ⚠️ Box and segment counts should be equal, but got len(segments) = 126, len(boxes) = 5
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%
                   all        399        540      0.943      0.899      0.941      0.747
   Guiding Block Garis        399         64      0.952       0.93      0.936      0.755
   Guiding Block Titik        399         49      0.917      0.939      0.932      0.826
                Lubang        399         44      0.976      0.914      0.964       0.64
                 Mobil        399         55      0.912      0.891      0.923      0.747
                 Motor        399         54      0.936      0.944      0.984      0.831
                 Orang        399         51      0.965      0.824      0.924      0.756
                Sepeda        399         43      0.933      0.953      0.985      0.887
                Tangga        399         44          1      0.993      0.995      0.913
                 Tiang        399         80      0.904       0.82      0.914      0.495
               Tongkat        399         56      0.935      0.786      0.851      0.616
Speed: 1.4ms preprocess, 13.6ms inference, 0.0ms loss, 3.4ms postprocess per image
```

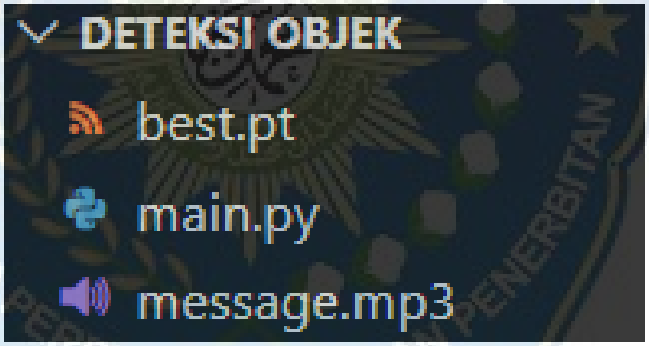
Gambar 25. Hasil Validasi

Evaluasi terhadap model YOLOv8 pada set validasi yang terdiri dari 399 gambar menunjukkan kinerja yang cukup memuaskan. Model ini mencapai precision sebesar 0.943 dan recall sebesar 0.899, yang menggambarkan ketepatan dan kemampuan model dalam mendeteksi objek dengan benar. Selain itu, model ini juga menghasilkan nilai mAP50 sebesar 0.941 dan mAP50-95 sebesar 0.747, menunjukkan bahwa model dapat melakukan deteksi dengan baik pada berbagai ukuran objek. Rata-rata waktu pemrosesan per gambar adalah 13,6 ms selama inferensi, menandakan bahwa model ini mampu melakukan deteksi secara *real-time* dengan kecepatan yang cukup tinggi dan andal.

B. Pengujian sistem

1. Pengujian Sistem Deteksi Objek Secara *Real-Time*

Memasukkan model yang telah dilatih sebelumnya ke dalam pengujian deteksi objek secara *real-time* adalah langkah yang sangat penting dalam tahap pengujian. Selama fase pelatihan, model bernama "best.pt" akan dimuat ke dalam script Python menggunakan YOLOv8. *Script* ini akan mengambil video langsung dari kamera, memproses setiap frame yang diterima, dan menggunakan model tersebut untuk melakukan inferensi. Setelah objek yang terdeteksi digambarkan dalam kotak pembatas dan diberi label padanya, hasil deteksi akan ditampilkan. Langkah ini memastikan bahwa sistem deteksi objek secara *real-time* menggunakan model yang paling optimal untuk memberikan hasil yang akurat dan andal ketika diuji dengan data langsung. Ini sangat penting untuk memastikan bahwa model beroperasi dengan baik dalam situasi nyata dan dapat memberikan deteksi yang cepat dan tepat pada berbagai objek dalam kondisi *real time*



Gambar 26. Testing Model

Ini adalah *script* Python yang digunakan untuk mendeteksi model:

```
from ultralytics import YOLO
import cvzone
import cv2
import math
import pygame
import time
from gtts import gTTS
```

```
import os

# Inisialisasi pygame untuk audio
pygame.mixer.init()

# Pengaturan kamera
cap = cv2.VideoCapture(1)
model = YOLO('./best.pt')

# Daftar class yang diinginkan
classnames = [
    'Guiding Block Garis', 'Guiding Block Titik', 'Lubang',
    'Mobil', 'Motor', 'Orang', 'Sepeda', 'Tangga', 'Tiang',
'Tongkat'
]

# Parameter kamera dan objek
actual_width = 50  # Lebar sebenarnya objek (cm)
focal_length = 800  # Fokus kamera (px)

# Pengaturan waktu
time_to_speak = time.time()
time_interval = 5  # Interval waktu untuk pengeluaran suara
(detik)
boxd = []
can_speak = True

# Fungsi perhitungan jarak
def calculate_distance(focal_length, actual_width,
pixel_width):
    return (actual_width * focal_length) / pixel_width

# Main loop
while True:
    ret, frame = cap.read()
    if not ret:
        break

    frame = cv2.resize(frame, (640, 480))
    detections = model(frame, stream=True)
    current_time = time.time()

    for det in detections:
        boxes = det.boxes
        if len(boxes) > 0:
```

```
            boxd = boxes  # Simpan kotak jika ada
        else:
            boxd = []
            can_speak = True

    if current_time - time_to_speak >= time_interval:
        for box in boxd:
            confidence = math.ceil(box.conf[0] * 100)
            Class = int(box.cls[0])

            if 0 <= Class < len(classnames) and confidence >
50:
                x1, y1, x2, y2 = map(int, box.xyxy[0])
                cv2.rectangle(frame, (x1, y1), (x2, y2), (0,
0, 255), 5)

                pixel_width = x2 - x1
                distance = calculate_distance(focal_length,
actual_width, pixel_width)

                cvzone.putTextRect(frame,
f'{classnames[Class]} {confidence}% {distance:.2f} cm',
                                   [x1 + 8, y1 + 30],
scale=1.5, thickness=2)

                if can_speak:
                    speech_text = f"Di Depan Anda ada
{classnames[Class]} dengan jarak {distance:.2f} centimeter"
                    speech = gTTS(speech_text, lang="id")
                    try:
                        speech.save("message.mp3")
                        alert_sound =
pygame.mixer.Sound('message.mp3')
                        alert_sound.play()
                        can_speak = False
                        time_to_speak = current_time  #
Reset waktu pengeluaran suara
                    except Exception as e:
                        print(e)

    cv2.imshow("deteksi", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
# Clean up
cap.release()
```

```
cv2.destroyAllWindows()
pygame.quit()
```

Program ini mengimplementasikan sistem deteksi objek secara *real-time* menggunakan model YOLOv8 dan memberikan feedback audio menggunakan `gTTS` (Google Text-to-Speech) dan `pygame`. Pertama, berbagai pustaka yang diperlukan diimpor, termasuk `YOLO` dari `ultralytics` untuk deteksi objek, `cvzone` dan `cv2` untuk pengolahan gambar, `math` untuk perhitungan, `pygame` untuk audio, `time` untuk manajemen waktu, serta `gTTS` untuk mengonversi teks menjadi suara. Inisialisasi `pygame` dilakukan untuk pemutaran audio, diikuti dengan pengaturan kamera untuk menangkap video secara *real-time* dan memuat model YOLO dari file `best.pt`. Parameter kamera dan objek, seperti lebar sebenarnya objek dan panjang fokus kamera, serta daftar nama kelas objek yang ingin dideteksi juga ditentukan.

Pada kode tersebut, terdapat fitur perhitungan jarak yang digunakan untuk menentukan jarak antara kamera dan objek yang terdeteksi oleh model YOLO (You Only Look Once). Fitur ini berguna untuk memberikan informasi visual dan audio terkait jarak objek tersebut. Mari kita bedah satu per satu fitur perhitungan jaraknya dan bagaimana kode ini bekerja secara keseluruhan:

a. Pengaturan Awal

- Inisialisasi Pygame:

```
pygame.mixer.init()
```

Digunakan untuk inisialisasi modul audio Pygame agar dapat memutar file suara ketika dibutuhkan.

- Pengaturan Kamera:

```
cap = cv2.VideoCapture(1)
model = YOLO('./best.pt')
```

Kamera diakses dengan cv2.VideoCapture(1), dan model deteksi objek YOLO yang sudah dilatih dimuat dari file best.pt.

- Parameter Kamera dan Objek:

```
actual_width = 50  # Lebar sebenarnya objek (cm)
```

```
focal_length = 800  # Fokus kamera (px)
```

Ini adalah dua parameter penting untuk menghitung jarak. actual_width adalah lebar sebenarnya dari objek dalam satuan centimeter (cm), sementara focal_length adalah panjang fokus kamera dalam satuan piksel (px). Panjang fokus biasanya diperoleh dari kalibrasi kamera.

b. Fungsi Perhitungan Jarak

Fungsi ini bertanggung jawab untuk menghitung jarak objek yang terdeteksi dari kamera berdasarkan ukuran objek dalam piksel.

```
def calculate_distance(focal_length, actual_width, pixel_width):
return (actual_width * focal_length) / pixel_width
```

- Parameter:
  - focal_length: Panjang fokus kamera dalam satuan pixel. Ini merupakan ukuran yang menentukan seberapa jauh objek terlihat oleh kamera. Nilai ini biasanya diperoleh dari spesifikasi kamera atau hasil kalibrasi.
  - actual_width: Lebar sebenarnya dari objek dalam satuan cm. Ini adalah ukuran fisik dari objek yang diketahui. Dalam contoh ini, lebar objek yang dianggap sebagai actual_width adalah 50 cm.
  - pixel_width: Lebar objek dalam satuan pixel yang diukur dari gambar (frame) yang diambil oleh kamera. Lebar ini diperoleh dari *bounding box* (kotak batas) yang menandai objek dalam gambar
- .Cara Kerja: Fungsi ini menghitung jarak dengan menggunakan rumus:

$$distance = \frac{actual\_width \times focal\_length}{pixel\_width} \quad (1)$$

Dengan kata lain, rumus ini bekerja dengan memanfaatkan perbandingan antara lebar aktual objek (actual_width), panjang fokus kamera (focal_length), dan lebar objek yang terlihat dalam gambar (pixel_width). Ini menghasilkan jarak antara objek dan kamera dalam satuan yang sama dengan actual_width, yaitu cm.

c. Proses Deteksi dan Perhitungan Jarak dalam Main Loop

Dalam loop utama, frame dari kamera diambil dan dideteksi dengan model YOLO:

```
detections = model(frame, stream=True)
```

Deteksi ini menghasilkan kotak pembatas (bounding boxes) yang menunjukkan lokasi objek pada gambar. Jika kotak pembatas (boxes) ditemukan, kode akan menyimpan kotak-kotak tersebut dalam variabel boxd.

Setelah itu, jika interval waktu tertentu sudah tercapai (setiap 5 detik), perhitungan jarak akan dilakukan:

```
pixel_width = x2 - x1
distance = calculate_distance(focal_length, actual_width,
pixel_width)
```

Di sini, x2 - x1 adalah lebar objek yang terdeteksi pada gambar dalam piksel (pixel_width), yang kemudian digunakan dalam fungsi calculate_distance untuk menghitung jarak dalam cm.

Jarak yang dihitung dan informasi lainnya akan ditampilkan pada layar:

```
cvzone.putTextRect(frame, f'{classnames[Class]} {confidence}%
{distance:.2f} cm', [x1 + 8, y1 + 30], scale=1.5,thickness=2)
```

d. Pengeluaran Suara Berbasis Jarak

Selain menampilkan informasi pada layar, kode juga menggunakan gTTS (*Google Text-to-Speech*) untuk menghasilkan audio terkait jarak objek yang terdeteksi:

```
speech_text = f"Di Depan Anda ada {classnames[Class]} dengan
jarak {distance:.2f} centimeter"
speech = gTTS(speech_text, lang="id")
```

- speech_text akan berisi teks dengan nama objek dan jarak dalam centimeter.
- gTTS mengubah teks ini menjadi audio dalam format .mp3, yang kemudian disimpan dan diputar menggunakan Pygame:

```
alert_sound = pygame.mixer.Sound('message.mp3')
alert_sound.play()
```

e. Menampilkan dan Mengumumkan Jarak Objek

Untuk setiap objek yang terdeteksi, kode juga menggambar kotak pembatas pada objek tersebut di frame video dengan menggunakan OpenCV:

```
cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), 5)
```

- Visualisasi: Hasil perhitungan jarak akan ditampilkan pada frame, bersamaan dengan nama kelas objek dan tingkat kepercayaan deteksi dalam bentuk teks pada video yang sedang berjalan.
- Audio (Pengeluaran Suara): Jika jarak sudah dihitung dan interval waktu yang diatur sudah terpenuhi, sistem akan menghasilkan suara melalui teks ke suara (TTS) dengan menggunakan gTTS. Suara akan berisi pesan seperti: "Di Depan Anda ada [nama objek] dengan jarak [jarak] centimeter".

Code ini merupakan implementasi sistem deteksi objek menggunakan model YOLO dengan fitur pemberitahuan suara. Input video dari kamera diproses untuk mendeteksi objek seperti "Guiding Block Garis", "Guiding Block Titik", "Lubang", "Mobil", "Motor", "Orang", "Sepeda", "Tangga", "Tiang", dan "Tongkat". Untuk setiap objek yang terdeteksi, sistem menghitung jarak antara kamera dan objek menggunakan rumus perhitungan jarak berdasarkan lebar objek dalam piksel serta lebar sebenarnya. Objek yang terdeteksi kemudian ditampilkan pada video dengan kotak pembatas (*bounding box*) beserta nama objek, tingkat kepercayaan, dan jaraknya.

Selain itu, sistem dilengkapi dengan pemberitahuan audio yang dihasilkan secara dinamis menggunakan gTTS. Setiap beberapa detik, jika ada objek yang terdeteksi dengan tingkat kepercayaan di atas 50%, sistem akan mengeluarkan suara untuk memberi tahu pengguna mengenai objek yang terdeteksi serta jaraknya dalam bahasa Indonesia. Suara tersebut dihasilkan melalui modul pygame dengan file audio yang dibuat berdasarkan teks hasil deteksi. Loop utama dari program terus berjalan hingga pengguna menekan tombol 'q', yang akan menghentikan kamera dan menutup semua jendela aplikasi.

2. Pengujian *Black-Box*

Pada tahap ini, dilakukan pengujian untuk memastikan algoritma dan kode aplikasi berfungsi dengan baik tanpa hambatan. Hasil analisis pada tabel di bawah menunjukkan bahwa secara keseluruhan aplikasi beroperasi sesuai rencana. Namun, ditemukan satu skenario yang tidak valid, yaitu skenario ke-3: Menampilkan objek di depan kamera yang tidak termasuk dalam daftar classnames (seperti botol, buku). Meskipun ada kendala pada skenario tersebut, pengujian menunjukkan bahwa aplikasi berjalan sesuai harapan pada skenario lainnya.

Tabel 2. Hasil Pengujian *Black-Box*

No	Skenario Pengujian	Hasil Yang Di Harapkan	Hasil Pengujian	Kesimpulan
1	Saat program di jalankan akan muncul kamera untuk mendeteksi	Jendela kamera terbuka dengan tampilan *live* dan mulai mendeteksi.	Sesuai Harapan	*Valid*
2	Menampilkan beberapa objek yang termasuk dalam daftar classnames (misalnya, mobil, sepeda) di depan kamera.	Kode harus mendeteksi objek, menggambar kotak di sekitarnya (*bounding box*), dan menampilkan teks nama kelas serta jaraknya.	Sesuai Harapan	*Valid*
3	Menampilkan objek di depan kamera yang tidak termasuk dalam daftar classnames	Program tidak boleh memberikan deteksi atau informasi yang salah objek harus diabaikan jika tidak	Tidak Sesuai Harapan	Tidak *Valid*

	(misalnya, botol, buku).	ada deteksi yang relevan.		
4	Ketika objek terdeteksi, program memproses dan menampilkan informasinya..	Setiap objek terdeteksi akan diberi *bounding box* merah, disertai teks nama objek, tingkat kepercayaan, dan jarak dari kamera.	Sesuai Harapan	*Valid*
5	Menampilkan objek (misalnya, mobil) di depan kamera selama lebih dari 5 detik untuk memicu keluaran suara beberapa kali.	Program hanya harus mengeluarkan suara setiap 5 detik, bukan terus menerus.	Sesuai Harapan	*Valid*
6	Menampilkan objek di jarak yang jauh dari kamera misalnya, lebih dari 3 meter.	Program harus mendeteksi objek dengan tepat, menampilkan jarak, dan memproyeksikan suara saat jarak terdeteksi.	Sesuai Harapan	*Valid*
7	Menampilkan objek di	Program harus mendeteksi objek	Sesuai Harapan	*Valid*

	jarak yang dekat dari kamera misalnya, kurang dari 70 cm.	dengan benar, menampilkan jarak yang sesuai, dan memproyeksikan suara saat jarak terdeteksi.		
8	Uji beberapa objek dengan confidence yang rendah (di bawah 50%).	Objek dengan confidence di bawah 50% tidak boleh menghasilkan output suara atau teks sesuai aturan.	Sesuai Harapan	*Valid*
9	Menguji apa yang terjadi ketika tidak ada objek yang terdeteksi dalam frame.	Sistem tidak mengeluarkan notifikasi suara atau visual jika tidak ada objek.	Sesuai Harapan	*Valid*
10	Tekan tombol q untuk keluar dari program kapan saja selama deteksi berlangsung	Program harus berhenti dengan rapi, menutup jendela, menghentikan suara, dan melepaskan kamera tanpa *error*..	Sesuai Harapan	*Valid*

Hasil dari pengujian program deteksi secara *real time*:

(Orang)

Gambar 27. Hasil Pengujian Objek Orang

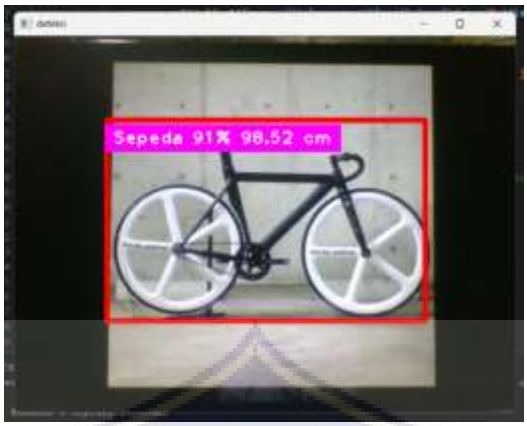
(Mobil)

Gambar 28. Hasil Pengujian Objek Mobil

(Motor)

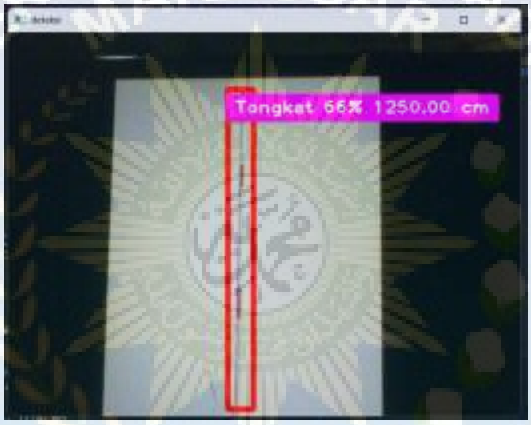
Gambar 29. Hasil Pengujian Objek Motor

(Sepeda)



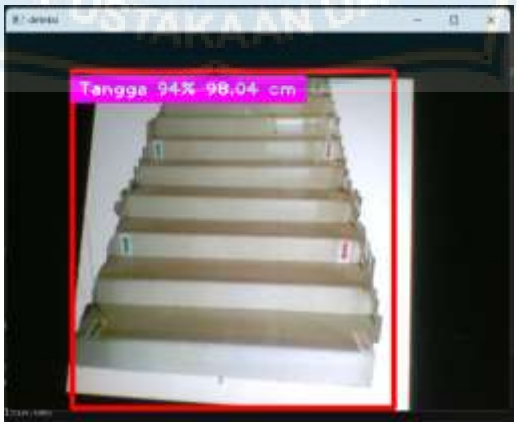
Gambar 30. Hasil Pengujian Objek Sepeda

(Tongkat)



Gambar 31. Hasil Pengujian Objek Tongkat

(Tangga)



Gambar 32. Hasil Pengujian Objek Tangga

(Lubang)

Gambar 33. Hasil Pengujian Objek Lubang

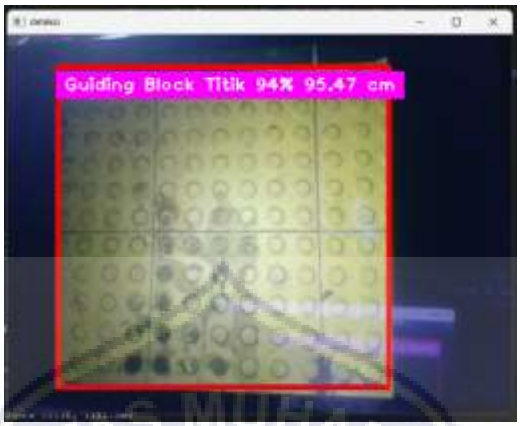
(Tiang)

Gambar 34. Hasil Pengujian Objek Tiang

(Gambar Guiding block garis)

Gambar 35. Hasil Pengujian Objek Guiding Block Garis

(Gambar Guiding block titik)



Gambar 36. Hasil Pengujian Objek Guiding Block Titik

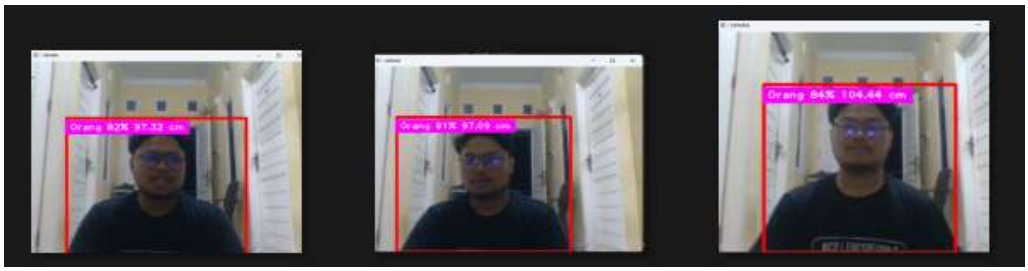
Gambar di atas adalah hasil dari pengujian deteksi objek secara *real-time* dengan menggunakan gambar yang ditampilkan pada HP menggunakan model YOLO, yang telah dilatih untuk mengenali objek yang biasa ditemui penyandang tunanetra di jalan, seperti orang, mobil, motor, sepeda, tongkat, tangga, lubang, tiang, guiding block garis, dan guiding block titik. Dalam pengujian ini, webcam digunakan untuk menangkap video secara langsung. Kemudian, model YOLO digunakan untuk mengidentifikasi dan mengklasifikasikan objek yang muncul di setiap frame video.

Hasil dari pengujian program prediksi jarak objek dengan kamera:

(Jarak 70 cm)

Gambar 37. Hasil Pengujian Jarak 70 cm

(Jarak 100 cm)

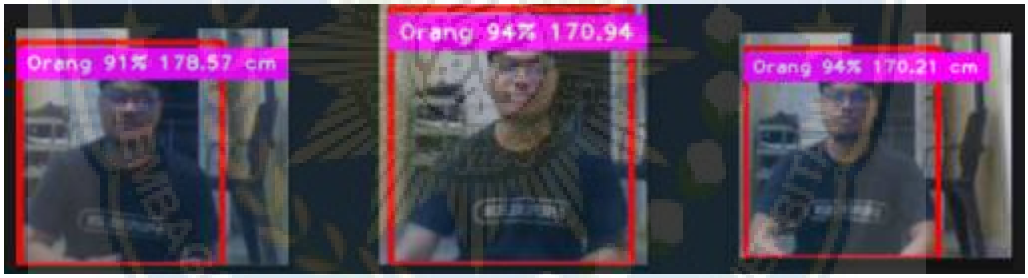


Gambar 38. Hasil Pengujian Jarak 100 cm

(Jarak 150 cm)

Gambar 39.Hasil Pengujian Jarak 150 cm

(Jarak 170 cm)



Gambar 40. Hasil Pengujian Jarak 170 cm

(Jarak 200 cm)

Gambar 41. Hasil Pengujian Jarak 200 cm

Gambar yang ditampilkan di atas merupakan hasil dari proses pengujian prediksi jarak antara objek dan kamera secara *real-time*. Pengujian ini dilakukan untuk mengukur dan memverifikasi kemampuan sistem dalam menghitung jarak dengan akurasi yang cukup baik. Dalam konteks ini, *real-time* berarti sistem mampu melakukan perhitungan jarak secara langsung dan segera memberikan hasil yang dapat diamati pada gambar tersebut.

Untuk mengetahui hasil dari akurasi dari program prediksi jarak objek dengan kamera kita harus mengetahui dulu selisih serta nilai *error* dari hasil deteksi yang yang diperoleh berikut adalah rumus mencari nilai *error* dari menggunakan rumus berikut :

$$Error(\%) = \left(\frac{Jarak\ Yanng\ Diperoleh - Jarak\ Sebenarnya}{Jarak\ Sebenarnya}\right) \times 100\% \qquad (2)$$

Berikut tabel hasil pengujian jarak antara kamera dan objek serta analisisnya:

Tabel 3. Hasil Pengujian Jarak

Jarak Sebenarnya (cm)	**Jarak yang diperoleh (cm)**	**Selisih (cm)**	**Error (%)**
70	72.99	2.99	4.27
70	74.35	4.35	6.21
70	77.22	7.22	10.31
100	97.32	-2.68	-2.68
100	97.09	-2.91	-2.91
100	104.44	4.44	4.44
150	147.60	-2.40	-1.60
150	152.09	2.09	1.39
150	153.85	3.85	2.57

170	178.57	8.57	5.04
170	170.94	0.94	0.55
170	170.21	0.21	0.12
200	202.02	2.02	1.01
200	204.08	4.08	2.04
200	200.00	0.00	0.00
Total			**30,86**

Untuk mencari nilai dari rata rata *error* kita memaukkan rumus rata-rata berikut:

$$Rata - rata\ Error(\%) = \left(\frac{\sum Error}{Jumlah\ Percobaan}\right) \quad (3)$$

$\sum$ *error* = Penjumlahan dari nilai semua nilai *error*

$$Rata - rata\ Error(\%) = \left(\frac{30{,}86}{15}\right)$$

$$Rata - rata\ Error(\%) = 2.05$$

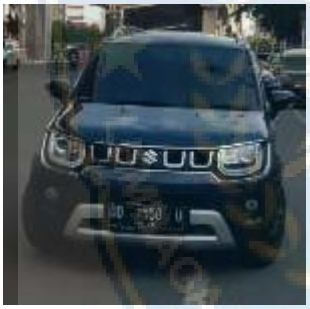
Sehinngga di peroleh rata-rata error: sebesar 2.05%, dari data ini kita dapat melihat bahwa rata-rata *error* perhitungan jarak adalah 2.05%, yang menunjukkan bahwa fitur perhitungan jarak antara kamera dan objek cukup akurat dalam berbagai kondisi jarak.

Hasil dari pengujian yang dilakukan menunjukkan bahwa model YOLO mampu mendeteksi berbagai objek yang muncul dalam gambar atau video dengan tingkat akurasi yang cukup memuaskan. Setiap objek yang terdeteksi diberi kotak pembatas merah dengan label di atasnya, mencantumkan nama objek, tingkat kepercayaan model, dan perkiraan jarak objek dari kamera. Misalnya, jika model mendeteksi 'orang' dengan keyakinan 92% dan jarak 138.41 cm, label akan

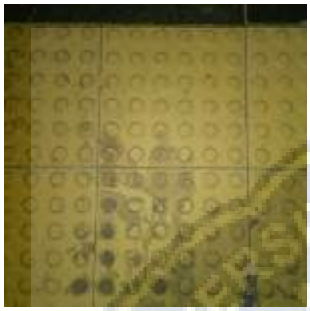
berbunyi "orang 92% 138.41 cm", memberikan informasi yang jelas dan memudahkan interpretasi hasil deteksi.

C. **Hasil Pengujian**

Tabel 4. Hasil Pengujian

Data pengujian	Kelas dataset	Hasil Deteksi	Tingkat Kepastian
	Orang		92%
	Mobil		91%
	Motor		86%
	Sepeda		91%

	Tonkat		66%
	Tangga		94%
	Lubang		90%
	Tiang		80%

	Guiding Block Garis		86%
	Guiding Block Titik		94%

Selama proses pengujian, data yang diuji terdiri dari 10 kelas gambar objek yang mencakup orang, mobil, motor, sepeda, tongkat, tangga, lubang, tiang, guiding block garis, dan guiding block titik. Setiap kelas memiliki 220 data gambar. Dalam pengujian deteksi menggunakan model YOLOv8, nilai kepastian yang diperoleh menunjukkan variasi yang cukup signifikan. Nilai kepastian tertinggi yang dicapai adalah 94%, sementara nilai kepastian terendah adalah 66%. Data dengan tingkat kepastian yang rendah ini mengindikasikan bahwa model YOLOv8 mengalami kesulitan dalam mendeteksi atau mengklasifikasikan objek dengan tingkat akurasi yang memadai. Salah satu faktor utama yang mempengaruhi rendahnya tingkat kepastian adalah keberagaman latar belakang gambar pada objek yang diuji. Latar belakang yang terlalu bervariasi dapat menyebabkan model YOLOv8 kesulitan dalam mengenali dan mengkategorikan objek dengan tepat, sehingga menurunkan akurasi deteksi secara keseluruhan. Hal ini menunjukkan bahwa untuk meningkatkan performa model, mungkin diperlukan penyesuaian atau pelatihan tambahan dengan data yang lebih homogen dalam hal latar belakang.

BAB V
PENUTUP

A. Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, penulis dapat menarik beberapa kesimpulan sebagai berikut:

1. Proses implementasi metode YOLO untuk deteksi objek di sekitar, yang meliputi objek-objek umum yang sering ditemui dalam kehidupan sehari-hari dan memiliki relevansi khusus bagi penyandang tunanetra saat berada di jalan, menunjukkan hasil yang memuaskan. Objek-objek tersebut meliputi orang, mobil, motor, sepeda, tongkat, tangga, lubang, tiang, guiding block garis, dan guiding block titik. Penelitian ini berhasil mendeteksi objek-objek tersebut dengan tingkat akurasi yang cukup baik, sehingga dapat dikatakan bahwa metode YOLO mampu memberikan dukungan yang signifikan dalam membantu penyandang tunanetra untuk lebih mengenali dan menghindari berbagai rintangan di lingkungan sekitar mereka.
2. Setelah melakukan percobaan pelatihan dengan YOLOv8 menggunakan 70% data untuk pelatihan, 20% data untuk validasi, dan 10% data untuk pengujian, diperoleh akurasi model sebesar 94%. Model ini menunjukkan kinerja yang sangat baik dalam mendeteksi objek, dengan precision sebesar 0.943, recall sebesar 0.899, mAP50 sebesar 0.941, dan mAP50-95 sebesar 0.747 pada set validasi yang terdiri dari 399 gambar. Proses inferensi dilakukan dalam rata-rata waktu 13,6 ms per gambar, menandakan kemampuan model untuk melakukan deteksi secara *real-time* dengan kecepatan tinggi. Pengujian lebih lanjut dilakukan pada 10 kelas gambar objek, masing-masing kelas terdiri dari 220 gambar, dan hasilnya menunjukkan variasi nilai kepastian dari 66% hingga 94%. Rendahnya nilai kepastian pada beberapa gambar disebabkan oleh keberagaman latar belakang yang tinggi, yang membuat model kesulitan dalam mengenali dan mengkategorikan objek dengan tepat. Untuk meningkatkan performa

model, disarankan penyesuaian atau pelatihan tambahan dengan data yang lebih homogen terkait latar belakang. Secara keseluruhan, sistem deteksi objek secara *real-time* menggunakan YOLO memberikan hasil yang akurat dan andal saat diuji. Fitur perhitungan jarak menunjukkan rata-rata *error* 2,05%, menandakan keandalannya dalam mengukur jarak antara kamera dan objek. Namun, beberapa jarak memiliki *error* lebih tinggi yang perlu diperbaiki.

B. Saran

Berdasarkan hasil penelitian, terdapat beberapa saran untuk pengembangan lebih lanjut. Pertama, meskipun metode YOLO menunjukkan akurasi yang baik dalam mendeteksi objek relevan bagi penyandang tunanetra, pelatihan tambahan dengan data yang lebih homogen terkait latar belakang dapat meningkatkan performa model dan mengurangi variasi nilai kepastian akibat keberagaman latar belakang. Kedua, untuk fitur perhitungan jarak, disarankan melakukan kalibrasi tambahan dan pengujian dalam berbagai kondisi pencahayaan dan posisi objek untuk meningkatkan keandalan pengukuran. Fitur audio yang sudah ada dapat lebih dioptimalkan untuk memberikan umpan balik jarak yang lebih akurat, membantu penyandang tunanetra memperkirakan jarak dengan lebih baik.

Selanjutnya, pengembangan tampilan sistem yang lebih intuitif dan informatif sangat penting. Antarmuka yang ramah pengguna dan fitur interaktif dapat membantu penyandang tunanetra memahami hasil deteksi dengan lebih baik. Pengujian lapangan yang lebih ekstensif dalam berbagai kondisi lingkungan nyata juga disarankan untuk memastikan keandalan sistem dalam situasi sebenarnya. Terakhir, kolaborasi dengan ahli di bidang aksesibilitas dan penyandang tunanetra akan memberikan masukan berharga untuk menyempurnakan sistem, sehingga lebih sesuai dengan kebutuhan dan kondisi pengguna. Dengan demikian, diharapkan sistem deteksi objek berbasis YOLO ini dapat memberikan kontribusi yang signifikan dalam meningkatkan mobilitas dan kemandirian penyandang tunanetra.

DAFTAR PUSTAKA

Aini, Q., Lutfiani, N., Kusumah, H., & Zahran, M. S. (2021). Deteksi dan Pengenalan Objek Dengan Model Machine Learning: Model Yolo. *CESS (Journal of Computer Engineering, System and Science)*, *6*(2), 192. https://doi.org/10.24114/cess.v6i2.25840

Astuti, F. A. (2021). *Pemanfaatan Teknologi Artificial Intelligence untuk Penguatan Kesehatan dan Pemulihan Ekonomi Nasional*. *04*(01), 25–34.

Eriana, E. S. (2023). *ARTIFICIAL INTELLIGENCE (AI)* (1st ed.). EUREKA MEDIA AKSARA, DESEMBER 2023 ANGGOTA IKAPI JAWA TENGAH NO. 225/JTE/2021. https://repository.penerbiteureka.com/publications/567027/artificial-intelligence-ai

Fauzi, Y., Andiono, E., & Khamali, M. (2020). Aplikasi *Object* Detection and Tracking Untuk Penyandang Tunanetra dengan Internet of Things (IoT) (Menggunakan Bahasa Pemrograman Phyton). *Universitas Budiluhur, Jakarta 1 Jln. Raya Cilegon Serang KM.08 Kramatwatu, 12260*.

Indaryanto, F., Nugroho, A., & Suni, A. F. (2021). Aplikasi Penghitung Jarak dan Jumlah Orang Berbasis YOLO Sebagai Protokol Kesehatan Covid-19. *Edu Komputika Journal*, *8*(1), 31–38. https://doi.org/10.15294/edukomputika.v8i1.47837

Kadiva Dwilia Rosadi Putri. (2023). *Difabel Netra dan Dunia Visual*. https://sie.telkomuniversity.ac.id/difabel-netra-dan-dunia-visual/

Liunanda, C. N., Rostianingsih, S., & Purbowo, A. N. (2020). Implementasi Algoritma YOLO pada Aplikasi Pendeteksi Senjata Tajam di Android. *Jurnal Infra*, *Vol 8*, *No.*, 1–7.

Manu, G. A., Masan, P. L., Bangsa, U. C., Nusa, P., Timur, T., & Learning, A. (2020). *APLIKASI TEXT TO SPEECH UNTUK MENINGKATKAN PEMBELAJARAN BAHASA INGGRIS BAGI SISWA DISABILITAS Gerlan*. *03*(02), 17–26.

Nur, M., Muhlashin, I., & Stefanie, A. (2023). *KLASIFIKASI PENYAKIT MATA*

BERDASARKAN CITRA FUNDUS MENGGUNAKAN YOLO V8. *7*(2), 1363–1368.

Pendidikan, T., Pendidikan, F. I., & Malang, U. N. (2020). *PENERAPAN BAHAN AJAR AUDIO UNTUK ANAK TUNANETRA TINGKAT SMP DI INDONESIA Agnes Praptaningrum seperangkat tunanetra dan noncetak . Bahan ajar cetak dapat*. *5*, 1–19.

Pradana, T., & Sula, E. H. (2023). Pengembangan Aplikasi Pendeteksi Objek Untuk Tunanetra Menggunakan Operator Tepi Canny Dengan Library Opencv Berbasis Android. *Spirit*, *15*(1), 23–27. https://doi.org/10.53567/spirit.v15i1.285

Prisky Ratna Aningtiyas, A. S. dan S. W. (2020). *Pembuatan Aplikasi Deteksi Objek Menggunakan TensorFlow Object Detection API dengan Memanfaatkan SSD MobileNet V2 Sebagai Model Pra-Terlatih*. *19*(September), 421–430.

Randy Moh Yusup, Aldof Faris Anugrah, Muslimah, D. D., Permana, S. M. W. N., & Shindi Yuliani. (2024). PENDETEKSIAN OBJEK MENGGUNAKAN OPENCV DAN METODE YOLOv4-TINY UNTUK MEMBANTU TUNANETRA. *Journal of Computer Science and Information Technology*, *1*(2), 59–68. https://doi.org/10.59407/jcsit.v1i2.532

Raup, A., Ridwan, W., Khoeriyah, Y., & Zaqiah, Q. Y. (2022). *Deep Learning dan Penerapannya dalam Pembelajaran*. *5*(September), 3258–3267.

Rosaly, R. (2019). *Pengertian Flowchart Beserta Fungsi dan Simbol-simbol Flowchart yang Paling Umum Digunakan*.

Sarmah, A. J., Bhagawati, K., Duwarah, K., & Purkayastha, S. D. (2023). *Object detection and conversion of text to speech for visually impaired*. *September*.

Tamang, S., Sen, B., Pradhan, A., Sharma, K., & Singh, V. K. (2023). *INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING Enhancing COVID-19 Safety : Exploring YOLOv8 Object Detection for Accurate Face Mask Classification*.

Yudha, Y. F., Aditya, A. A. Y., Rasyid, R. A. Y., Indra, N. I. A., & Melati, M. W. W. (2024). Perancangan Sistem Deteksi Objek Pada Robot Transporter

Menggunakan Metode Darknet YOLOv8. *Electrician : Jurnal Rekayasa Dan Teknologi Elektro*, *18*(2), 161–170. https://doi.org/10.23960/elc.v18n2.2595

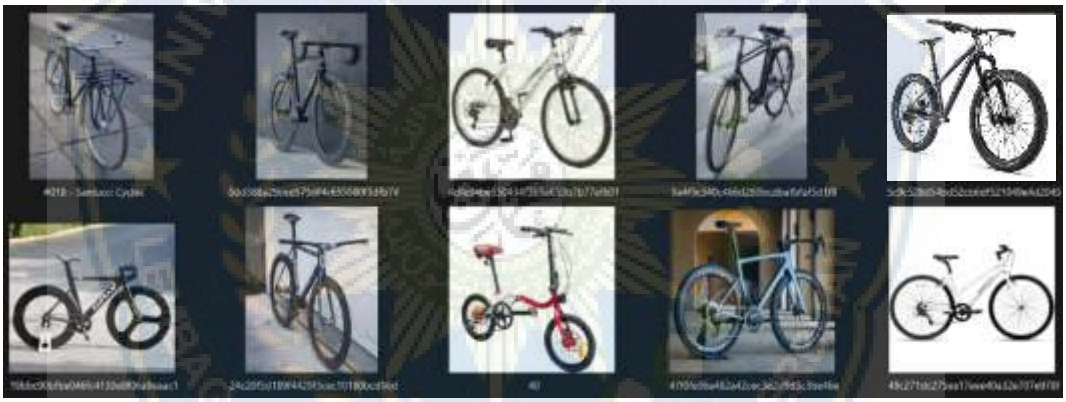
LAMPIRAN

Lampiran 1. Gambar Dataset Orang

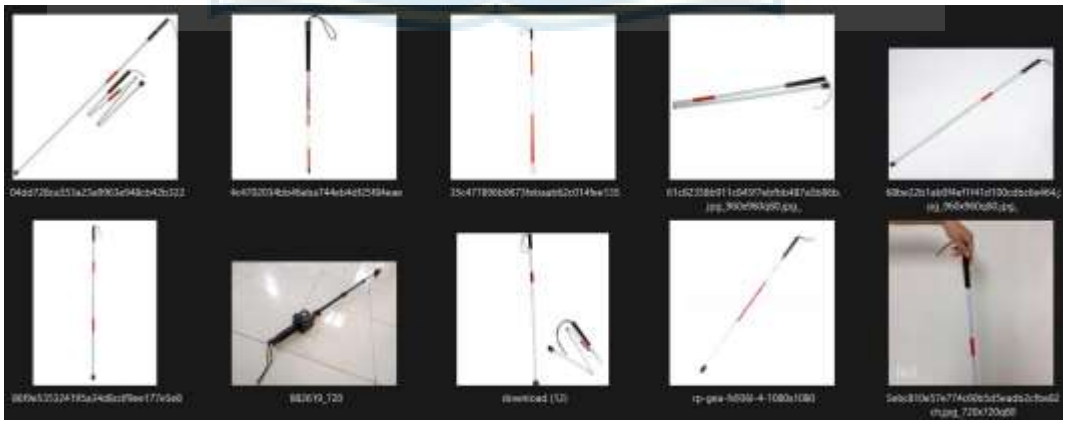
Lampiran 2. Gambar Dataset Mobil

Lampiran 3. Gambar Dataset Motor

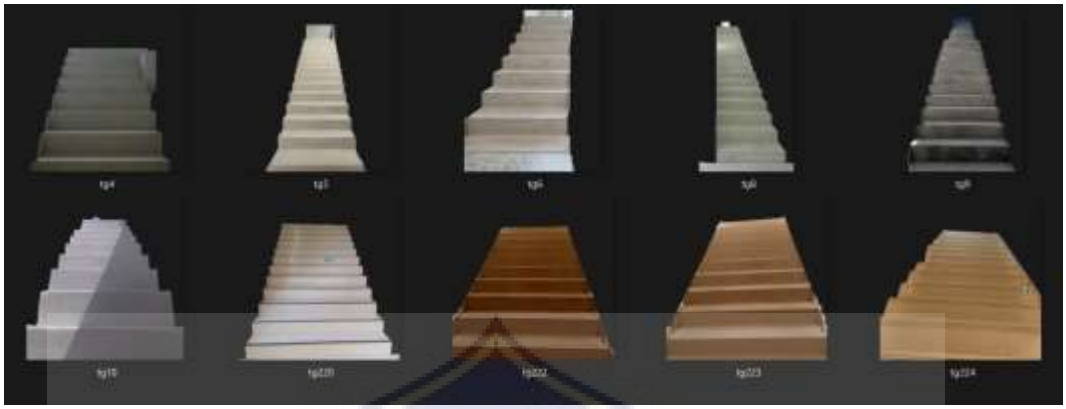
Lampiran 4. Gambar Dataset Sepeda



Lampiran 5. Gambar Dataset Tongkat



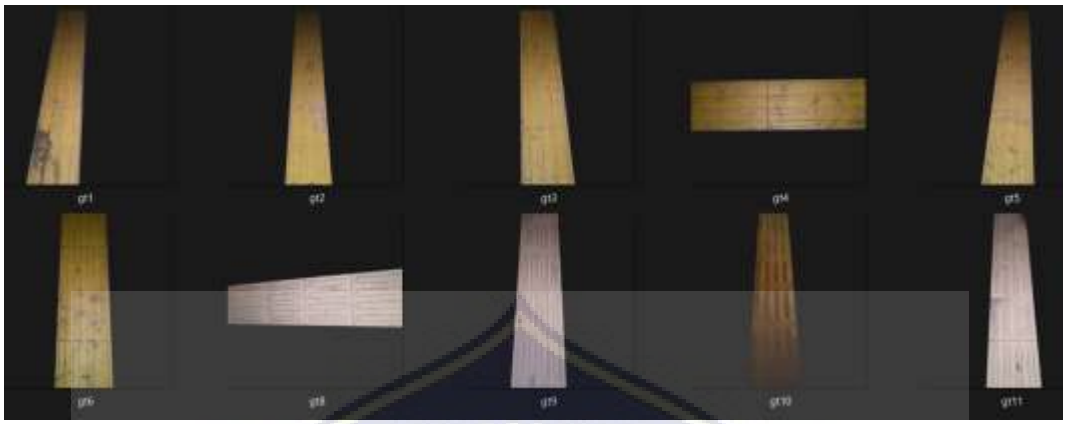
Lampiran 6. Gambar Dataset Tangga



Lampiran 7. Gambar Dataset Lubang

Lampiran 8. Gambar Dataset Tiang

Lampiran 9. Gambar Guiding Block Garis



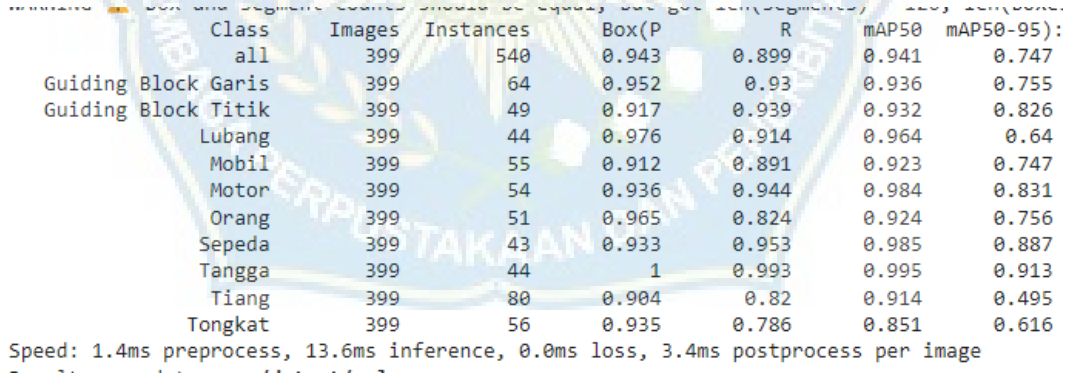
Lampiran 10. Gambar Guiding Block Titik

Lampiran 11. Source Code Training

```
import os
HOME = os.getcwd()
print(HOME)
# Pip install method (recommended)

!pip install ultralytics==8.0.196

from IPython import display
display.clear_output()

import ultralytics
ultralytics.checks()

from ultralytics import YOLO

from IPython.display import display, Image

!mkdir {HOME}/datasets
%cd {HOME}/datasets
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="Iqowf7szQMvCdAH8ibaX")
project = rf.workspace("deteksi-objek-
fvfsn").project("deteksi-objek-sekitar-di-jalan")
version = project.version(7)
dataset = version.download("yolov8")

%%cd {HOME}

!yolo task=detect mode=train model=yolov8s.pt
data=/content/datasets/datasets/Deteksi-Objek-Sekitar-
Di-Jalan-7/data.yaml epochs=50 imgsz=800 plots=True

%cd {HOME}

!yolo task=detect mode=val
model={HOME}/runs/detect/train/weights/best.pt
data=/content/datasets/datasets/Deteksi-Objek-Sekitar-
Di-Jalan-7/data.yaml
```

Lampiran 12. Proses Training

Lampiran 13. Hasil Dari Training



Lampiran 14. Source Code Pendeteksi Model

```
from ultralytics import YOLO
import cvzone
import cv2
import math
import pygame
import time
from gtts import gTTS
import os

# Inisialisasi pygame untuk audio
pygame.mixer.init()

# Pengaturan kamera
cap = cv2.VideoCapture(1)
model = YOLO('./best.pt')

# Daftar class yang diinginkan
classnames = [
    'Guiding Block Garis', 'Guiding Block Titik', 'Lubang',
    'Mobil', 'Motor', 'Orang', 'Sepeda', 'Tangga', 'Tiang',
'Tongkat'
]

# Parameter kamera dan objek
actual_width = 50  # Lebar sebenarnya objek (cm)
focal_length = 800  # Fokus kamera (px)

# Pengaturan waktu
time_to_speak = time.time()
time_interval = 5  # Interval waktu untuk pengeluaran suara
(detik)
boxd = []
can_speak = True

# Fungsi perhitungan jarak
def calculate_distance(focal_length, actual_width,
pixel_width):
    return (actual_width * focal_length) / pixel_width

# Main loop
while True:
    ret, frame = cap.read()
```

```
    if not ret:
        break

    frame = cv2.resize(frame, (640, 480))
    detections = model(frame, stream=True)
    current_time = time.time()

    for det in detections:
        boxes = det.boxes
        if len(boxes) > 0:
            boxd = boxes  # Simpan kotak jika ada
        else:
            boxd = []
            can_speak = True

    if current_time - time_to_speak >= time_interval:
        for box in boxd:
            confidence = math.ceil(box.conf[0] * 100)
            Class = int(box.cls[0])

            if 0 <= Class < len(classnames) and confidence >
50:
                x1, y1, x2, y2 = map(int, box.xyxy[0])
                cv2.rectangle(frame, (x1, y1), (x2, y2), (0,
0, 255), 5)

                pixel_width = x2 - x1
                distance = calculate_distance(focal_length,
actual_width, pixel_width)

                cvzone.putTextRect(frame,
f'{classnames[Class]} {confidence}% {distance:.2f} cm',
                                   [x1 + 8, y1 + 30],
scale=1.5, thickness=2)

                if can_speak:
                    speech_text = f"Di Depan Anda ada
{classnames[Class]} dengan jarak {distance:.2f} centimeter"
                    speech = gTTS(speech_text, lang="id")
                    try:
                        speech.save("message.mp3")
                        alert_sound =
pygame.mixer.Sound('message.mp3')
                        alert_sound.play()
                        can_speak = False
```

```
                        time_to_speak = current_time  # Reset waktu pengeluaran suara
                    except Exception as e:
                        print(e)

    cv2.imshow("deteksi", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
# Clean up
cap.release()
cv2.destroyAllWindows()
pygame.quit()
```

Lampiran 15. Hasil Deteksi Objek

MAJELIS PENDIDIKAN TINGGI PIMPINAN PUSAT MUHAMMADIYAH
UNIVERSITAS MUHAMMADIYAH MAKASSAR
UPT PERPUSTAKAAN DAN PENERBITAN
Alamat kantor: Jl.Sultan Alauddin NO.259 Makassar 90221 Tlp.(0411) 866972,881593, Fax.(0411) 865588

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

SURAT KETERANGAN BEBAS PLAGIAT

UPT Perpustakaan dan Penerbitan Universitas Muhammadiyah Makassar, Menerangkan bahwa mahasiswa yang tersebut namanya di bawah ini:

Nama : Akram

Nim : 105841111920

Program Studi : Teknik Informatika

Dengan nilai:

No	Bab	Nilai	Ambang Batas
1	Bab 1	9 %	10 %
2	Bab 2	23 %	25 %
3	Bab 3	4 %	10 %
4	Bab 4	10 %	10 %
5	Bab 5	5 %	5 %

Dinyatakan telah lulus cek plagiat yang diadakan oleh UPT- Perpustakaan dan Penerbitan Universitas Muhammadiyah Makassar Menggunakan Aplikasi Turnitin.

Demikian surat keterangan ini diberikan kepada yang bersangkutan untuk dipergunakan seperlunya.

Makassar, 21 Agustus 2024

Mengetahui,

Kepala UPT- Perpustakaan dan Penerbitan,

Nursiah S.Mum.,M.I.P

NBM. 964 591

AKRAM 105841111920 BAB I

by Tahap Tutup

Submission date: 21-Aug-2024 02:36PM (UTC+0700)
Submission ID: 2435428784
File name: BAB_I_-_2024-08-21T153549.275.docx (34.17K)
Word count: 1073
Character count: 7171

AKRAM 105841111920 BAB I

ORIGINALITY REPORT

9%	9%	0%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	repository.unisma.ac.id Internet Source	3%
2	repository.radenintan.ac.id Internet Source	2%
3	repository.upnvj.ac.id Internet Source	2%
4	core.ac.uk Internet Source	2%
5	repo.stie-pembangunan.ac.id Internet Source	2%

Exclude quotes On
Exclude bibliography On
Exclude matches < 2%

AKRAM 105841111920 BAB II

by Tahap Tutup

AKRAM 105841111920 BAB II

ORIGINALITY REPORT

23%	20%	8%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	ejournal.jak-stik.ac.id Internet Source	4%
2	journal.ppmi.web.id Internet Source	2%
3	doaj.org Internet Source	1%
4	Submitted to Universitas Negeri Semarang - iTh Student Paper	1%
5	www.jurnal.stmik-yadika.ac.id Internet Source	1%
6	Yudha Febrian Yudha, Aditya Ari Yudha Aditya, Rasyid Ammary Yahya Rasyid, Naufal Indra Ardhana Indra et al. "Perancangan Sistem Deteksi Objek Pada Robot Transporter Menggunakan Metode Darknet YOLOv8", Electrician : Jurnal Rekayasa dan Teknologi Elektro, 2024 Publication	1%

7	Robby Kamil Siregar, Anton Anton, Widiastuti Widiastuti. "Perancangan Aplikasi Bahasa Isyarat "Isyaratku" Dengan Deep Learning Serta Google Cloud Platform", Simpatik: Jurnal Sistem Informasi dan Informatika, 2021 Publication	1 %
8	jurnal.unimed.ac.id Internet Source	1 %
9	repo.itera.ac.id Internet Source	1 %
10	Submitted to STT PLN Student Paper	1 %
11	fendikcancer.wordpress.com Internet Source	1 %
12	Submitted to Universitas Muhammadiyah Makassar Student Paper	<1 %
13	repository.itk.ac.id Internet Source	<1 %
14	aau.e-journal.id Internet Source	<1 %
15	www.kompasiana.com Internet Source	<1 %
16	www.coursehero.com Internet Source	<1 %

17	apaitupengertian.com Internet Source	<1 %
18	jim.unisma.ac.id Internet Source	<1 %
19	eprints.walisongo.ac.id Internet Source	<1 %
20	geograf.id Internet Source	<1 %
21	repository.ump.ac.id Internet Source	<1 %
22	www.edukasimu.org Internet Source	<1 %
23	Ary Suryadi, Wahid Andika Syb'an, Nazzala Alfa'inna, Eni Heni Hermaliani. "Implementasi Web Scraping dan Sentiment Analysis Terhadap Berita Menggunakan Machine Learning", Swabumi, 2023 Publication	<1 %
24	Submitted to Konsorsium Perguruan Tinggi Swasta Indonesia II Student Paper	<1 %
25	forumarkeologi.kemdikbud.go.id Internet Source	<1 %
26	digilib.uir.ac.id Internet Source	<1 %

27	etd.repository.ugm.ac.id Internet Source	<1 %
28	kc.umn.ac.id Internet Source	<1 %
29	vdocuments.net Internet Source	<1 %
30	www.neliti.com Internet Source	<1 %
31	www.scribd.com Internet Source	<1 %
32	www.suarantb.com Internet Source	<1 %
33	www.unud.ac.id Internet Source	<1 %
34	Rismawati Rismawati, Solmin Paembonan, Rinto Suppa. "RANCANG BANGUN KEAMANAN PINTU OTOMATIS MENGGUNAKAN E-KTP BERBASIS ARDUINO UNO", Jurnal Informatika dan Teknik Elektro Terapan, 2024 Publication	<1 %
35	ejurnal.its.ac.id Internet Source	<1 %

AKRAM 105841111920 BAB III

by Tahap Tutup

Submission date: 21-Aug-2024 02:37PM (UTC+0700)
Submission ID: 2435429058
File name: BAB_III_-_2024-08-21T153629.369.docx (68.21K)
Word count: 1495
Character count: 9557

AKRAM 105841111920 BAB III

ORIGINALITY REPORT

10%	7%	4%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	digilibadmin.unismuh.ac.id Internet Source	3%
2	Submitted to Universitas Muhammadiyah Makassar Student Paper	1%
3	Submitted to Universitas PGRI Palembang Student Paper	1%
4	digilib.iain-palangkaraya.ac.id Internet Source	1%
5	Submitted to STT PLN Student Paper	1%
6	frangao.net Internet Source	1%
7	repository.uinjambi.ac.id Internet Source	1%
8	repository.untag-sby.ac.id Internet Source	1%

AKRAM 105841111920 BAB IV

by Tahap Tutup

AKRAM 105841111920 BAB IV

ORIGINALITY REPORT

4%	4%	1%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	1%
2	dspace.lib.ntua.gr Internet Source	1%
3	repositori.uji.es Internet Source	1%
4	repository.ipb.ac.id Internet Source	1%
5	Rudi Kurniawan, Antoni Zulius. "Smart Home Security Menggunakan Face Recognition Dengan Metode Eigenface Berbasis Raspberry Pi", Jurnal Sustainable: Jurnal Hasil Penelitian dan Industri Terapan, 2019 Publication	1%

Exclude quotes Off

Exclude bibliography Off

Exclude matches < 1%

J

AKRAM 105841111920 BAB V

by Tahap Tutup

AKRAM 1058411111920 BAB V

ORIGINALITY REPORT

5%	5%	0%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	digilib.iain-palangkaraya.ac.id Internet Source	4%
2	digilibadmin.unismuh.ac.id Internet Source	2%

LULUS

Exclude quotes

Exclude bibliography

Exclude matches