Lembar Pengesahan

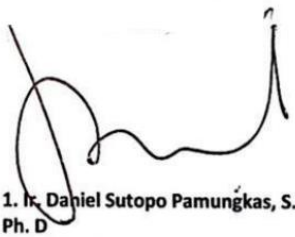
Proposal Tugas Akhir disusun untuk digunakan sebagai rencana kerja pada pelaksanaan Tugas Akhir

Disusun oleh:
Elsie Tria Paramian (4212331021)

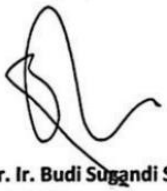
Tanggal Seminar: 21 Mei 2024

Disetujui oleh :

1. Ir. Indra Hardian Mulyadi, S.T., M.Eng., Ph.D
NIK: 117179



1. Ir. Daniel Sutopo Pamungkas, S.T., M.T., Ph. D
NIK: 100006



2. Dr. Ir. Budi Sugandi S.T., M.Eng., IPM
NIK: 100002

FORMULIR LOOGBOOK BIMBINGAN DAN PENGAJUAN SEMINAR PROPOSAL/SIDANG TUGAS AKHIR*

Nama : Elsie Tria Paramian
NIM : 4212331021
Pembimbing I : Ir. Daniel Sutopo Pamungkas S.T., M.T., Ph.D
Judul : Sistem Klasifikasi Bahasa Isyarat secara Realtime menggunakan SSD Mobilnet dengan Tensorflow

No	Hari/Tgl	Rincian Kegiatan	TTD Pembimbing I
1	26/11/24	Penyampaian Progres Tugas Akhir	
2	27/11/24	Konsultasi Problem labelImg.	
3	29/11/24	Konsultasi program	
4	3/12/24	menentukan cara menghasilkan data	
5	5/12/24	pengambilan data.	
6	10/12/24	pengolahan data.	
7	12/12/24	Penyusunan Buku TA	
8	19/12/24	Penyusunan Buku TA.	
9	20/12/24	Revisi penulisan Buku.	
10	24/12/24	Revisi akhir	

Berdasarkan hasil bimbingan yang telah dilaksanakan selama 2. bulan dan telah disetujui oleh dosen pembimbing, maka dengan ini saya mengajukan diri sebagai peserta Seminar Proposal /Sidang Tugas Akhir*.

Batam, 24 - 12 - 2024
Peserta

Elsie Tria Paramian

NIM: 4212331021

Perbaikan Proposal dan Buku Tugas Akhir (v0)

Program Studi Teknik Mekatronika
polibatam

Judul lama: Sistem Klasifikasi Bahasa Isyarat secara Realtime menggunakan SSD MobilNet dengan tensorflow

Judul baru: Sistem Klasifikasi Bahasa Isyarat secara Realtime menggunakan SSD MobilNet dengan tensorflow

Nama Mahasiswa:Elsie Tria Paramian

NIM Mahasiswa: 4212331021

Pembimbing 1 (P1):Daniel Sutopo Pamungkas

Pembimbing 2 (P2): -

Penguji 1 (E1): Indra Hardia Mulyadi

Penguji 2 (E2): Budi Sugandi

Tanggal Sidang/Seminar: 21 Januai 2025

Tabel Daftar Perbaikan

No.	Komentar	Perbaikan	Halaman lama (s)	Halaman baru (s)
Komentar General				
1 E1	Perbaikan penulisan, masih banyak yang typo	Buku TA	-	Keseluruhan halaman
Abstract				
1 E1	Tambahin Hasil dari pengujian	Abstrak sudah direvisi, dengan menambahkan hasil dari pengujian	iii and iv	iv and iv
Bab 2				
1 E1	Penambahasan penjelasan dari convolusi dari gambar 3.2 prosess ssd	Sudah ditambahkan penjelasan kerja convolusi	11	11
1 E1	Penambahan literatur yang mirip	Sudah ditambahkan literatur (penelitian terdahulu)	-	17 & 18
1 E1	Memperjelas gambar 3.2	Sudah diganti dengan gambar yang lebih jelas1	11	11
Bab 3				
1 E2	Penambahan data uji 4 orang	Sudah ditambahkan metode data uji dengan 4 orang	-	31
1 E2	Penambahan jarak 0cm, 25cm, 50cm, 100cm, 150cm	Sudah ditambahkan metode pengujian jarak	27 and 28	27 and 28
1 E1	Penambahan spesifikasi komputer	Sudah ditambahkan	-	19
1 E1	Penambahn keterangan metode pengambilang gambar dengan ekspresi dan tidak	Sudah ditambahkan	-	20 and 21
1 E1	Menambahkan timestamp	Sudah ditambahkan proses deteksi dan output sistem	-	32
Bab 4				
1 E2	Penambahan hasil untuk data uji 4 subjek (orang)	Sudah ditambahakan pengujian untuk 4 orang	-	40

1 E2	Penambahan jarak 0c,. 25 cm. 50cm. 100cm, 150cm	Sudah ditambahkan hasil pengujian dengan tambahan jarak 0cm dan 25cm	33 and 34	33 and 34

Batam, 28 Januari 2025

Elsie Tria Paramian

Sistem Klasifikasi Bahasa Isyarat secara *Realtime* menggunakan SSD MobilNet dengan Tensorflow

Tugas Akhir

Oleh:
Elsie Tria Paramian (4212331021)

Program Studi Teknik Mekatronika
Jurusan Teknik Elektro
Politeknik Negeri Batam
2022

Pernyataan Keaslian Tugas Akhir

Saya yang bertandatangan dibawah ini menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya yang berjudul : "Sistem Klasifikasi Bahasa Isyarat secara *Realtime* menggunakan SSD MobilNet dengan Tensorflow" adalah **hasil karya sendiri**, **diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan**, dan **bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.** Semua referensi yang dikutip atau dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan saya ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Batam, 24 Januari 2025

Elsie Tria Paramian
NIM: 4212331021

Lembar Pengesahan

Tugas Akhir disusun untuk memenuhi salah satu syarat memperoleh gelar Sarjana Terapan Teknik (S.Tr.T)
di
Politeknik Negeri Batam

Oleh:
Elsie Tria Paramian (4212331021)

Tanggal Sidang: 21 Januari 2025

Disetujui oleh :

1. Ir. Indra Hardian Mulyadi S.T., M.Eng., Ph.D
NIK: 117179

1. Ir. Daniel Sutopo Pamungkas S.T., M.T., Ph.D
NIK: 100006

2. Dr. Ir Budi Sugandi S.T.,M.Eng., IPM
NIK: 100002

Sistem Klasifikasi Bahasa Isyarat secara *Realtime* menggunakan SSD MobileNet dengan TensorFlow

Abstrak

Bahasa isyarat adalah metode komunikasi penting bagi komunitas tunarungu. Namun, pengembangan sistem deteksi bahasa isyarat secara *realtime* masih menghadapi tantangan, terutama dalam hal akurasi dan keandalan. Dalam penelitian ini, kami mengusulkan sistem deteksi bahasa isyarat real-time berbasis Tensorflow dengan arsitektur SSD MobileNet. Sistem ini diuji dalam berbagai skenario, termasuk pengujian berdasarkan jarak, pengujian offline, serta pengujian *realtime* dengan berbagai subjek.

Hasil pengujian menunjukkan bahwa jarak memengaruhi performa sistem secara signifikan. Pada pengujian jarak, akurasi tertinggi (90,2%) dicapai pada jarak 50 cm, sementara pada jarak 0 cm dan 150 cm, sistem tidak mampu mendeteksi dengan baik (0%). Dalam pengujian *offline*, sistem mencapai akurasi 100% untuk semua kelas bahasa isyarat, menunjukkan performa optimal dalam kondisi terkontrol. Namun, pada pengujian real-time, akurasi bervariasi tergantung pada subjek dan kelas bahasa isyarat, dengan akurasi tertinggi dicapai pada kelas "Wah Keren" (96%) dan terendah pada kelas "Tolong" (76%).

Pengujian dengan empat subjek tambahan menunjukkan bahwa tingkat akurasi deteksi bergantung pada gestur yang diuji, dengan beberapa gestur seperti "Halo" dan "Sama-sama" menunjukkan akurasi tinggi di berbagai subjek. Secara keseluruhan, penelitian ini menunjukkan bahwa sistem berbasis SSD MobileNet dapat mengenali bahasa isyarat secara *realtime* dengan akurasi yang baik, terutama pada jarak optimal. Hasil ini memberikan kontribusi terhadap pengembangan teknologi yang dapat meningkatkan aksesibilitas dan komunikasi bagi komunitas tunarungu.

Kata kunci: Deteksi Bahasa Isyarat secara *Realtime, TensorFlow, SSD* MobilNet

Real-Time Sign Language Classification System Using SSD MobileNet with TensorFlow.

Abstract

Sign language is an essential communication method for the deaf community. However, the development of real-time sign language detection systems still faces significant challenges, especially regarding accuracy and reliability. In this study, we propose a real-time sign language detection system based on TensorFlow with the SSD Mobilenet architecture. The system was tested in various scenarios, including distance-based testing, offline testing, and real-time testing with different subjects. The results show that distance significantly impacts the system's performance. In distance-based testing, the highest accuracy (90.2%) was achieved at 50 cm, while at 0 cm and 150 cm, the system failed to detect accurately (0%). In offline testing, the system achieved 100% accuracy for all sign language classes, showing optimal performance in controlled conditions. However, in real-time testing, accuracy varied depending on the subject and sign language class, with the highest accuracy in the "Wah Keren" class (96%) and the lowest in the "Tolong" class (76%). Testing with four additional subjects revealed that the detection accuracy depends on the gestures being tested, with gestures like "Halo" and "Sama-sama" showing high accuracy across subjects. Overall, the study demonstrates that the SSD Mobilenet-based system can recognize sign language in real-time with good accuracy, especially at an optimal distance. These results contribute to the development of technology that can improve accessibility and communication for the deaf community.

Keywords: Real-time sign language detection, TensorFlow, SSD MobilNet

Kata Pengantar

Puji syukur kehadirat Tuhan Yang Maha Esa, atas limpahan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul **"Sistem Klasifikasi Bahasa Isyarat Secara *Realtime* Menggunakan *SSD*** MobilNetV2 **dengan *Tensorflow*".** Tugas Akhir ini disusun sebagai salah satu syarat untuk menyelsaikan pendidikan pada program Studi Teknik Mekatronika, Politeknik Negeri Batam.

Penyusunan Tugas Akhir ini tentu tidak terlepas dari bantuan, dukungan, dan motivasi dari berbagai pihak yang telah memberikan kontribusi besar. Oleh karena itu, dengan penuh rasa hormat dan terima kasih, penulis ingi menyampaikan penghargaan kepada:

1. Allah S.W.T, atas kekuatan dan petunjuk yang diberikan selama proses penyusunan Tugas Akhir ini.
2. Orang tua dan keluarga tercinta, atas doa, kasih sayang, dan dukungan moril maupun materil yang tiada henti sepanjang perjalanan pendidikan ini.
3. Bapak Ir. Daniel Sutopo Pamungkas S.T., M.T., Ph.D., yang telah memberikan bimbingan, masukan, dan arahan yang sangat berarti dalam proses penyusunan Tugas Akhir ini.
4. Bapak Dr. Budi Sugandi, S.T., M. Eng., Selaku Ketua Jurusan Teknik Elektro.
5. Bapak Ir. Indra Hardian Mulyadi S.T., M.Eng., Ph.D., Selaku wali dosen kelas RPL**.**
6. Seluruh dosen dan staf pengajar di Politeknik Negeri Batam, atas ilmu, pengalaman, dan dedikasi yang diberikan selama masa perkuliahan.
7. Teman-teman seperjuangan, yang selalu memberikan semangat, bantuan,dan kebersamaan selama masa penyelesain Tugas Akhir.
8. M. Zulfikarlingga Pratama, yang dengan tulus telah memberikan dukungan, bantuan, serta motivasi yang luar biasa selama proses penyelesain Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih jauh dari sempurna. Oleh karena itu, penulis terbuka terhadap kritik dan saran yang membangun demi pengembangan karya ini di masa depan. Akhir kata, semoga Tugas Akhir ini dapat memberikan manfaat bagi semua pihak, khususnya dalam pengembangan ilmu pengetahuan di bidang mekatronika.

Batam, 21 Desember 2024

Elsie Tria Paramian

Daftar Isi

Daftar Gambar

Daftar Tabel

Bab 1. Pendahuluan

1.1. Latar Belakang

Manusia adalah makhluk hidup yang saling berinteraksi. Sebagai makhluk sosial, manusia tidak dapat hidup sendiri dan selalu membutuhkan kerjasama dengan orang lain [1]. Manusia membutuhkan media interaksi untuk berkomunikasi agar dapat saling berhubungan.

Komunikasi dapat didefinisikan sebagai aktivitas yang dilakukan untuk menyampaikan pesan dari satu sumber ke sumber lainnya [2]. Komunikasi sangat diperlukan agar dapat memberikan informasi, menyampaikan emosi dan menyampaikan persepsi. Terdapat banyak macam media komunikasi yang bisa digunakan salah satunya adalah bahasa isyarat.

Bahasa isyarat umumnya digunakan sebagai media komunikasi bagi para Tuli [1]. Meskipun bahasa isyarat memudahkan komunikasi antar Tuli, bahasa isyarat sulit dipahami oleh masyarakat pada umumnya. Oleh karena itu, pengenalan bahasa isyarat sangat diperlukan sebagai upaya untuk mempermudah komunikasi antara Tuli dan non Tuli.

Bentuk dari isyarat tersebut biasanya direpresentasikan dalam bentuk isyarat tangan. Terdapat beberapa penelitian yang telah dilakukan mengenai hand gesture to text dalam berbagai bahasa isyarat seperti: *American Sign Language* [3], *Arabic Sign Language* [4*], Bengali Sign Language* [5], *Peruvian Sign Language* [6] dengan menggunakan berbagai metode.

Bryan Berrú-Novoa et al. dalam penelitiannya [6] melakukan pengenalan Peruvian Sign Language menggunakan kamera resolusi rendah dengan metode *Support Vector Machine* (SVM) dan mendapatkan hasil akurasi yang masih belum maksimal. Hal ini dikarenakan sistem masih mengalami kesulitan untuk mengenali latar belakang di area dengan cahaya redup.

Indonesia sendiri memiliki dua jenis bahasa isyarat: Sistem Isyarat Bahasa Indonesia (SIBI) dan Bahasa Isyarat Indonesia. Sejak tahun 1994, sistem Bahasa isyarat yang ditetapkan oleh pemerintah Indonesia sebagai bahasa pengantar di Sekolah Luar Biasa (SLB) adalah Sistem Bahasa Isyarat Indonesia (SIBI). Namun sejak awal penerapannya, SIBI tidak membantu komunikasi para Tuli dan membuat hubungan sosial mereka menjadi terbatas. Kaum Tuli lebih memilih untuk menggunakan Bahasa Isyarat Indonesia (BISINDO) yang dikembangkan oleh penyandang Tuli melalui Gerakan Kesejahteraan Tunarungu Indonesia (GERKATIN).

Penelitian tentang bahasa isyarat Sibi ke teks sudah cukup banyak. Salah satunya adalah yang dilakukan oleh I Putu Wijaya Merta et al. [7] menggunakana metode K-Nearest Neighbor. Pada penelitian ini, tangan akan terdeteksi secara optimak pada jarak 110cm dan posisi tegak. Apabila jarak tangan ke kamera terlalu dekat atau posisi tangan miring sebesar 90^{o} makan tangan tidak akan terdeteksi.

Terdapat juga penelitian lainnya membahas tentang pengenalan SIBI yang dilakukan oleh Rosalina et al. [8] dengan menggunakan Artificial Neural Network. Akurasi yang didapatkan dari penelitian ini suda cukup baik akan tetapi jarak optimal tangan ke kamera kurang lebih 50 cm serta pengujian deteksi tangan harus dilakukan menggunakan sarung tangan berwarna.

Sedangkan penelitian BISINDO mash sangat terbatas dan metode yang digunakan umumnya menggunakan classifier yang membutuhkan ekstraksi ciri sehingga proses komputasi sangat bergantung pada ketepatan ciri yang digunakan. Untuk mengatasi masalah ini, pada penelitian ini akan dikembangkan pengenalan isyarat tangan menggunakan deep learning.

Terdapat beberpa penelitian mengenai pengenalan bahasa isyarat ke teks menggunakan algoritma deep learning, seperti SSD MobilNetV2. Menggunakan SSD MobilNetV2 untuk mengenali disable patient's hand gesture [8]. Penelitian lain yang dilakukan oleh Kin Yun Lun. Menggunakanan SSD MobilNetV2V2 untuk mpengenalan American Sign Language [3]. Penelitian ini menggunakan jaringan Deep Transfer learning.

Hasil dari beberpa penelitian tersebut menunjukkan bahwa hasil akurasi klasifikasi dan kecepatan efisiensi yang di dapatkan sudah cukup baik. Sehingga, pada penelitian ini membuat sistem pengenalan Bahasa Isyarat Indonesia (BISINDO) Ke teks yang memiliki 2000 gambar dan terbagi kedalam 10 kelas pose BISINDO, yaitu: "Halo", "Bagaimana?", "Bermain", "Ingat", "Sedih", "Tolong", "Terima Kasih", "Sama-sama", dan "Wah, keren". dengan menggunakan metode SSD MobileNetV2. Adapun *dataset* yang digunakan pada tugas akhir ini adalah yang telah dikumpulkan oleh peneliti sendiri sebanyak 2000 citra gambar, teridiri dan 10 subjek dimana setiap simbol terdiri dari 200 citra gambar.

1.2. Rumusan Masalah

Berdasarkan latar belakang tersebut, penelitian tentang pengenalan Bahasa Isyarat Indonesia (BISINDO) masih sedikit. Selain itu, metode penelitian yang ada membutuhkan ketepatan dalam menentukan ekstraksi ciri sehingga akurasi *classifier* sangat ditentukan oleh *feature extraction*. Oleh karena itu, penelitian ini akan membahas tentang pengenalan BISINDO dengan menggunakan SSD MobilNetV2 sebagai upaya untuk memudahkan komunikasi antara Tuli dan Masyarakat.

1.3. Tujuan

Penelitian ini bertujuan untuk melihat performansi dari SSD MobilNetV2 dalam mengenali isyarat tangan BISINDO.

1.4. Manfaat

Adapun dari Tugas Akhir ini dapat diperoleh manfaat sebagai berikut:

1. Mahasiswa akan mendapatkan pengalaman praktis dalam pengembangan sistem berbasis deep learning menggunakan TensorFlow, serta implementasi model deteksi objek seperti SSD (*Single Shot Multibox Detector*) MobileNetV2. Ini akan meningkatkan pemahaman mereka tentang konsep-konsep fundamental dalam pembelajaran mesin dan pengolahan citra.
2. Proyek ini memiliki dampak sosial yang signifikan karena membantu memperluas aksesibilitas bagi komunitas yang menggunakan bahasa isyarat. Mahasiswa akan merasa memiliki kontribusi positif terhadap masyarakat dengan mengembangkan teknologi yang memungkinkan komunikasi yang lebih baik bagi individu dengan kebutuhan khusus.

1.5. Batasan

Pada penelitian ini, terdapat beberapa hal yang membatasi masalah diantaranya:

1. Data yang diolah berupa citra.
2. Pembelajaran yang dibuat terbatas pada 10 gerakan yaitu : Halo., Tolong, Terima kasih, Maaf, Sama-sama, Wah, keren, Bagaimana, Bermain, Ingat!, dan Sedih. Bahasa Isyarat Indonesia (BISINDO).
3. Algoritma digunakan adalah MobilNetV2.

Bab 2. Tinjauan Pustaka

2.1. Bahasa Isyarat

Bahasa isyarat adalah Bahasa yang disampaikan melalui gerak tubuh. Dalam Kamus Besar Bahasa Indonesia (KBBI), Bahasa isyarat artinya dalam sistem perlambangannya Bahasa isyarat tidak diucapkan oleh manusia baik menggunakan bunyi atau tulisan. Bahasa isyarat dibuat secara khusus untuk kaum tuna rungu, tuna wicara, tuna netra, dan sebagainya yang menggunakan isyarat (gerakan tangan, kepala, badan, dan sebagainya) [9]. Terlihat pada gambar 2.1 yaitu gerakan tangan bahasa isyarat alfabet.

Gambar 2. 1 Gerakan Tangan Bahasa Isyarat [9]

2.2. Sitem Isyarat Bahasa Indonesia (SIBI)

Sistem Isyarat Bahasa Indonesia (SIBI) merupakan Bahasa isyarat yang resmi di Indonesia dan sudah diresmikan dalam Undang-Undang No.2 Tahun 1989 [4]. SIBI sendiri mengadaptasi Bahasa isyarat *America Sign Language* (ASL) yang hanya menggunakan gerakan satu tangan, oleh karena itu SIBI dan ASL memiliki kemiripan. Sistem Isyarat Bahasa Indonesia (SIBI) lebih sering digunakan pada kegiatan acara resmi seperti milik pemerintah dan kegiatan di sekolah. Namun, pada penggunaanya SIBI tergolong sulit sehingga banyak kaum tuna rungu menggunakan jenis Bahasa isyarat Bahasa Isyarat Indonesia (BISINDO).

2.3. Bahasa Isyarat Indonesia (BISINDO)

BISINDO (Bahasa Isyarat Indonesia) adalah sistem komunikasi visual yang digunakan oleh komunitas Tuli di Indonesia. Bahasa ini berkembang secara alami di antara komunitas Tuli dan berbeda dengan Sistem Isyarat Bahasa Indonesia (SIBI) yang lebih formal dan dikembangkan oleh pemerintah. BISINDO dianggap lebih representatif karena lebih mencerminkan penggunaan sehari-hari oleh penyandang Tuli.

BISINDO mulai dikenal secara luas pada era 2000-an dengan meningkatnya perhatian terhadap hak-hak penyandang disabilitas di Indonesia. Sebelumnya, SIBI lebih sering digunakan dalam pendidikan dan komunikasi resmi. Namun, seiring waktu, BISINDO mulai mendapatkan pengakuan lebih luas karena kemampuannya untuk lebih mudah dipahami dan digunakan oleh komunitas Tuli [33].

2.4. Artificial Intelligence (AI)

Pada tahun 1956, terdapat seorang tokoh terkenal dari *Massacuhetts Institute of Technology* (MIT) adalah John McCarthy, yang pertama kali mengusulkan suatu istilah tentang *'Arificial Intelligence*" di mana beliau menjelaskan tujuan yang dimaksud dengan imu kecerdasan bu atan atau *Artificial Intelligence* (AI) adalah, '*The goal of A1 is to develop machines that behave as though they were intelligent. It is the science and engineering of making intelligent machines, especially intelligent computer programs- It is related to the similar task of using computers to understand human intelligence, but* AI *does not have to confine itself to methods that are biologically oberservable*',

Diterjemahkan dalam pengertian yang dimaksud John McCarthy yang dikutip dari Budiharto (2018), *Artificial Intelligence* merupakan cabang dari ilmu komputer yang berfokus pada pengembangan mesin yang berperilaku cerdas. Perpaduan ilmu dan teknik yang membuat mesin memiliki kecerdasan, khususnya pada program komputer yang cerdas.

Dimulai dari pemaparan yang disampaiakan oleh McCarthy dan peneliti lainnya, kecerdasan buatan menjadi berkembang yang awal mulanya berada pada

tingkat penelitian dilaboratorium menjadi semakin berkembang pesat sampai pada produk-produk bisnis teknologi besar maupun kecil, sehingga disebabkan pengaruh yang sangat dirasakan oleh pemakai. Beberapa pakar mendefinisikan pengertian dari kecerdasan buatan atau Artificial Intelligence (AI) [10], sebagai berikut :

1. Schalkoff (1990): AI adalah bidang studi yang berusaha menerangkan dan meniru perilaku cerdas dalam bentuk proses komputasi [10].
2. Rich dan Knight (1991): AI adalah studi tentang cara membuat komputer melakukan sesuatu yang, sampai saat ini, orang dapat melakukannya lebih baik [10].
3. Luger dan Stubblefield (1993): AI adalah cabang ilmu komputer yang berhubungan dengan otomasi perilaku yang cerdas [10].
4. Haag dan Keen (1996): AI adalah bidang studi yang berhubungan dengan penangkapan, pemodelan, dan penyimpanan kecerdasan manusia dalam sebuah sistem teknologi informasi sehingga sistem tersebut dapat memfasilitasi proses pengambilan keputusan yang biasanya dilakukan oleh manusia [10].

Dari penjelasan yang diungkapkan beberapa ahli tersebut dapat disimpulkan bahwa kecerdasan bauatan atau Artificial Intelligence (AI) merupakan suatu bidan dari cabang ilmu komputer yang berfokus pada peniruan, pengkapan, perilaku, pemodelan serta penyimpanan yang ada pada kecerdasan manusia di dalam sebuah sistem teknologi informasi sehingga sistem dapat menjembatani proses pengambilan keputusan secara langsung yang biasanya hanya dapat dilakukan oleh manusia.

2.5. Machine Learning

Algoritma pada pembelajaran mesin adalah pengembangan teknologi yang mampu meniru kecerdasan manusia [12]. instruksi program sulit untuk dipecahkan. Seperti contohnya, sebuah komputer yang melakukan tugas untuk mengenali wajah dalam gambar [13]. Machine Learning memerlukan pelatihan yang sesuai kegunaanya dengan menggunakan data sampel untuk pelatihannya. Machine learning tidak perlu diprogram ulang, pembelajaran mesin dikembangkan dengan aspek ilmu lainnya seperti statistika, matematika, dan data mining [14].

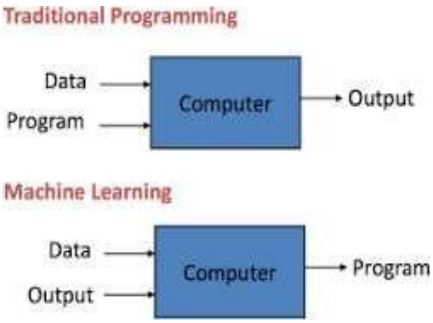
Dalam beberapa tahun terakhir, machine learning yang bagian dari kecerdasan buatan memiliki peran penting dalam penelitian ilmiah [15]. Dalam kehidupan sehari-hari penerapan *Machine Learning* mudah ditemukan. Seperti contohnya fitur *face unlock*, *recommendation search*, dan rekomendasi produk *Marketplace*. Machine Learning memiliki beberapa Teknik, dua diantaranya adalah *Supervised* dan *Unsupervised.* Teknik Supervised merupakan pembelajaran mesin yang diawasi, yang artinya pembelajaran mesin ini memberikan label tertentu pada informasi yang sudah ada pada data, sedangkan Teknik Unsupervised digunakan

pada data yang tidak memiliki informasi. Dalam menerima prediksinya, algoritma ini tidak menerima feedback. Teknik ini menemukan polanya sendiri dalam data [16].

Machine Learning tumbuh Ketika urutan *machine learning* merupakan salah satu dari bidang ilmu komputer yang memberikan pembelajaran kepada komputer untuk mengetahui sesuatu tanpa pemrogram yang jelas. Sedangkan pendapat lain menyatakan bahwa *machine learning* dapat didefinisikan sebagai metode komputasi berdasarkan pengalaman untuk meningkatkan performa atau membuat prediksi yang akurat [17]. Di dalam definisi pengalaman tersebut dimaksudkan informasi yang ada sebelumnya yang telah tersedia dan bisa dijadikan data pembelajar. Adapun dalam definisi lain dari *machine learning* yang dijabarkan oleh beberapa ahli [18], sebagai berikut:

1. Mitchel (1997): Komputer yang memiliki kemampuan melakukan belajar dari pengalaman terhadap tugas-tugasnya dan mengalami peningkatan kinerja [18].
2. Budiharto (2016): Tipe dari kecerdasan buatan yang menyediakan komputer dengan kemampuan untuk belajar dari data, tanpa secara eksplisit harus mengikuti instruksi terprogram [18].

Dapat disimpulkan bahwa pengertian yang dimaksud tentang *machine learning* adalah suatu bidang ilmu komputer yang merupakan cabang dari *Artificial Intelligence* (AI) yang memiliki kemampuan pembelajaran dari pengalaman yang ada seperti data-data atau informasi yang sudah ada sebelumnya untuk meningkatkan performa ataupun membuat prediksi yang tepat dan akurat. Dapat dilihat pada gambar 2.2 diagram mechine learning.



Gambar 2. 2 Diagram *Machine Learning* [19]

Dalam keterangan pada Gambar 2.2 bahwa terdapat perbedaan antara konsep pemrograman tradisional dengan pemrograman menggunakan algoritma machine learning. Dari sisi metode pembelajaran machine learning yang diungkapkan bahwa algoritma machine learning dapat dikategorikan dalam beberpa skenario [19], yaitu:

1. **Supervised Learning**
   Pada penggunaan tipe *Supervised Learning*, ialah teknik pembelajaran mesin atau *machine learning* dengan memasukkan data latih untuk melakukan kegiatan pembelajaran yang telah diberikan tiap label, kemudian membuat prediksi dari data yang telah diberi label.
2. ***Unsupervised Learning***
   Pada penggunaan tipe *Unsupervised Learning*, ialah teknik pembelajaran mesin yang memasukan data latih tanpa diberikan label, kemudian mencoba untuk mengelompokkan data berdasarkan karakteristik- karakteristik yang diketahui.
3. ***Reinforcement Learning***
   Pada penggunaan tipe *Reinforcement Learning*, ialah teknik yang mengajarkan bagaimana cara bertindak untuk menghadapi suatu masalah, di mana tindakan tersebut mempunyai dampak sehingga metode ini diterapkan pada agen cerdas agar dapat menyesuaikan dengan kondisi lingkungannya.

2.6. Deep Learning

Dalam suatu penelitian yang dijelaskan oleh Deng & Yu (2013) menerangkan apa yang dimaksud dengan konsep *Deep Learning*, bahwsanya konsep tersebut merupakan salah satu teknik pada machine learning yang memanfaatkan banyak layer pengolahan informasi nonlinier untuk melakukan ekstraksi fitur, baik pengenalan pola, dan klasifikasi [25].

Ditambah lagi penjelasan lain yang diungkapkan oleh Wehle (2017) bahwa *deep learning* adalah, "*a form of machine learning that can utilize either supervised or unsupervised algorithms, or both*..", dan diungkapkan kembali, "..*deep learning has recently seen a surge in popularity as way to accelerate the solution of certain types of difficult computer problems, most notably in the computer vision and natural language processing (NLP) fields*." [26].

Diterjemahkan bahwa yang dimaksud *deep learning* adalah suatu cabang pada bidang machine learning yang bisa memanfaatkan algoritma dari *supervised* atau *unsupervised*, maupun keduanya sehingga dapat mempercepat pencarian solusi dari bermacam-macam masalah komputer, khususnya pada bidang *computer vision* dan *natural language processing* (NLP) [26].

Adapun dalam definisi lain, *deep learning* merupakan cabang ilmu dari machine learning yang berbasis Jaringan Syaraf Tiruan (JST) atau bisa dikatakan

dari JST tersebut yang mengajarkan komputer untuk dapat melakukan tindakan yang dianggap alami oleh manusia dan direpresentasikan dengan membentuk arsitektur jaringan syaraf yang mempunyai banyak layer (lapisan).

Dapat disimpulkan dari pengertian yang dijelaskan oleh beberapa ahli yakni, deep learning merupakan suatu cabang teknik dari *machine learning* yang memanfaatkan Jaringan Syaraf Tiruan (JST) sehingga memiliki banyak lapisan-lapisan layer yang mengelola informasi nonlinier sehingga meningkat perecepatan dalam pencarian solusi dari berbagai masalah seperti regresi klasifikasi, dan sebagainya.

2.7. Klasifikasi

Salah satu metode algoritma yang umum digunakan pada konsep Machine Learning maupun *Deep Learning* untuk mencari solusi dari permasalahan yang sering muncul ialah, metode klasifikasi. Menurut beberapa ahli pun memaparkan definisi masing-masing terhadap istilah klasifikasi tersebut, yaitu sebagai berikut:

1. Klasifikasi diartikan sebagai suatu proses untuk memperoleh model ataupun fungsi yang melukiskan dan membedakan kelas data maupun konsep yang mempunyai tujuan memprediksikan kelas untuk data yang tidak dikenali kelasnya[27].
2. Klasifikasi merupakan suatu pekerjaan menilai objek data untuk memasukkannya ke dalam kelas tertentu dari sejumlah kelas yang tersedia[28].

Jadi dapat disimpulkan dari beberapa definisi tersebut bahwa klasifikasi merupakan suatu pekerjaan dalam memperoleh model atau fungsi untuk menilai objek data atau memprediksi data pada kelas sesuai dengan perolehan kelas yang telah tersedia sebelumnya.

2.8. Objek Deteksi

Objek deteksi adalah proses pengenalan yang dapat membedakan objek dari latar belakang dan objek mencurigakan lainnya, seperti mobil dan jalan [29]. Sedangkan menurut Michelluci & Moolayil (2019), objek deteksi ialah pengenalan instance objek dalam gambar dan menentukan objek yang diketahui dengan tanda kotak pembatas di sekitaran objek [30].

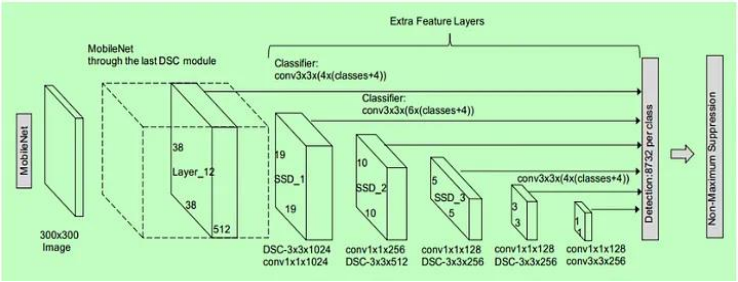
Kemudian, detector objek akan menampilkan daftar objek yang terdeteksi dengan beberapa informasi terkait yang terdiri atas [31] :

1. Kelas objek (Misal: orang, mobil, hewan, dan sebagainya)
2. Probabilitas (skor kepercayaan) dengan rentang sekitar 0–1 yang menandai seberapa yakin detektor terhadap objek yang berada di lokasi tersebut layaknya output klasifikasi biasa.
3. Koordinat wilayah segi empat dari gambar tempat objek berada, kotak tersebut biasanya disebut ground truth atau kotak pembatas.

2.9. Single Shot Multiboc Detector (SSD)

Single Shot Multibox Detector (SSD) merupakan salah satu algoritma pendeteksian object berupa gambar paling populer, hal tersebut dikarenakan algoritma SSD mudah diimplementasikan. Hasil akurasi pada algoritma SSD cukup baik relatif terhadap komputasi yang dibutuhkan. Metode ini termasuk kedalam pendeteksian objek real time [20]. Cara kerja pada algoritma SSD didasarkan pada *feed-forward convolutional network*, yaitu pada suatu objek akan ditentukan bounding boxes yang berukuran tetap dan pada area *bounding boxes* tersebut akan menampilkan nilai skor masing-masing pada setiap kelas objek [21]. arsitektur VGG- 16 dan extra feature layers, kedua arsitektur jaringan inilah yang ada pada arsitektur algoritma SSD. Kinerja pada jaringan VGG-16 sangat kuat dengan kualitas yang tinggi untuk citra gambar sehingga digunakan sebagai based-network. Bagian lapisan fitur konvolusional ditambahkan ke ujung base network oleh SSD sebanyak enam layer konvolusi ekstra yang berfungsi memprediksi dengan aspek rasio berbeda [22].

SSD (*Single Shot MultiBox Detector*) bekerja dengan menggunakan jaringan saraf konvolusi untuk memproses gambar dan menghasilkan prediksi lokasi dan kelas objek pada gambar. Cara kerja SSD secara umum dimulai dengan ekstraksi fitur. SSD menggunakan jaringan saraf konvolusi untuk mengekstraksi fitur dari gambar input. Setiap layer pada jaringan saraf konvolusi menghasilkan fitur dengan resolusi yang berbeda. Langkah kedua adalah prediksi lokasi dan kelas objek. SSD menggunakan beberapa layer konvolusi terakhir untuk memprediksi lokasi dan kelas objek pada gambar. Setiap layer konvolusi terakhir menghasilkan beberapa kotak pembatas (bounding box) dan skor kelas untuk setiap kotak pembatas. Langkah ketiga Non-maximum suppression. SSD menggunakan teknik non-maximum suppression untuk menghilangkan kotak pembatas yang tumpang tindih dan memilih kotak pembatas dengan skor kelas tertinggi sebagai hasil deteksi. Langkah terakhir post-processing. SSD melakukan *post-processing* pada kotak pembatas yang dipilih untuk meningkatkan akurasi deteksi. *Post-processing* termasuk regresi kotak pembatas untuk meningkatkan presisi lokasi objek dan *filtering* kotak pembatas yang tidak memenuhi kriteria tertentu. Dengan cara kerja ini, SSD dapat menghasilkan deteksi objek yang akurat dan cepat dengan menggunakan satu jaringan saraf dalam satu tahap. Terlihat pada gambar 2.3 yaitu proses SSD.



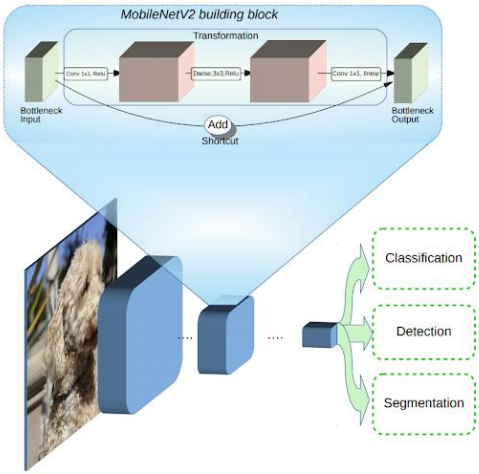
Gambar 2. 3 Proses SSD [22]

Arsitektur SSD dimulai dengan lapisan konvolusi 3x3, yang bertanggung jawab untuk mengekstraksi fitur-fitur lokal pada gambar input. Proses ini diikuti oleh lapisan konvolusi 3x3 Depthwise, yang memperdalam pemahaman terhadap fitur-fitur tersebut dengan melakukan konvolusi terhadap setiap saluran inputsecara terpisah. Selanjutnya, dilakukan konvolusi 1x1 untuk menggabungkan informasi dari seluruh saluran dan mereduksi dimensi. Proses ini diulang dengan konvolusi 3x3 Depthwise, diikuti oleh konvolusi 1x1, untuk terus meningkatkan kompleksitas representasi fitur.Rangkaian konvolusi 3x3 Depthwise dan 1x1 terus dilakukan secara berulang, memungkinkan model untuk mengekstraksi hierarki fitur yang semakin kompleks. Proses ini memberikan fleksibilitas yang diperlukan untuk mengenali pola-pola yang lebih bstrak dan kontekstual dalam data citra. Terakhir, setelah serangkaian konvolusi, dilakukan lapisan *Average Pooling* untuk merata-ratakan hasil dari konvolusi-konvolusi sebelumnya, menghasilkan representasi fitur yang global. Selanjutnya, *Fully Connected layer* digunakan untuk menyatukan informasi dari seluruh fitur dan Softmax diaplikasikan pada lapisan terakhir untuk menghasilkan probabilitas kelas *output*. Dengan demikian, arsitektur ini memberikan pendekatan yang efektif dalam ekstraksi fitur pada citra, memungkinkan pengenalan pola dengan tingkat akurasi yang tinggi.

2.10. MobilNet

MobileNet merupakan salah satu arsitektur *Convolutional Neural Network* (CNN) yang dapat digunakan untuk mengatasi kebutuhan akan sumber komputasi berlebih. MobileNet dapat digunakan pada platform mobile maupun tertanam, arsitektur ini dibangun oleh para peneliti dari Google atas dasar kebutuhan arsitektur pada CNN. Perbedaan arsitektur MobileNet dan arsitketur CNN adalah penggunaan pada layer konvolusi yang dimana ketebalan filternya sesuai dengan ketebalan dari input gambar [23].

MobileNet menggunakan *depthwise separable convolution* yang secara signifikan mengurangi jumlah parameter jika dibandingkan dengan *reguler convolution,* dari hal tersebut MobileNet menjadi *deep neural network* yang sangat ringan. Pada model MobileNetv2 terdapat 53 convolution layer dan memperkenalkan 2 fitur baru ke arsitektur yaitu Linear Bootleneck diantara layer dan *shortcut* diantara *bootleneck layg* (Inverted Residua) Terdapat 2 tipe *convolution layer* pada mobileNetV2 yaitu 1x1 *convolution* dan 3x3 *Depthwise Convolution* Pada Gambar 2.6 kotak biru menggambarkan blok pembentukan *linear convolution* atau *bootleneck*. Dengan penambahan fitur tersebut pada percobaan dengan dataser COCO untuk deteksi *realtime* pada paper (Sandler, 2019) dengan perbandingan oleh model sebelumnya yaitu MobileNetV2 terdapat parameter yang digunakan MobileNetV2 lebih sedikit 800 ribu parameter dan penggunaan CPU lebih rendah 70ms yang dapat diartikan 2 lebih ringan pada komputasinya. Maka dari itu penelitian ini menjadi menggunakan model MobileNet-SSD. Terlihat pada gambar 2.4 arsitektur MobilNetV2.



Gambar 2. 4 Arsitektur MobilNetV2 [32]

MobileNet adalah arsitektur jaringan saraf konvolusional (CNN) yang dapat digunakan di berbagai platform dan kerangka kerja machine learning selain TensorFlow. Meskipun MobileNet awalnya dikembangkan dengan menggunakan TensorFlow, banyak platform dan kerangka kerja lainnya telah mengimplementasikan versi MobileNet mereka sendiri atau menyediakan dukungan untuk MobileNet.

Beberapa contoh kerangka kerja dan platform yang mendukung atau menggunakan MobileNet di luar TensorFlow adalah:

a) PyTorch: PyTorch, kerangka kerja deep learning populer lainnya, memiliki implementasi MobileNet di dalamnya. MobileNet di PyTorch untuk berbagai tugas seperti klasifikasi gambar atau deteksi objek.
b) Keras: Keras, sebuah antarmuka neural network high-level yang kompatibel dengan beberapa backend seperti TensorFlow dan Theano, juga memiliki implementasi MobileNet yang tersedia.
c) Caffe: Caffe, kerangka kerja deep learning yang dikembangkan oleh BVLC (Berkeley Vision and Learning Center), juga memiliki implementasi MobileNet yang tersedia. Anda dapat menggunakan MobileNet di Caffe untuk berbagai tugas penglihatan komputer.
d) OpenVINO: Intel OpenVINO (Open Visual Inference and Neural Network Optimization) Toolkit menyediakan dukungan untuk banyak arsitektur jaringan saraf, termasuk MobileNet, untuk dijalankan di berbagai perangkat keras Intel, seperti CPU, GPU, dan FPGA.
e) ONNX: ONNX (Open Neural Network Exchange) adalah format model open-source yang dirancang untuk portabilitas antara berbagai kerangka kerja machine learning. Anda dapat mengonversi model MobileNet dari TensorFlow ke ONNX dan menggunakan model tersebut di berbagai kerangka kerja yang mendukung ONNX.

2.11. SSD MobilNetV2

Kombinasi SSD dengan MobileNet menghasilkan model deteksi objek yang cepat dan efisien, cocok untuk aplikasi real-time. SSD MobileNet menggunakan MobileNet sebagai backbone untuk ekstraksi fitur dari gambar, sementara SSD berfungsi sebagai detektor yang menghasilkan bounding box dan klasifikasi objek. Implementasi ini memungkinkan deteksi objek yang cepat dengan konsumsi memori dan daya yang rendah, sangat penting untuk aplikasi di perangkat mobile dan embedded.

SSD MobileNet adalah salah satu kombinasi arsitektur yang paling sering digunakan untuk tugas ini karena keunggulannya dalam hal kecepatan dan efisiensi komputasi. MobileNet, yang menggunakan depthwise separable convolutions, mengurangi jumlah parameter dan kompleksitas komputasi, sementara SSD (Single Shot MultiBox Detector) memungkinkan deteksi objek secara cepat dalam satu kali inferensi[34][35].

Berikut adalah perbandingan akurasi antara SSD MobileNet dan beberapa algoritma lainnya yang telah digunakan dalam penelitian sebelumnya untuk tugas klasifikasi bahasa isyarat:

1. SSD MobileNet: Penelitian menunjukkan bahwa SSD MobileNet dapat mencapai akurasi sekitar 92% hingga 95% dalam klasifikasi bahasa isyarat pada dataset yang kompleks. Keunggulan utama dari kombinasi ini adalah kecepatan dan efisiensinya, yang membuatnya sangat cocok untuk aplikasi real-time[34][35].
2. YOLO (You Only Look Once): YOLO juga digunakan untuk deteksi dan klasifikasi objek secara real-time. Akurasi YOLO untuk tugas klasifikasi bahasa isyarat biasanya berkisar antara 88% hingga 92%. Meskipun cepat, YOLO sering kali tidak seefisien SSD MobileNet dalam hal konsumsi daya dan sumber daya komputasi[36].
3. Faster R-CNN: Algoritma ini dikenal dengan akurasinya yang tinggi, sering mencapai sekitar 94% hingga 96% dalam tugas klasifikasi bahasa isyarat. Namun, Faster R-CNN membutuhkan lebih banyak sumber daya komputasi dan tidak secepat SSD MobileNet, membuatnya kurang ideal untuk aplikasi real-time yang memerlukan respons cepat[37].
4. InceptionV3 dengan RPN (Region Proposal Network): InceptionV3 yang digabungkan dengan RPN juga menunjukkan akurasi tinggi, sekitar 93% hingga 95%. Namun, seperti Faster R-CNN, kombinasi ini lebih lambat dan lebih kompleks dibandingkan SSD MobileNet [38].

Dari perbandingan ini, dapat dilihat bahwa SSD MobileNet tidak hanya menawarkan akurasi yang kompetitif tetapi juga unggul dalam hal kecepatan dan efisiensi komputasi. Ini menjadikannya pilihan yang sangat cocok untuk implementasi sistem klasifikasi bahasa isyarat secara real-time. Dengan menggunakan TensorFlow sebagai kerangka kerja, SSD MobileNet dapat

dioptimalkan lebih lanjut untuk perangkat mobile dan aplikasi yang memerlukan deteksi dan klasifikasi langsung.

2.12. Tensorflow

Tensorflow merupakan *open source library* yang dibangun menggunakan bahasa pemrograman python. Tensorflow digunakan pada komputasi numerik pembelajaran mesin dengan skala besar [24]. Model algoritma pembelajaran mesin dan jaringan syaraf digabungkan dalam satu set pada tensorflow. *Library* ini dikembangkan untuk melakukan pekerjaan pada pembelajaran mesin dan jaringan syaraf dalam. Untuk mempermudah perhitungan ekspresi matematis, *tensorflow* menggabungkan aljabar komputasi dan teknik pengotimalan kompilasi, sehingga masalah waktu dapat dikurangi dalam melakukan perhitungan [39].

Tensorflow adalah *framework* yang sangat serbaguna dan mendukung berbagai algoritma dalam pembelajaran mesin dan deep learning. Berikut adalah beberapa jenis algoritma yang dapat diimplementasikan menggunakan TensorFlow:

1. Faster R-CNN: Model deteksi objek yang lebih akurat tetapi lebih lambat dibandingkan SSD. Menggunakan region proposal *network* untuk menghasilkan kandidat bounding box sebelum klasifikasi[37].
2. YOLO (*You Only Look Once*): YOLO juga menawarkan deteksi objek dalam satu langkah, tetapi SSD MobileNet mungkin lebih unggul dalam hal akurasi, terutama dalam mendeteksi objek kecil, sementara YOLO lebih unggul dalam hal kecepatan[36].
3. SSD MobilNetV2: SSD MobileNet dirancang untuk memungkinkan deteksi objek real-time dengan kecepatan tinggi. Kombinasi SSD (*Single Shot Detector*) dengan MobileNet sebagai *backbone* (yang ringan dan efisien) memungkinkan model untuk menghasilkan prediksi dengan cepat, menjadikannya cocok untuk aplikasi yang memerlukan respons instan. [20].

SSD MobileNet adalah pilihan yang sangat baik untuk aplikasi yang membutuhkan deteksi objek real-time yang cepat dan efisien. Dengan menyatukan kecepatan, efisiensi, dan akurasi yang diterima, SSD MobileNet memenuhi kebutuhan banyak aplikasi visi komputer modern yang berfokus pada deteksi objek.

2.13. Open CV

OpenCV (*Open Source Computer Vision Library*) adalah sebuah pustaka perangkat lunak yang ditujukan untuk pengolahan citra dinamis secara *realtime*.

OpenCV, singkatan dari *Open Source Computer Vision Library*, adalah perpustakaan sumber terbuka yang populer digunakan untuk pengolahan citra dan penglihatan komputer. Dengan antarmuka yang mudah digunakan dan berbagai fungsi yang kuat, OpenCV telah menjadi alat utama dalam berbagai aplikasi, mulai dari deteksi objek hingga pengenalan wajah[42].

2.14. Phyton

Dalam sejarahnya, bahasa pemograman python dirilis pertama kali oleh Guido *Van Rossum di Scitchting Mathematisch Centrum* Belanda pada tahun 1991. Sebab mula dia menggunakan nama Python dikarenakan ia penggemar grup komdei Inggris bernama *Monty Python* (Wahyono,_2018), Dari versi awal yaitu Python versi 1.0 pada tahun 1994 telah berkembang sampai saat ini menjadi 2 jenis Python dengan versi berbeda dan terbaru yaitu versi 3.7 dan versi 2.7. Perbedaan mendasar pada *Python* versi 2.0 mempunyai *Python Enhacement Proposal* (PEP) dengan kemampuan dalam pemberian tuntunan informasi bagi para pengguna serta peningkatan dukungan untuk Unicode dan berbagai macam fitur-fitur *programmatical* yang berguna untuk manajemen memori. Sedangkan pada *Python* versi 3.0 adalah pengembangan dari *Python* 2.0 dengan berfokus pada *codebase* dan menghilangkan *redundancy*, bahkan perbedaan terbesamnya yaitu membuat statement print dalam fungsi yang built-in dan ditambah pengingkatan dukungan yang beralih ke Phyton 3.0 sehingga berkurangnya dukungan *Phyton* 2.0.

2.15. Penelitian Terdahulu

Berikut adalah penelitian terdahulu yang relevan dengan proyek ini, yang terlihat pada tabel 1 :

Tabel 1 Penelitian Terdahulu

No	Judul Literatur	Penulis	Fokus Penulisan
1	Sistem Deteksi Bahasa Isyarat Secara *Realtime* Dengan Tensorflow Object Detection dan Python Menggunakan Metode Convolutional Neural Network (CNN)	Muhammad Fikri, Muhammad Iqbal, Muhammad Rizki	Mengembangkan sistem deteksi bahasa isyarat secara real-time menggunakan TensorFlow Object Detection API dan Python. Menggunakan SSD dan deep learning.
2	Deteksi Objek Bahasa Isyarat Huruf Bisindo Menggunakan SSD MobileNet	Dwi Handayani, Rini Anggraeni, Siti Nurjanah	Menggunakan SSD MobileNet untuk mengidentifikasi huruf dalam bahasa isyarat Bisindo.
3	Sistem Deteksi Simbol pada SIBI (Sistem Isyarat Bahasa Indonesia) Secara *Realtime* Menggunakan MobileNet-SSD	Muhammad Rizki	Deteksi simbol bahasa isyarat SIBI secara real-time menggunakan MobileNet-SSD.
4	Deteksi Gambar Gestur Kosakata Bahasa Isyarat Indonesia dengan Metode Convolutional Neural Network (CNN) Berbasis SSD MobileNet	Ferdian Rachardi	Deteksi gestur kosakata bahasa isyarat Indonesia menggunakan CNN berbasis SSD MobileNet.
5	Analisis Akurasi Object Detection Menggunakan Tensorflow Untuk Pengenalan Bahasa Isyarat Tangan Menggunakan Metode SSD	Elisa Tikasni, Ema Utami, Dhani Ariatmanto	Menganalisis akurasi deteksi objek untuk pengenalan bahasa isyarat tangan menggunakan metode SSD.

2.16 Perbedaan Penelitian terdahulu dengan Proyek

Deteksi bahasa isyarat secara *realtime* menggunakan teknologi pembelajaran mesin dan pengolahan citra telah menarik perhatian dalam beberapa tahun terakhir. Berbagai penelitian sebelumnya telah mengeksplorasi metode deteksi berbasis sensor serta berbasis citra untuk mengenali gerakan tangan yang digunakan dalam bahasa isyarat. Dalam penelitian ini, digunakan model SSD MobileNet yang dioptimalkan dengan Tensorflow, yang telah terbukti efektif dalam deteksi objek pada perangkat dengan keterbatasan sumber daya.

Beberapa penelitian sebelumnya juga telah menggunakan SSD MobileNet dengan Tensorflow dalam mendeteksi bahasa isyarat. Namun, akurasi sistem deteksi tersebut sering kali bergantung pada kualitas data pelatihan yang digunakan dan kondisi pencahayaan dalam lingkungan pengujian. Sistem deteksi berbasis SSD MobileNet yang diterapkan dalam penelitian ini menunjukkan kemampuan untuk melakukan deteksi secara *realtime*. Dapat dilihat pada tabel 2. Perbedaan dengan penelitian terdahulu.

Tabel 2 Perbedaan dengan Penelitian Terdahulu

Aspek	**Penelitian Sebelumnya**	**Penelitian ini (SSD dengan Tensorflow)**
Metode Deteksi	SSD MobileNet (dengan Tensorflow)	SSD MobileNet (dengan Tensorflow)
Kecepatan Deteksi	*Realtime,* tergantung perangka	*Realtime* dengan optimasi Tensorflow
Efisiensi	Memadai pada perangkat dengan GPU	Efisien pada latar belakang kompleks dan sederhana
Kelemahan	Bergantung pada kualitas data pelatihan	Terpengaruh oleh pencahayaan dan noise pada gambar
Kelebihan	Mudah diimplementasikan, model ringan	lebih ringan, menggunakan *time stamp* dan cepat dibandingkan model lain

Bab 3. Metodologi Penelitian / Metode Pelaksanaan

Pada rancangan penelitian pembuatan tugas akhir yang berjudul "Sistem Deteksi Bahasa Isyarat secara *Realtime* menggunakan Tensorflow dengan SSD MobilNetV2". Penulis menggunakan metode SSD MobilNetV2 dikarenakan metode pengolahan citra yang jarang dilakukan. Secara keseluruhan penelitian ini menggunakan perancangan perangkat lunak.

3.1. Perangkat yang Digunakan

Pada penelitian ini, digunakan metode *Object Detection* menggunakan SSD MobileNetV2 yang dijalankan menggunakan Tensorflow. SSD MobileNetV2 dipilih karena kemampuannya untuk mendeteksi objek sambil mempertahankan kinerja yang cepat pada perangkat dengan sumber daya terbatas, seperti laptop. SSD MobileNetV2 adalah model yang ringan dan efisien, menjadikannya ideal untuk pengenalan objek real-time dalam aplikasi deteksi bahasa isyarat.

Spesifikasi Laptop yang Digunakan:

a. **Laptop**: Predator Helios 300
b. **Prosesor**: Intel Core i7
c. **RAM**: 16 GB
d. **GPU**: NVIDIA GeForce RTX 3060
e. **Sistem Operasi**: Windows 10 64-bit
f. **Perangkat Lunak**: TensorFlow, Python 3.x, OpenCV

Dengan menggunakan Predator Helios 300, yang memiliki kemampuan GPU yang mumpuni, model SSD MobileNetV2 dapat dijalankan dengan optimal, memungkinkan deteksi objek dalam bahasa isyarat secara real-time.

3.2 Pembagian Bahasa Isyarat Berdasarkan Deteksi Ekspresi Wajah

Pada penelitian ini, bahasa isyarat yang dideteksi dibagi menjadi dua kelompok berdasarkan kebutuhan ekspresi wajah:

1. **Bahasa Isyarat dengan Deteksi Ekspresi Wajah**
   Bahasa isyarat dalam kelompok ini membutuhkan ekspresi wajah sebagai pendukung untuk memberikan makna yang lebih jelas. Deteksi mencakup gerakan tangan dan analisis ekspresi wajah. Contoh bahasa isyarat yang memerlukan ekspresi wajah meliputi:
   a. Halo
   a. Tolong
   b. Terima Kasih
   c. Wah, Keren
   d. Bagaimana

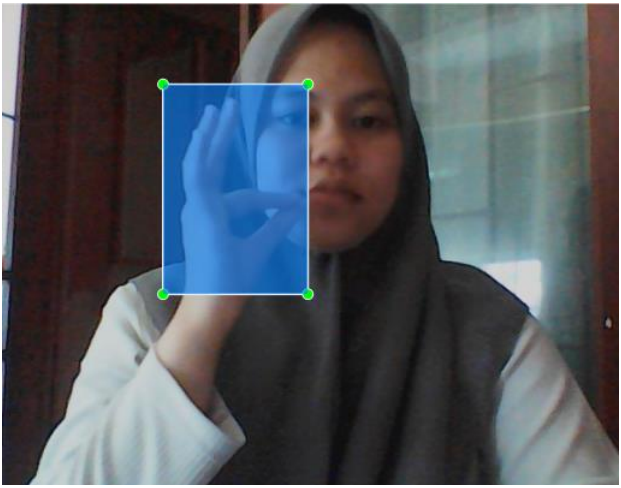
   berikut menunjukkan contoh deteksi gerakan tangan dan ekspresi wajah pada bahasa isyarat "Halo":

Gambar 3. 1 Deteksi Gerakan Tangan dan Ekspresi Wajah untuk Bahasa Isyarat "Halo"

Keterangan*: Bounding box* menandai gerakan tangan, sedangkan fitur wajah yang relevan (alis dan mulut) dianalisis untuk mendeteksi ekspresi wajah.

2. **Bahasa Isyarat Tanpa Deteksi Ekspresi Wajah**
   Bahasa isyarat dalam kelompok ini hanya bergantung pada gerakan tangan tanpa memerlukan ekspresi wajah. Deteksi dilakukan menggunakan model SSD Mobilenet. Contoh bahasa isyarat yang tidak membutuhkan ekspresi wajah meliputi:
   a. Sedih
   b. Sama-sama
   c. Bermain
   d. Maaf
   e. Ingat

   Gambar berikut menunjukkan contoh deteksi gerakan tangan pada bahasa isyarat "Maaf":



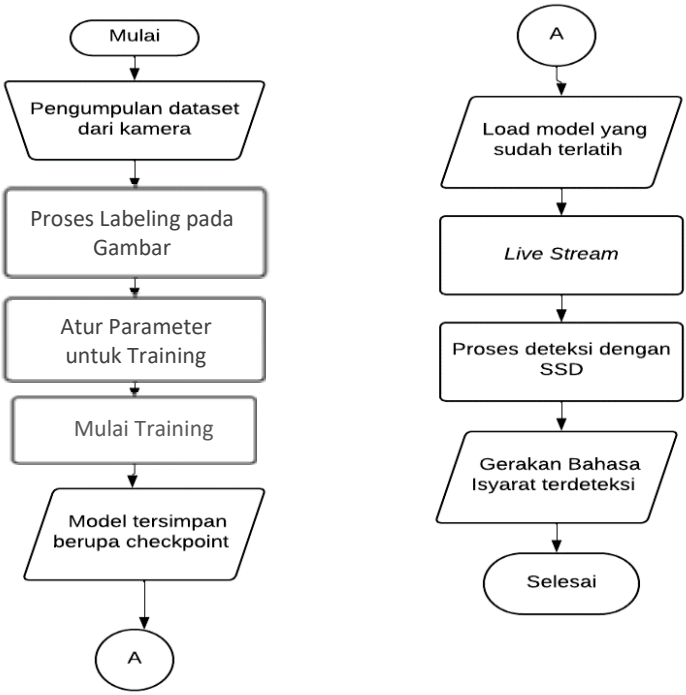
Gambar 3. 2 Deteksi Gerakan Tangan untuk Bahasa Isyarat "Maaf"

Keterangan: *Bounding box* hanya menandai posisi tangan tanpa analisis ekspresi wajah.

3.3. Perancangan Perangkat Lunak

Perancangan perangkat lunak pada penelitian ini meliputi pelatihan dan pengujian.

Tahap pertama yang dilakukan adalah dengan mengumpulkan gambar menggunakan phyton dan Opencv lalu kita akan memberi label pada object tersebut menggunakan paket gambar label, lalu yang akan kita lakukan adalah memanfaatkan kembali beberapa kode yang telah kita buat untuk pendeteksi object lain dan membuat pendeteksi bahasa isyarat menggunakan pembelajaran transfer dan API deteksi object Tensorflow, yang terakhir kita akan dapatt mendeteksi berbagai pose bahasa isyarat secara *realtime*. Proses pelatihan dan pengujian menggunakan metode SSD MobilNetV2 dapat dilihat gambar 3.3.

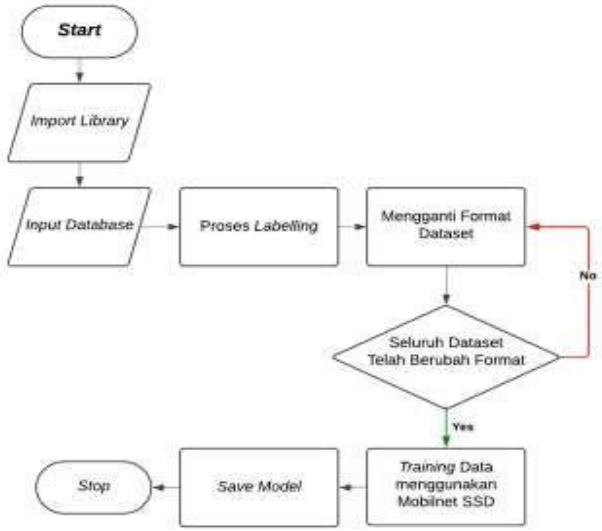


Gambar 3. 3 Flowchart Proses Pelatihan dan Pengujian

Pada gambar 3.3 menunjukkan langkah-langkah pada flowchart dimulai dari mengumpulkan dataset dari kamera, proses labeling anotasi pada gambar, atur konfigurasi parameter dari SSD mulai dari learning rate, epoch, batch size, dan step, mulai training dataset menggunakkan SSD, dan menyimpan model yang sudah terlatih. Berikutnya akan di uji pada load model yang sudah terlatih pada live stream video. Model akan mendeteksi sebanyak 10 kelas BISINDO.

3.4. *Process Training* model *Pre-Trained* SSD MobilNetV2

Pada Gambar 3.4 menunjukkan flowchart proses *Training model MobileNet SSD* yang diajukan pada Tugas Akhir ini.



Gambar 3. 4 Flowchart Proses Training SSD MobilNet

Dimulai dari Start lalu melakukan Import library yang akan digunakan, setelah itu melakukan input lokasi dataset. Dalam proses ini dilakukan labeling dengan menggunakan LabelImg untuk memberi koordinat prediksi dari citra yang akan di training berupa format XML. Karena TensorFlow API hanya menerima format tfrecord, maka format citra dari XML dikonversi menjadi format tfrecord dengan menggunakan program yang telah disediakan oleh TensorFlow sendiri. Jika masih ada format xml yang belum di konversi maka harus diubah menjadi format tfrecord terlebih dahulu, setelah itu dataset siap untuk dilakukan proses training dengan model MobileNet-SSDv2. Pada saat proses training dataset, citra dibagi kedalam folder train dan test untuk dilakukan proses training secara terpisah.

Pada saat training model berakhir training command memunculkan hasil nilai loss, learning_rate dan total step. Dengan selesainya proses raining, hasil model tersebut otomatis tersimpan berupa .ckpt file. Penggunaan model training pada objek deteksi *realtime* sering menggunakan file .ckpt dikarenakan dapat melakukan tuning lagi atau ingin melakukan eksperimen pada model jika model terlalu overtrain atau hasil akurasi masih kurang optimal.

3.5. Dataset yang digunakan

Pada sebuah model Depp Learning dibutuhkan dataset untuk melatih model dalam mengenali objek. Dataset yang digunakan pada penelitian ini berupa gambar dengan format .jpg yang terbagi menjadi 10 kelas yang terdiri dari BISINDO. Agar mendapatkan hasil akurasi baik dibutuhkan background yang mirip dengan dataset yang telah diambil.

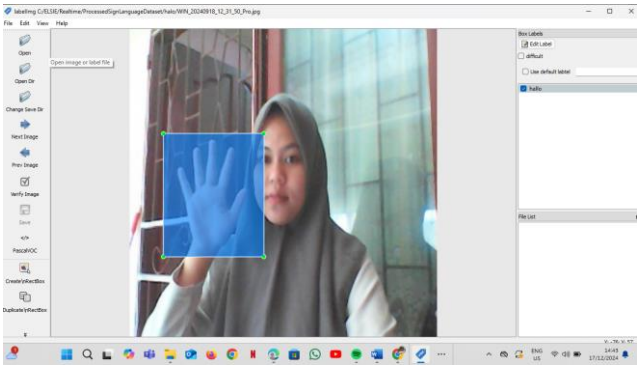
Dataset penelitian tugas akhir ini diambil secara manual menggunakan kamera webcam total data yang telah diambil sebanyak 2000 dataset, masing-masing kelas memiliki 200 dataset. Contoh dataset yang diambil untuk kelas “Ingat” dan “Halo” terlihat pada gambar 3.5.

Gambar 3. 5 Dataset "Ingat" dan "Halo"

Gambar 3.5 merupakan salah satu contoh dataset yang digunakan dalam penelitian. Dataset ini diambil untuk kelas gestur tangan "Halo" dan "Ingat" yang akan digunakan sebagai data latih dalam sistem klasifikasi bahasa isyarat secara *realtime*.

3.6. Proses Labeling Gambar

Setelah gambar dikumpulkan menggunakan kamera webcam secara manual, selanjutnya adalah labeling pada gambar. Angar model mengenali objek pada gambar harus dibuat anotasi. Terlihat pada gambar 3.6 proses labeling gambar.

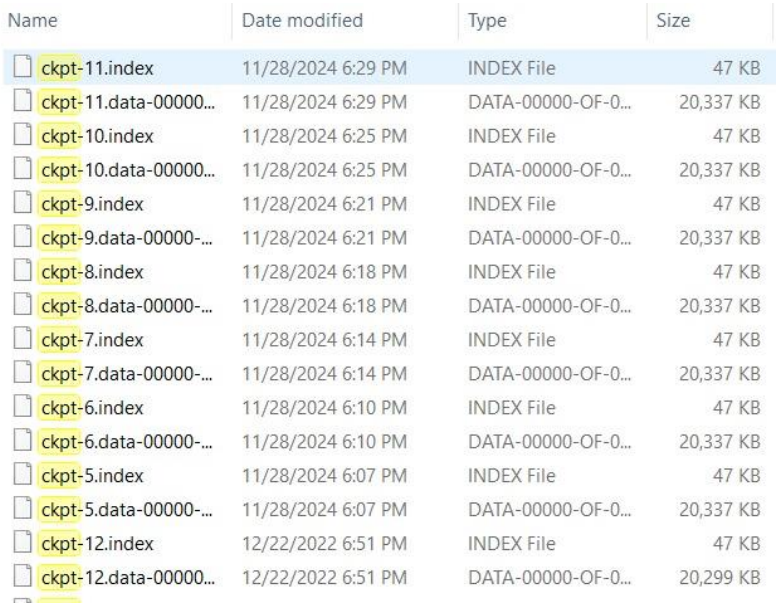


Gambar 3. 6 Proses labeling gambar

Pada tahap ini adalah cara memberi label menggunakan labelimg. Caranya adalah dengan memberi reactbox pada tangan yang memberikan gerakan bahasa isyarat. Jika gerakannya menunjukkan kata “halo” maka kelas nya diberi nama “halo” dan seterusnya sebanyak jumlah total datset yang diambil. Hasil dari labelimg adalah sebuah file yang berformat XML. Selanjutnya file tersebut akan di load ke model dengan path nya.

3.7. Training

Traing merupakan proses untuk melatih model agar mengenali objek dataset yang telah di anotasi. Proses satu kali training dengan step 10000 memakan waktu kurang lebih 1 jam. Hasail dari proses training berbentuk checkpoint untuk menyimpan hasil training. Checkpoitn yang disimpah tersebut di loat ke sistem untuk menjalankan klasifikasi objek. Terlihat pada gamabr 3.7 hasi *file checkpoint.*



Name	Date modified	Type	Size
ckpt-11.index	11/28/2024 6:29 PM	INDEX File	47 KB
ckpt-11.data-00000...	11/28/2024 6:29 PM	DATA-00000-OF-0...	20,337 KB
ckpt-10.index	11/28/2024 6:25 PM	INDEX File	47 KB
ckpt-10.data-00000...	11/28/2024 6:25 PM	DATA-00000-OF-0...	20,337 KB
ckpt-9.index	11/28/2024 6:21 PM	INDEX File	47 KB
ckpt-9.data-00000-...	11/28/2024 6:21 PM	DATA-00000-OF-0...	20,337 KB
ckpt-8.index	11/28/2024 6:18 PM	INDEX File	47 KB
ckpt-8.data-00000-...	11/28/2024 6:18 PM	DATA-00000-OF-0...	20,337 KB
ckpt-7.index	11/28/2024 6:14 PM	INDEX File	47 KB
ckpt-7.data-00000-...	11/28/2024 6:14 PM	DATA-00000-OF-0...	20,337 KB
ckpt-6.index	11/28/2024 6:10 PM	INDEX File	47 KB
ckpt-6.data-00000-...	11/28/2024 6:10 PM	DATA-00000-OF-0...	20,337 KB
ckpt-5.index	11/28/2024 6:07 PM	INDEX File	47 KB
ckpt-5.data-00000-...	11/28/2024 6:07 PM	DATA-00000-OF-0...	20,337 KB
ckpt-12.index	12/22/2022 6:51 PM	INDEX File	47 KB
ckpt-12.data-00000...	12/22/2022 6:51 PM	DATA-00000-OF-0...	20,299 KB

Gambar 3. 7 File Checkpoint

Gambar 3.7 menunjukkan file checkpoint yang dihasilkan dari proses pelatihan model. File checkpoint berisi parameter dan bobot model yang telah dioptimalkan selama pelatihan, yang nantinya digunakan untuk evaluasi dan pengujian sistem klasifikasi bahasa isyarat secara *realtime*.

Setelah file checkpoint berhasil dailanjutkan ke proses traing pada gambar yang terlihat pada gambar 3.8.

Gambar 3. 8 Salah satu Porses Training untuk Kelas "Tolong"

Gambar 3.8 merupakan salah satu hasil keluaran model setelah proses pelatihan unyuk kelas gestur “Tolong”. Hasil ini menunjukka kemampuan model dalam mengenali pola gestur yang sesuai dengan kelas yang dilatih.

3.8. Pengujian

Setelah melakukan proses pelatihan, model yang telah dilatih dievaluasi untuk mengukur kinerja dengan menggunakan data uji. Evaluasi dilakukan melalui beberapa skenario pengujian, termasuk pengujian jarak dan pengujian akurasi setiap kelas. Selain itu, dilakukan pengujian tambahan secara real-time menggunakan empat orang subjek dengan karakteristik berbeda untuk memastikan model mampu bekerja di kondisi nyata.

3.8.1. Pengujian Jarak

Pada skenario pengujian ini berfokus untuk mencari jarak optimal dalam mendeteksi gerakan bahasa isyarat. Masing-masing nilai jarak yang akan diujikan adalah 0cm, 25cm, 50cm, 100cm, dan 150cm. Nilai yang dapat dihitung pada pengujian ini adalah akurasi dengan menghitung banyaknya kelas yang terdeteksi pada masing-masing jarak.

A. Metode Pengujian Jarak

Pengujian dilakukan secara :

1. Data Input: Model menerima gambar dari gerakan tangan sebagai input.
2. Jarak Pengujian: Setiap pengujian dilakukan pada jarak 0cm, 25cm, 50cm, 100cm, dan 150cm, yang mempengaruhi kualitas citra yang diterima oleh model.
3. Evaluasi Model: Model klasifikasi akan memberikan hasil prediksi berdasarkan input gambar. Hasil evaluasi akan dihitung dalam bentuk True Positive (TP), False Positive (FP), dan False Negative (FN).
4. Menghitung Akurasi setiap kelas: Untuk setiap kelas, perlu mengetahui berapa banyak data yang terdeteksi dengan benar (TP) oleh model pada jarak tertentu, dibandingkan dengan jumlah total data aktual pada kelas tersebut (yaitu 50 data uji untuk setiap kelas).
5. Jumlah data uji : Jumlah data uji yanag digunakan sebanyak 50 data uji.

Langkah 1. Menghitung Akurasi untuk Setiap Kela Sesuai Jarak tertentu:

Rumus 1 :

$$\boldsymbol{Akurasi\ kelas}\ (\%) = (\boldsymbol{TP} \mid \boldsymbol{Aktual}) \times \mathbf{100}$$

Dimana :

a. TP adalah jumlah data yang terdeteksi dengan benar untuk kelas tersebut pada jarak tertentu.
b. Aktual adalah jumlah total data uji untuk kelas tersebut, yang dalam hal ini adalah 50 data uji per setiap kelas.

Langkah 2: Menghitung Akurasi Keseluruhan untuk Setiap jarak

Setalah menghitung akurasi untuk setiap kelas, rata-rata akurasi untuk setiap jarak dihitung dengan rumus:

Rumus 2 :

$$\boldsymbol{Akurasi\ Total}\ (\%) = (\sum \boldsymbol{Akurasi\ untuk\ setiap\ kelas} \mid \mathbf{10})$$

Karena ada 10 kelas, kita menjumlahkan akurasi setiap kelas dan membaginya dengan 10.

3.8.2. Pengujian Akurasi Setiap Kelas

Accuracy merupakan data seberapa akurat sistem mengklasifikasi dengan benar. ujuan dari pengujian ini adalah untuk mengukur **akurasi** deteksi gerakan bahasa isyarat menggunakan model klasifikasi. Pengukuran dilakukan dengan menghitung **True Positive (TP)**, **False Positive (FP)**, dan **False Negative (FN)** untuk setiap kelas yang diuji. Dari nilai-nilai ini, akurasi model akan dihitung untuk menilai kinerja model dalam mendeteksi bahasa isyarat yang tepat.

A. Metode Pengujian Akurasi Setiap kelas

Pada pengujian ini, model klasifikasi akan menerima input berupa gambar gerakan tangan yang mewakili bahasa isyarat. Model kemudian akan menghasilkan output berupa kelas prediksi. Untuk mengevaluasi hasil, kami akan menghitung TP, FP, DAN FN untuk setiap kelas berdasarkan perbandingan antar hasil prediksi dan label aktual.

a. **True Positive (TP)**: Jumlah sampel yang benar-benar terdeteksi dengan kelas yang tepat oleh model.
b. **False Positive (FP)**: Jumlah sampel yang tidak termasuk dalam kelas tertentu tetapi salah diklasifikasikan oleh model sebagai kelas tersebut.
c. **False Negative (FN)**: Jumlah sampel yang seharusnya diklasifikasikan dalam kelas tertentu, tetapi tidak dikenali oleh model.

Akurasi dihitung dengan rumus:

$$\boldsymbol{Akurasi}\ (\%) = \ (\boldsymbol{TP} \mid \boldsymbol{TP} + \ \boldsymbol{FP} \ + \boldsymbol{FN}) \ \times \mathbf{100}$$

Langkah-langkah untuk Menghitung Akurasi:

1. Jumlahkan TP, FP, dan FN untuk setiap kelas.
2. Masukkan nilai TP, FP, dan FN ke dalam rumus akurasi.
3. Hitung hasilnya untuk mendapatkan akurasi dalam persentase.

3.8.3 Pengujian Secara *Offline*

Pada tahap ini, dilakukan pengujian sistem klasifikasi bahasa isyarat secara offline menggunakan dataset yang telah dibagi menjadi dua bagian utama, yaitu data pelatihan *(train)* dan data pengujian (*test).* Sebanyak 80% data digunakan untuk pelatihan model, sementara 20% sisanya digunakan untuk pengujian akurasi. Pengujian *offline* ini bertujuan untuk mengevaluasi seberapa baik sistem dapat mengklasifikasikan bahasa isyarat dengan menggunakan data yang telah dilatih sebelumnya tanpa adanya interaksi langsung dengan input secara real-time.

Proses pengujian dimulai dengan mempersiapkan data pengujian yang sudah dipisahkan. Model yang telah dilatih sebelumnya menggunakan Tensorflow dengan SSD MobileNet kemudian diuji terhadap data pengujian untuk menilai kinerjanya dalam mengklasifikasikan 10 kelas bahasa isyarat yang berbeda. Hasil pengujian dihitung berdasarkan parameter seperti ***True Positive (TP), False Positive (FP),*** dan ***False Negative (FN)*** untuk masing-masing kelas bahasa isyarat. Setelah pengujian, hasil akurasi dihitung untuk masing-masing kelas berdasarkan banyaknya prediksi yang benar, dan kesalahan yang terjadi pada tiap kelas juga dianalisis.

Pada bagian ini, kami juga akan membahas hasil dan interpretasi dari pengujian *offline* yang dilakukan, serta membandingkan hasil yang diperoleh dengan ekspektasi yang telah ditetapkan sebelumnya.

Pengujian ini memberikan gambaran awal tentang seberapa efektif model klasifikasi bahasa isyarat yang dibangun dalam mengenali gerakan tangan dan menerjemahkannya menjadi teks. Selain itu, pengujian *offline* ini juga bertujuan untuk mengidentifikasi potensi masalah yang dapat muncul dalam proses klasifikasi sebelum implementasi sistem secara real-time.

3.8.3 Penguian secara *Realtime*

A. Pengujian Secara Realtime dengan Subjek Penulis

Pengujian dilakukan dengan menggunakan data uji yang dikumpulkan secara realtime oleh penulis. Setiap kelas dalam sistem klasifikasi bahasa isyarat diuji menggunakan 50 sampel data uji. Langkah-langkah pengujian adalah sebagai berikut:

1. **Persiapan Pengujian**
   a. Sistem telah dikembangkan menggunakan model SSD Mobilenet dengan TensorFlow dan diintegrasikan ke dalam GUI desktop.
   b. Data uji terdiri dari gestur tangan yang direkam langsung melalui kamera secara realtime.
   c. Lingkungan pengujian memiliki pencahayaan stabil dan tidak ada gangguan latar belakang yang signifikan.
2. **Proses Pengujian**
   a. Penulis melakukan setiap kelas gestur sebanyak 50 kali.
   b. Data hasil pengujian direkam secara otomatis oleh sistem, termasuk waktu respon dan hasil klasifikasi.

B. Pengujian Secara Realtime dengan 4 Subjek Orang Lain

Pengujian ini dilakukan untuk menguji robustness sistem terhadap variasi pengguna lain. Empat orang dengan latar belakang berbeda dipilih sebagai subjek, masing-masing melakukan pengujian untuk setiap kelas sebanyak 10 kali.

1. **Persiapan Pengujian**
   a. Subjek diberikan panduan singkat mengenai cara melakukan gestur bahasa isyarat dengan benar.
   b. Kamera dan sistem diuji sebelumnya untuk memastikan stabilitas selama pengujian.
   c. Lingkungan pengujian disesuaikan untuk meminimalkan gangguan eksternal.
2. **Proses Pengujian**
   a. Setiap subjek diminta melakukan setiap kelas bahasa isyarat sebanyak 10 kali.
   b. Data hasil pengujian dari keempat subjek digabungkan untuk analisis keseluruhan.
3. **Subjek Pengujian:**
   a. Subjek 1: Lingga
   b. Subjek 2: Gladys
   c. Subjek 3: Isa
   d. Subjek 4: Azifah

4. **Metode Pengujian:**
    a. **Jumlah Kelas Bahasa Isyarat:** 10 gerakan.
    b. **Jumlah Pengulangan:** Setiap subjek melakukan setiap gerakan aktual data 10.
    c. **Total Data Uji:**
        a) Per subjek: $10\ kelas\ \times 10\ data = 100\ data\ uji$
        b) Total 4 subjek: $100\ \times 4\ = 400\ data\ uji$

3.9 Proses Deteksi dan Output Sistem

Sistem deteksi bahasa isyarat ini dirancang untuk mendeteksi gerakan tangan secara real-time dengan menggunakan *bounding box*. Proses deteksi dilakukan dengan langkah-langkah berikut:

1. Model SSD MobileNet membaca frame video dari kamera.
2. *Bounding box* dihasilkan untuk menandai area yang terdeteksi sebagai bahasa isyarat.
3. *Timestamp* dicantumkan pada setiap bounding box untuk mencatat waktu deteksi pada setiap frame.

Contoh output sistem dengan timestamp pada bounding box, terlihat pada gambar 3.9:

Gambar 3. 9 Hasil *Output* dengan *Timestamp*

- *Bounding box* menampilkan bahasa isyarat dengan label "Halo" pada timestamp 93.99 ms.
- Deteksi diulang pada setiap frame untuk memastikan bahwa sistem dapat bekerja secara real-time.

Proses ini diuji pada perangkat Predator Helios 300 dengan spesifikasi tertentu, menggunakan Tensorflow sebagai platform utama untuk pengolahan data.

Bab 4. Hasil dan Pembahasan

4.1. Hasil Pengujian Berdasarkan Jarak

Pada pengujian jarak, dilakukan percobaan pada 5 jarak berbeda: 0cm, 25cm, 50cm, 100cm, dan 150cm. setiap jarak diuji dengan 50 percobaan untuk memastikan konsistensi hasil. Akurasi dihitung berdasarkan banyaknya kelas yang berhasil terdeteksi dengna benar.

Berikut adalah tabel 1. yang sudah mencakup akurasi per kelas pada pengujian jarak, serta tabel 2. Yang berisi jarak, aktual, prediksi, dan akurasi secara keseluruhan. Tabel 1 dan tabel 2. menunjukkan hasil pengujian sistem pada 5 jarak berbeda: 0cm, 25cm, 50cm, 100cm, dan 150cm. Setiap kelas memiliki data aktual (jumlah data sebenarnya) dan prediksi (jumlah data yang dikenali oleh sistem). Akurasi dihitung untuk menilai seberapa baik sistem mengenali data, dengan persentase akurasi yang lebih tinggi menunjukkan performa yang lebih baik. Pada jarak tertentu, beberapa kelas mencapai akurasi 100%, sementara di jarak lain, ada yang memiliki akurasi rendah hingga 0%. Terlihat pada tabel 1.

Tabel 3 Hasil Pengujian Jarak tertentu Setiap Kelas

Kelas Bahasa Isyarat	0 cm (Akurasi%)	25 cm (Akurasi%)	50 cm (Akurasi%)	100 cm (Akurasi%)	150 cm (Akurasi%)
`Halo	0% (0)	50% (25)	100% (50)	76% (38)	0% (0)
Maaf	0% (0)	40% (20)	80% (40)	76% (38)	0% (0)
Tolong	0% (0)	44% (22)	90% (45)	70% (35)	0% (0)
Terima Kasih	0% (0)	50% (25)	100% (50)	76% (38)	0% (0)
Sama-sama	0% (0)	42% (21)	86% (43)	76% (38)	0% (0)
Bermain	0% (0)	40% (20)	80% (40)	60% (30)	0% (0)
Ingat!	0% (0)	40% (20)	80% (40)	70% (35)	0% (0)
Wah Keren	0% (0)	56% (28)	100% (50)	80% (40)	0% (0)
Bagaimana	0% (0)	50% (25)	96% (48)	80% (40)	0% (0)
Sedih	0% (0)	46% (23)	85% (45)	74% (37)	0% (0)

Tabel ini menunjukkan akurasi deteksi bahasa isyarat pada berbagai jarak (0 cm, 25 cm, 50 cm, 100 cm, dan 150 cm) untuk setiap kelas bahasa isyarat yang diuji. Kolom pertama mencantumkan kelas bahasa isyarat yang diuji, sementara kolom-kolom berikutnya menunjukkan akurasi deteksi pada jarak tertentu, diikuti dengan jumlah prediksi yang benar dalam tanda kurung.

Dari hasil yang terlihat pada tabel, dapat dilihat bahwa akurasi deteksi mengalami penurunan pada jarak yang lebih jauh (terutama pada jarak 150 cm), dengan beberapa kelas bahasa isyarat seperti "Hallo", "Maaf", "Tolong", dan

lainnya menunjukkan akurasi yang rendah pada jarak 150 cm. Sebaliknya, pada jarak 50 cm, akurasi deteksi mencapai 100% untuk beberapa kelas, menunjukkan bahwa jarak yang lebih dekat memungkinkan sistem deteksi untuk bekerja dengan lebih efektif.

Hasil ini memberikan gambaran tentang kinerja sistem deteksi bahasa isyarat pada jarak yang berbeda-beda, serta pentingnya mempertimbangkan jarak dalam merancang sistem pengenalan gerakan yang lebih efisien dan akurat.

Setelah mendapatkan hasil akurasi setiap kelas, kita dapat menguji jarak secara keseluruhan pada tabel 2.

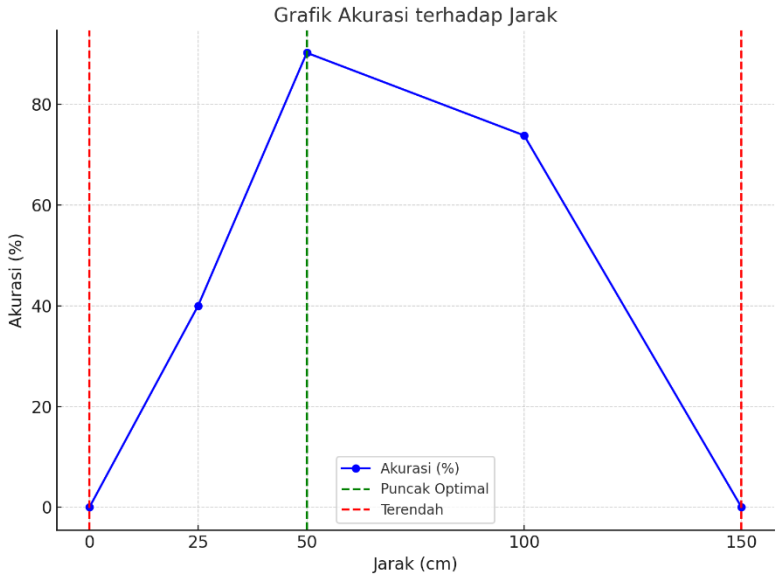
Tabel 4 Hasil Pengujian Jarak

Jarak	Aktual	Terprediksi	Akurasi
0cm	500	0	0%
25cm	500	200	40%
50cm	500	451	90.2%
100cm	500	369	73.8%
150cm	500	0	0%

Dari hasil pengujian akurasi berdasarkan jarak yang tercantum pada tabel 2, menunjukkan hasil pengujian sistem deteksi bahasa isyarat berdasarkan jarak yang diuji, yaitu 0 cm, 25 cm, 50 cm, 100 cm, dan 150 cm. Untuk setiap jarak, tercatat jumlah aktual (500 data) dan jumlah yang terprediksi dengan benar, beserta akurasinya.

a. Pada jarak 0 cm, sistem tidak berhasil memprediksi dengan benar, menghasilkan akurasi 0%.
b. Pada 25 cm, akurasi meningkat menjadi 40%, dengan 200 data terprediksi benar.
c. Pada 50 cm, akurasi mencapai 90.2% dengan 451 data terprediksi dengan benar, menunjukkan hasil terbaik.
d. Pada 100 cm, akurasi menurun menjadi 73.8%, dengan 369 data terprediksi dengan benar.
e. Pada 150 cm, tidak ada data yang terprediksi dengan benar, menghasilkan akurasi 0%.

Secara keseluruhan, jumlah data yang terprediksi dengan benar dari total 2500 data yang diuji adalah 1020, menunjukkan bahwa jarak yang lebih dekat (terutama 50 cm) memberikan hasil yang lebih akurat dalam sistem deteksi bahasa isyarat ini. Grafik pada Gambar 4.1 di bawah ini akan memperlihatkan dengan lebih jelas hubungan antara jarak dan akurasi sistem deteksi. Grafik ini menggambarkan perubahan akurasi pada berbagai jarak, yang memberikan gambaran lebih mendalam mengenai performa sistem dalam mendeteksi bahasa isyarat pada jarak yang berbeda.



Gambar 4. 1 Perubahan Akurasi pada Berbagai Jarak

Berdasarkan hasil pengujian jarak pada sistem dalam mendeteksi objek pada jarak yang berbeda, Grafik menunjukkan hubungan antara akurasi pengenalan bahasa isyarat dengan jarak kamera terhadap objek:

1. Akurasi Terendah (0%) terjadi pada jarak 0 cm dan 150 cm. Hal ini disebabkan oleh keterbatasan ruang pada jarak 0 cm dan hilangnya detail visual pada jarak yang terlalu jauh (150 cm).
2. Akurasi Optimal (90.2%) tercapai pada jarak 50 cm. Ini menunjukkan jarak ini adalah kondisi terbaik untuk sistem dalam menangkap dan mengenali pola bahasa isyarat secara akurat.
3. Akurasi mulai menurun pada jarak 100 cm (73.8%), mengindikasikan mulai berkurangnya kualitas pengenalan karena detail objek berkurang.

Grafik ini memberikan wawasan bahwa jarak 50 cm adalah jarak ideal untuk implementasi sistem pengenalan bahasa isyarat secara *realtime*.

4.2 Hasi Pengujian Akurasi secara Offline dan *Realtime*

Pada bagian ini, akan dijelaskan hasil pengujian akurasai yang akan dilakukan pada dua kondisi berbeda: Offline dan *Realtime*. Pengujian dilakukan pada dua bagian dataset, yaitu data train (80%) dan data pengujian (20%). Hasil pengujian akan dibahas secara terpisah untuk masing-masing metode.

4.2.1 Hasil Pengujian Akurasi *Offline*

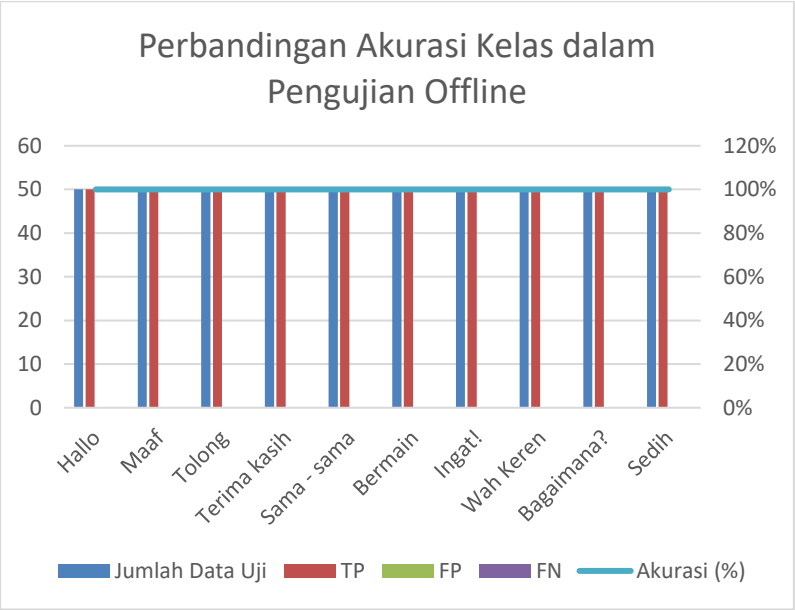
Pada Pengujian Offline, data yang digunakan terbagi menjadi dua kategori: data pelatihan (train) dan data pengujian (test). 80% data digunakan untuk pelatihan dan 20% untuk pengujian. Hasil pengujian ini tidak melibatkan data yang dikumpulkan secara *realtime*.

Tabel 3 di bawah ini menunjukkan hasil pengujian akurasi secara offline berdasarkan jumlah data uji TP (True Possitive), FP (False Positive), dan FN (False Negative), dan nilai akurasi yang dihitung darihasil tersebut.

Tabel 5 Hasil Pengujian Akurasi secara Offline

Class	Jumlah Data Uji	TP	FP	FN	Akurasi (%)
Hallo	50	50	0	0	100%
Maaf	50	50	0	0	100%
Tolong	50	50	0	0	100%
Terima kasih	50	50	0	0	100%
Sama - sama	50	50	0	0	100%
Bermain	50	50	0	0	100%
Ingat!	50	50	0	0	100%
Wah Keren	50	50	0	0	100%
Bagaimana?	50	50	0	0	100%
Sedih	50	50	0	0	100%

Dari tabel di atas, terlihat bahwa setiap kelas memiliki akurasi 100%, dengan semua data uji berhasil dikenali dengan benar tanpa adanya kesalahan. Grafik berikut akan memvisualisasikan performa model untuk mempermudah analisis



Gambar 4. 2 Perbandingan Akurasi Kelas dalam Pengujian *Offline*

Berdasarkan hasil pengujian pada tabel 3 dan grafik batang diatas, dapat dilihat bahwa akurasi deteksi bahasa isyarat secara offline mendapatkan akurasi yang optimal yang dimana masing-masing kelas mendapatkan akurasi 100%.

4.2.2. Hasil Pengujian Akurasi *Realtime*

Ada pengujian *Realtime*, data yang digunakan diambil secara langsung dan diuji pada kondisi nyata, yang menghasilkan pengukuran akurasi berdasarkan pengujia langsung. Pengujian ini dilakukan pada data pengujian yang tersedia secara *realtime* dan akurasi dihitung untuk setiap kelas berdasarkan jumlah TP,FP,FN, dan nilai akurasi yang terukur.

A. Pengujian Akurasi *RealTime* dengan Subjek Penulis

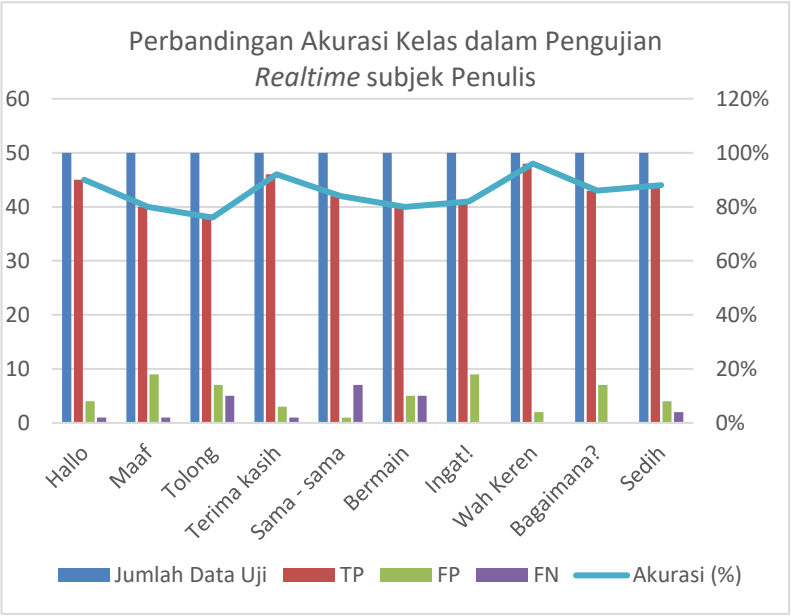
Uji akurasi secara real-time dilakukan dengan penulis sebagai subjek untuk mengevaluasi kinerja sistem dalam mengenali gestur secara langsung.

Pengujian diambil dengan masing-masing data uji setiap kelas yaitu 50 data uji. Terlihat hasil data uji secara realtime pada tabel 4 di bawah ini menunjukkan hasil pengujian akurasi secara *realtime* untuk masung-masing kelas yang subjeknya dari penulis.

Tabel 6 Hasil Pengujian Akurasi secara *Realtime* dengan subjek penulis

Class	Jumlah Data Uji	TP	FP	FN	Akurasi (%)
Hallo	50	45	4	1	90%
Maaf	50	40	9	1	80%
Tolong	50	38	7	5	76%
Terima kasih	50	46	3	1	92%
Sama - sama	50	42	1	7	84%
Bermain	50	40	5	5	80%
Ingat!	50	41	9	0	82%
Wah Keren	50	48	2	0	96%
Bagaimana?	50	43	7	0	86%
Sedih	50	44	4	2	88%

Dari hasil pengujian akurasi secara *realtime* yang tercantum pada Tabel 4, pengujian diambil oleh penulis sendiri. berikut ini adalah grafik batang yang menggambarkan perbandingan akurasi masing-masing kelas. Grafik ini memberikan gambaran visual tentang performa model dalam mengklasifikasikan berbagai gerakan bahasa isyarat.



Gambar 4. 3 Perbandingan Akurasi Kelas dalam Pengujian *Realtime subjek penulis*

Berdasarkan hasil pengujian pada tabel 4 dan grafik batang diatas, dapat dilihat bahwa akurasi deteksi bahasa isyarat bervariasi antara kelas-kelas yang diuji. Kelas dengan akurasi tertinggi adalah “Wah Keren” dengan 96%, diikuti oleh kelas “Halo” yang mencapai 90% kelas “Bagaimana?” yang mencapai 86%. Sebaliknya, kelas “Tolong” memiliki akurasi terendah, yaitu 70%.

B. Pengujian Akurasi per Kelas untuk Setiap 4 orang Subjek

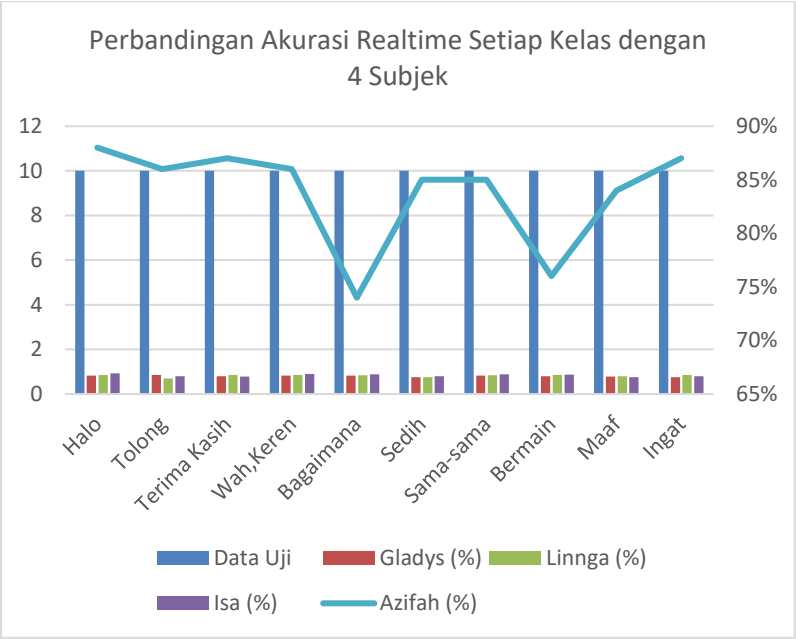
Pengujian akurasi dilakukan untuk mendeteksi 10 kelas bahasa isyarat. Pengujian ini melibatkan 4 subjek, yaitu Gladys, Lingga, Isa, dan Azifah. Setiap subjek melakukan gerakan bahasa isyarat untuk masing-masing kelas, dan hasil akurasi dicatat dalam bentuk persentase keberhasilan model mendeteksi gerakan dengan benar. Berikut adalah tabel 7 hasil pengujian akurasi untuk setiap subjek:

Tabel 7 Pengujian Akurasi secara *Realtime* untuk 4 subjek

Kelas Bahasa Isyarat	**Data Uji**	Gladys (%)	Linnga (%)	Isa (%)	Azifah (%)
Halo	10	82%	85%	92%	88%
Tolong	10	85%	70%	80%	86%
Terima Kasih	10	80%	86%	78%	87%
Wah,Keren	10	82%	85%	90%	86%
Bagaimana	10	83%	84%	89%	74%
Sedih	10	76%	75%	80%	85%
Sama-sama	10	82%	84%	89%	85%
Bermain	10	80%	85%	87%	76%
Maaf	10	78%	80%	75%	84%
Ingat	10	76%	86%	80%	87%

Tabel 7 di atas menunjukkan hasil uji deteksi bahasa isyarat yang dilakukan pada empat individu yang menggunakan sistem pengenalan bahasa isyarat secara *realtime*. Setiap individu menguji sepuluh kata bahasa isyarat dengan hasil akurasi yang bervariasi. Angka-angka pada tabel mencerminkan persentase akurasi deteksi untuk setiap kata yang diuji oleh setiap peserta.

Hasil menunjukkan bahwa ada variasi dalam akurasi deteksi antara peserta, dengan beberapa kata yang lebih mudah dikenali oleh sistem dibandingkan dengan yang lainnya. Kata-kata seperti "Halo", "Wah, Keren", dan "Sama-sama" memiliki tingkat akurasi yang cukup tinggi di semua peserta, sementara kata seperti "Bagaimana" dan "Sedih" menunjukkan tingkat akurasi yang lebih rendah pada beberapa individu. Bisa dilihat dari grafik pada gambar dibawah ini untuk hasil lebih jelas.



Gambar 4. 4 Perbandingan Akurasi Realtime Setiap Kelas dengan 4 Subjek

Grafik di atas memberikan visualisasi yang lebih jelas mengenai tingkat akurasi masing-masing peserta dalam mendeteksi setiap kelas bahasa isyarat. Dari grafik tersebut, dapat dilihat pola performa dari setiap individu terhadap berbagai kelas yang diuji. Grafik ini membantu untuk lebih mudah mengidentifikasi kelas bahasa isyarat mana yang memiliki tingkat akurasi tinggi maupun rendah. Selain itu, grafik ini juga menunjukkan bahwa beberapa gestur tertentu dapat dikenali dengan baik oleh semua peserta, sedangkan gestur lainnya masih memerlukan perbaikan untuk meningkatkan akurasi deteksi.

Bab 5. Kesimpulan dan Saran

5.1. Kesimpulan

Berdasarkan hasil pengujian, dapat disimpulkan bahwa:

1. **Pengaruh Jarak:** Jarak mempengaruhi akurasi deteksi. Jarak 50 cm memberikan hasil terbaik dengan akurasi 90.2%, sementara pada jarak 0 cm dan 150 cm, akurasi sangat rendah (0%).
2. **Offline vs Realtime:** Pengujian offline menunjukkan akurasi 100% untuk setiap kelas, sementara pengujian real-time menunjukkan variasi hasil, dengan akurasi tertinggi mencapai 96% pada kelas "Wah Keren".
3. **Variasi Antar Individu:** Akurasi deteksi bervariasi antar individu, dengan beberapa kelas lebih mudah dikenali dibandingkan yang lain.
4. **Implikasi Pengembangan:** Untuk meningkatkan akurasi, sistem perlu disesuaikan dengan jarak optimal (50 cm) dan melibatkan variasi gerakan dari berbagai individu.

5.2. Saran

Berdasarkan hasil penelitian, berikut beberapa saran yang dapat diberikan:

1. **Peningkatan Akurasi pada Jarak Jauh:** Perlu dilakukan pengembangan untuk meningkatkan akurasi deteksi pada jarak yang lebih jauh (di atas 50 cm), agar sistem dapat berfungsi secara optimal pada berbagai kondisi.
2. **Penyesuaian Model untuk Variasi Gerakan:** Sistem perlu dilatih dengan data dari berbagai individu untuk mengakomodasi perbedaan dalam gerakan bahasa isyarat, guna meningkatkan akurasi deteksi secara keseluruhan.
3. **Peningkatan Kecepatan Proses *Realtime*:** Untuk meningkatkan performa dalam pengujian real-time, sistem dapat dioptimalkan untuk memproses gerakan dengan lebih cepat dan akurat, terutama dalam kondisi dinamis.
4. **Uji Coba pada Berbagai Kondisi Lingkungan:** Pengujian lebih lanjut perlu dilakukan dalam berbagai kondisi pencahayaan dan latar belakang untuk memastikan ketahanan sistem dalam situasi dunia nyata.

Daftar Pustaka

[1] J. Purba, Pengelolaan Lingkungan Sosial. Jakarta: Kantor Menteri Negara Lingkungan Hidup, 2005.

[2] L. Alo, Komunikasi Antar Pribadi, Bandung: Citra Aditya Bakti, 1991.

[3] Lum, K. Y., Goh, Y. H., & Lee, Y. B. (2020). American sign language recognition based on MobileNetV2. Advances in Science, Technology and Engineering Systems Journal, 5(6), 481-488.

[4] S. Hayani, M. Benaddy, O. El Meslouhi, and M. Kardouchi, "Arab Sign language Recognition with Convolutional Neural Networks," Proc. 2019 Int. Conf. Comput. Sci. Renew. Energies, ICCSRE 2019, pp. 1–4, 2019, doi: 10.1109/ICCSRE.2019.8807586.

[5] M. A. Hossen, A. Govindaiah, S. Sultana, and A. Bhuiyan, "Bengali sign language recognition using deep convolutional neural network," 2018 Jt. 7th Int. Conf. Informatics, Electron. Vis. 2nd Int. Conf. Imaging, Vis. Pattern Recognition, ICIEV-IVPR 2018, pp. 369–373, 2019, doi: 10.1109/ICIEV.2018.8640962.

[6] M. A. Hossen, A. Govindaiah, S. Sultana, and A. Bhuiyan, "Bengali sign language recognition using deep convolutional neural network," 2018 Jt. 7th Int. Conf. Informatics, Electron. Vis. 2nd Int. Conf. Imaging, Vis. Pattern Recognition, ICIEV-IVPR 2018, pp. 369–373, 2019, doi: 10.1109/ICIEV.2018.8640962.

[7] I. P. W. Merta, I. M. G. Sunarya, and I. K. R. Arthana, "Handgesture To Text Dengan Metode Artificial Intelligence KNN (K-Nearest Neighbour)," Kumpul. Artik. Mhs. Pendidik. Tek. Inform., vol. 4, no. 1, pp. 18–27, 2015.

[8] Annisaa'F, N., Soekirno, S., & Aminah, S. (2021, October). Implementation of Single Shot Detector (SSD) MobileNet V2 on Disabled Patient's Hand Gesture Recognition as a Notification System. In 2021 International Conference on Advanced Computer Science and Information Systems (ICACSIS) (pp. 1-6). IEEE.

[9] K."bahasa isyarat," 2016. [Online]. Available: https://kbbi.kemdikbud.go.id/entri/bahasa%20isyarat. [Accessed 8 Maret 2024].

[10] Amrizal, V., & Aini, Q. (2013). Kecerdasan Buatan (Q. Aini, Ed.). Jakarta Halaman Moeka Publishing.

[11] Mohri, M. Rostamizadeh, A.. & Talwalkar. A. (2012). Foundation of Machine Learning. In MIT Press. https://doi.org/10.1007/978-3-642-34106-9_15

[12] E. D. Sandi, "Hari Bahasa Isyarat Internasional, Ini Ragam Bahasa Isyarat Berbagai Negara," 23 September 2020. [Online]. Available:

https://www.kompas.com/edu/read/2020/09/23/102739671/haribahAsa-isyarat-internasional-ini-ragam-bahasa-isyarat
berbagainegara?page=all. [Accessed 14 Maret 2024]
[13] E. Naqa and M. J. Murphy, "What Is Machine Learning?," in Machine Learning in Radiation Oncology, Switzerland, Springer International Publishing, 2015, pp. 3-11.
[14] A. Panesar, Machine Learning and AI for Healthcare, Coventry: Apress, 2021.
[15] dicoding, "Apa Itu Kecerdasan Buatan? Berikut Pengertian dan Contohnya," 15 Juli 2020. [Online]. Available: https://www.dicoding.com/blog/kecerdasan-buatan-adalah/. [Accessed 8 Maret 2024].
[16] M. Z. B. C. D. K. W. W. T. W. M. A. Andrew G. Howard, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision," Google Inc., 2017.
[17] "Platform pembelajaran mesin sumber terbuka ujung ke ujung," [Online]. Available: https://www.tensorflow.org/. [Accessed 14 Desember 2021].
[18] Aljabar, A., & Suharjito. (2020). BISINDO (Bahasa Isyarat Indonesia) Sign Language Recognition Using CNN and LSTM. ASTESJ.
[19] A. G. Howard et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv preprint arXiv:1704.04861, 2017.
[20] W. Liu et al., "SSD: Single Shot MultiBox Detector," in European Conference on Computer Vision (ECCV), 2016, pp. 21-37.
[21] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779-788.
[22] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137- 1149, June 2017.
[23] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2818-2826.
[24] S. Ren, K. He, R. Girshick, dan J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," di Advances in Neural Information Processing Systems (NIPS), 2015.
[25] Deng, L. and Yu, D. (2013) Deep Learning: Methods and Applications. Foundations and Trends? in Signal Processing, 7, 197-387.

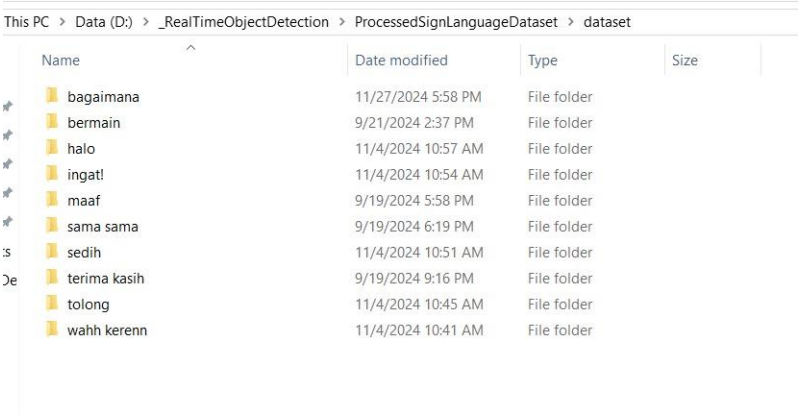
[26] H.D. Wehle, (2017). Machine Learning, Deep Learning, and AI: What's the Difference?
[27] Han, J.,&Kamber, M. (2006).Data Mining Concept and Tehniques.San Fransisco: Morgan Kauffman.
[28] Rahmatullah. H. R., Amrizal, V. & Rozy, N. F. (2018). Klasifikasi Jenis Golok Betawi dengan Naive Bayes Classifier.
[29] Gong. S. Liu, C., Ji, Y., Zhong, B., Li, Y. & Dong, H. (2019). Visual object) recognition. In Modeling and Optimization in Science and Technologies.
[30] Michelluci. U., & Moolayil, J. (2019). Advanced Applied Deep Learning.
[31] Vasilev, L., Slater. D.. Spacagna, G., Roelants, P., & Zocca. V. (2019) Python Deep Learning. Exploring Deep Learning Techniques and Neural Network.
[32] Google Research. (2018, April 16). *MobileNetV2: The Next Generation of On-Device Computer Vision Networks*. Retrieved December 24, 2024.
[33] Aljabar, A., & Suharjito. (2020). BISINDO (Bahasa Isyarat Indonesia) Sign Language Recognition Using CNN and LSTM. ASTESJ.
[34] Handayani, D., Anggraeni, R., & Nurjanah, S. (n.d.). *Deteksi Objek Bahasa Isyarat Huruf Bisindo Menggunakan SSD MobileNet*. Retrieved from pkm.tunasbangsa.ac.id
[35] Rizki, M. (n.d.). *Sistem Deteksi Simbol pada SIBI (Sistem Isyarat Bahasa Indonesia) Secara Realtime Menggunakan MobileNet-SSD*. Retrieved from repository.dinamika.ac.id
[36] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779-788.
[37] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137- 1149, June 2017.
[38] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2818-2826.
[39] "Platform pembelajaran mesin sumber terbuka ujung ke ujung," [Online]. Available: https://www.tensorflow.org/. [Accessed 14 Desember 2021].

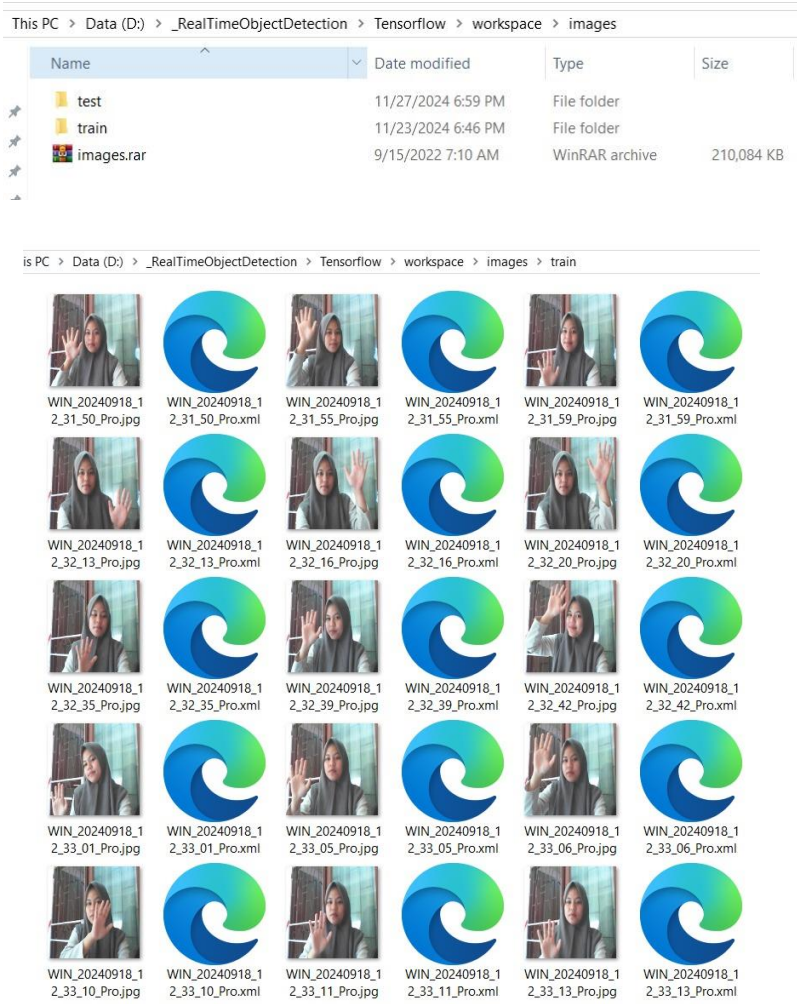
Biodata

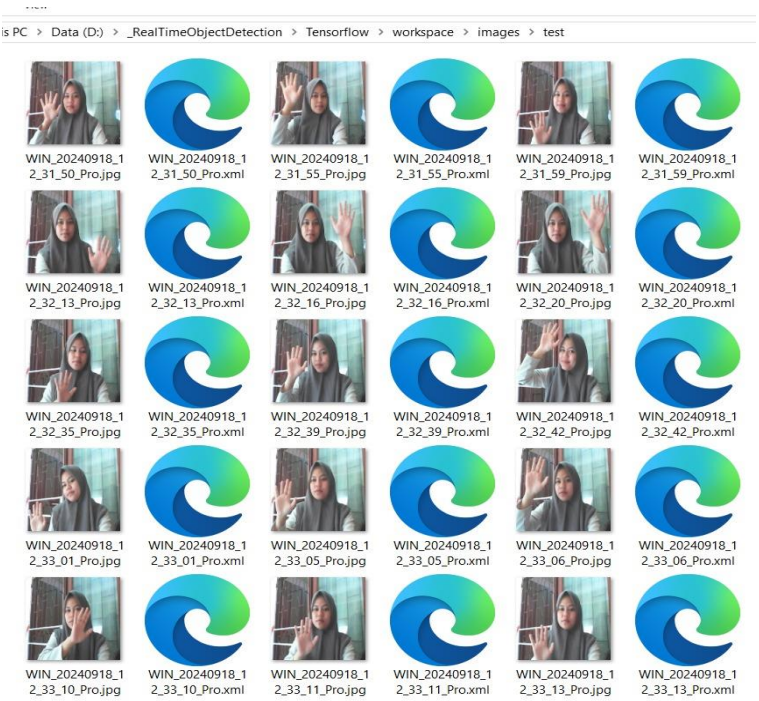
Nama	: Elsie Tria Paramian
TTL	: Grobogan, 31 Desember 2001
Agama	: Islam
Alamat	: Lobam Bestari, A4 No 22
Email	: elsieparamian31@gmail.com
Riwayat Pendidikan	SMA/SMK : SMAN 1 Bintan Utara
	SMP : SMPN 16 Bintan

Lampiran

LAMPIRAN A
DATASET

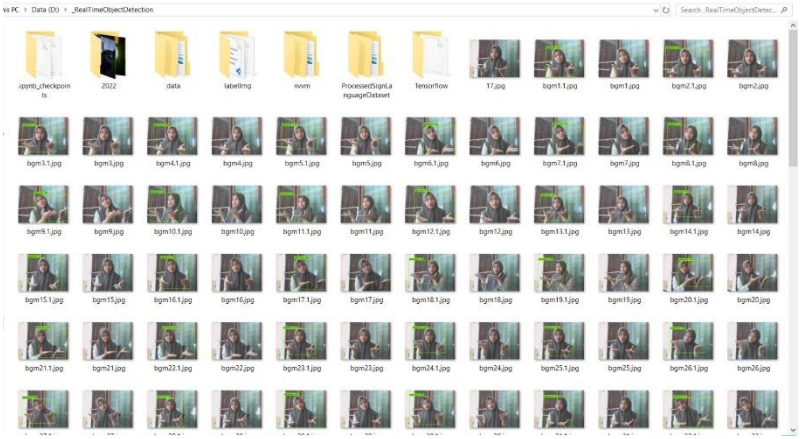


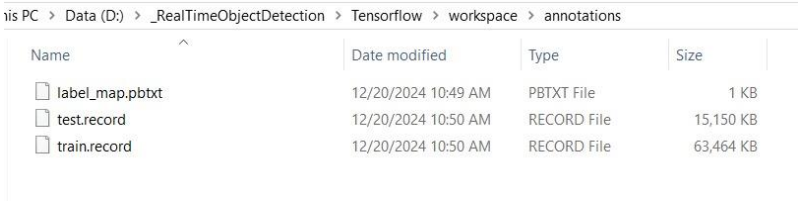

This PC > Data (D:) > _RealTimeObjectDetection > Tensorflow > workspace > images
Name
Date modified
Type
Size
test
11/27/2024 6:59 PM
File folder
train
11/23/2024 6:46 PM
File folder
images.rar
9/15/2022 7:10 AM
WinRAR archive
210,084 KB
is PC > Data (D:) > _RealTimeObjectDetection > Tensorflow > workspace > images > train
WIN_20240918_12_31_50_Pro.jpg
WIN_20240918_12_31_50_Pro.xml
WIN_20240918_12_31_55_Pro.jpg
WIN_20240918_12_31_55_Pro.xml
WIN_20240918_12_31_59_Pro.jpg
WIN_20240918_12_31_59_Pro.xml
WIN_20240918_12_32_13_Pro.jpg
WIN_20240918_12_32_13_Pro.xml
WIN_20240918_12_32_16_Pro.jpg
WIN_20240918_12_32_16_Pro.xml
WIN_20240918_12_32_20_Pro.jpg
WIN_20240918_12_32_20_Pro.xml
WIN_20240918_12_32_35_Pro.jpg
WIN_20240918_12_32_35_Pro.xml
WIN_20240918_12_32_39_Pro.jpg
WIN_20240918_12_32_39_Pro.xml
WIN_20240918_12_32_42_Pro.jpg
WIN_20240918_12_32_42_Pro.xml
WIN_20240918_12_33_01_Pro.jpg
WIN_20240918_12_33_01_Pro.xml
WIN_20240918_12_33_05_Pro.jpg
WIN_20240918_12_33_05_Pro.xml
WIN_20240918_12_33_06_Pro.jpg
WIN_20240918_12_33_06_Pro.xml
WIN_20240918_12_33_10_Pro.jpg
WIN_20240918_12_33_10_Pro.xml
WIN_20240918_12_33_11_Pro.jpg
WIN_20240918_12_33_11_Pro.xml
WIN_20240918_12_33_13_Pro.jpg
WIN_20240918_12_33_13_Pro.xml


is PC > Data (D:) > _RealTimeObjectDetection > Tensorflow > workspace > images > test
WIN_20240918_12_31_50_Pro.jpg
WIN_20240918_12_31_50_Pro.xml
WIN_20240918_12_31_55_Pro.jpg
WIN_20240918_12_31_55_Pro.xml
WIN_20240918_12_31_59_Pro.jpg
WIN_20240918_12_31_59_Pro.xml
WIN_20240918_12_32_13_Pro.jpg
WIN_20240918_12_32_13_Pro.xml
WIN_20240918_12_32_16_Pro.jpg
WIN_20240918_12_32_16_Pro.xml
WIN_20240918_12_32_20_Pro.jpg
WIN_20240918_12_32_20_Pro.xml
WIN_20240918_12_32_35_Pro.jpg
WIN_20240918_12_32_35_Pro.xml
WIN_20240918_12_32_39_Pro.jpg
WIN_20240918_12_32_39_Pro.xml
WIN_20240918_12_32_42_Pro.jpg
WIN_20240918_12_32_42_Pro.xml
WIN_20240918_12_33_01_Pro.jpg
WIN_20240918_12_33_01_Pro.xml
WIN_20240918_12_33_05_Pro.jpg
WIN_20240918_12_33_05_Pro.xml
WIN_20240918_12_33_06_Pro.jpg
WIN_20240918_12_33_06_Pro.xml
WIN_20240918_12_33_10_Pro.jpg
WIN_20240918_12_33_10_Pro.xml
WIN_20240918_12_33_11_Pro.jpg
WIN_20240918_12_33_11_Pro.xml
WIN_20240918_12_33_13_Pro.jpg
WIN_20240918_12_33_13_Pro.xml

LAMPIRAN B

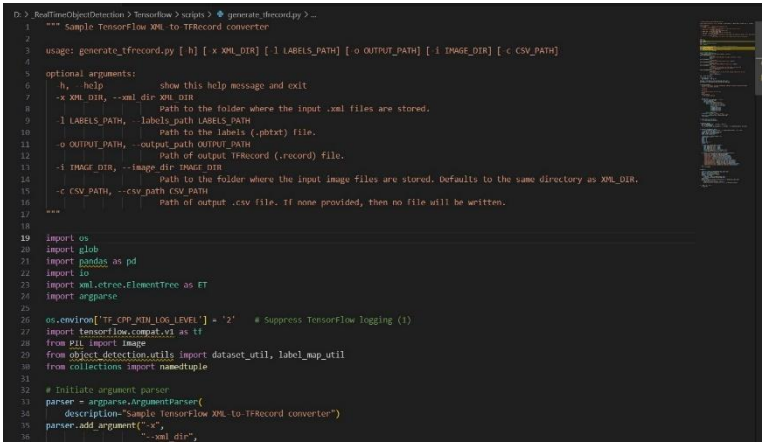
REALTIME OBJECT DETECTION FILE



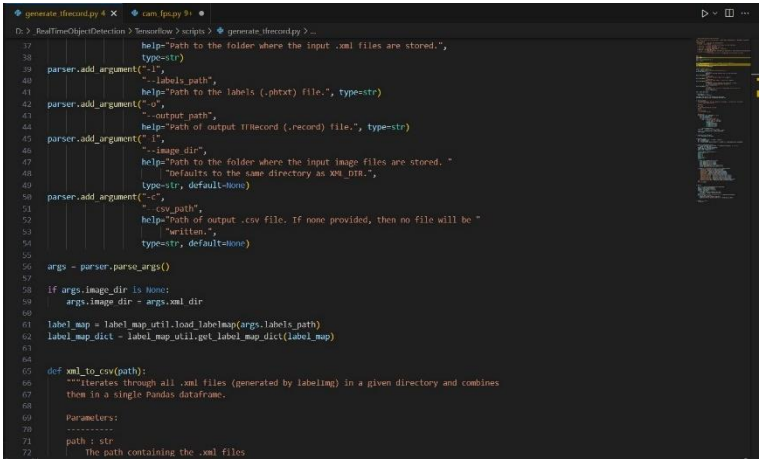


LAMPIRAN C
PROGRAM

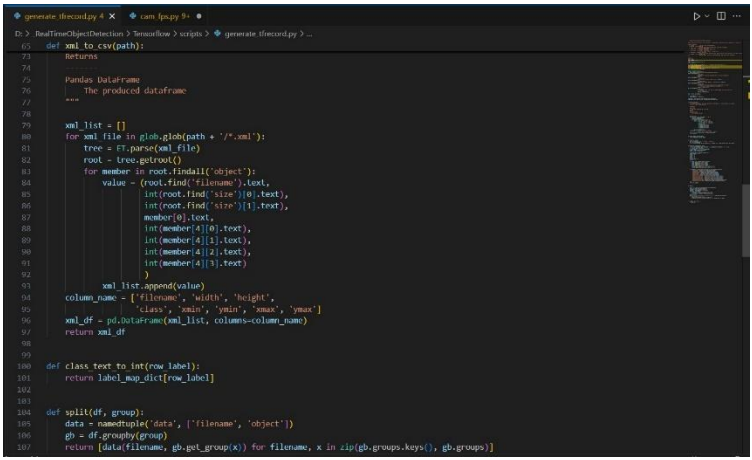
1.Labelimg



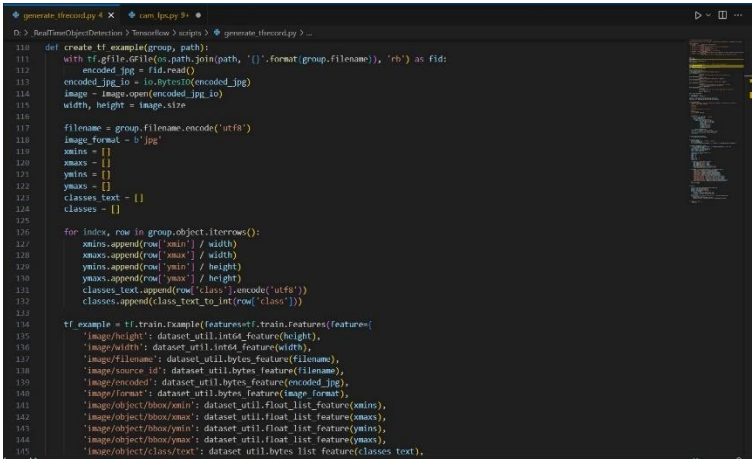
```
""" Sample TensorFlow XML-to-TFRecord converter

usage: generate_tfrecord.py [ h] [ x XML_DIR] [ l LABELS_PATH] [ o OUTPUT_PATH] [ i IMAGE_DIR] [ c CSV_PATH]

optional arguments:
  h, --help            show this help message and exit
  x XML_DIR, --xml_dir XML_DIR
                       Path to the folder where the input .xml files are stored.
  l LABELS_PATH, --labels_path LABELS_PATH
                       Path to the labels (.pbtxt) file.
  o OUTPUT_PATH, --output_path OUTPUT_PATH
                       Path of output TFRecord (.record) file.
  i IMAGE_DIR, --image_dir IMAGE_DIR
                       Path to the folder where the input image files are stored. Defaults to the same directory as XML_DIR.
  c CSV_PATH, --csv_path CSV_PATH
                       Path of output .csv file. If none provided, then no file will be written.
"""

import os
import glob
import pandas as pd
import io
import xml.etree.ElementTree as ET
import argparse

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'    # Suppress TensorFlow logging (1)
import tensorflow.compat.v1 as tf
from PIL import Image
from object_detection.utils import dataset_util, label_map_util
from collections import namedtuple

# Initiate argument parser
parser = argparse.ArgumentParser(
    description="Sample TensorFlow XML-to-TFRecord converter")
parser.add_argument("-x",
                    "--xml_dir",
```



```
                    help="Path to the folder where the input .xml files are stored.",
                    type=str)
parser.add_argument("-l",
                    "--labels_path",
                    help="Path to the labels (.pbtxt) file.", type=str)
parser.add_argument("-o",
                    "--output_path",
                    help="Path of output TFRecord (.record) file.", type=str)
parser.add_argument("-i",
                    "--image_dir",
                    help="Path to the folder where the input image files are stored. "
                         "Defaults to the same directory as XML_DIR.",
                    type=str, default=None)
parser.add_argument("-c",
                    "--csv_path",
                    help="Path of output .csv file. If none provided, then no file will be "
                         "written.",
                    type=str, default=None)

args = parser.parse_args()

if args.image_dir is None:
    args.image_dir = args.xml_dir

label_map = label_map_util.load_labelmap(args.labels_path)
label_map_dict = label_map_util.get_label_map_dict(label_map)


def xml_to_csv(path):
    """Iterates through all .xml files (generated by labelImg) in a given directory and combines
    them in a single Pandas dataframe.

    Parameters:
    ----------
    path : str
        The path containing the .xml files
```



```python
def xml_to_csv(path):
    Returns

    Pandas DataFrame
        The produced dataframe
    """

    xml_list = []
    for xml_file in glob.glob(path + '/*.xml'):
        tree = ET.parse(xml_file)
        root = tree.getroot()
        for member in root.findall('object'):
            value = (root.find('filename').text,
                     int(root.find('size')[0].text),
                     int(root.find('size')[1].text),
                     member[0].text,
                     int(member[4][0].text),
                     int(member[4][1].text),
                     int(member[4][2].text),
                     int(member[4][3].text)
                     )
            xml_list.append(value)
    column_name = ['filename', 'width', 'height',
                   'class', 'xmin', 'ymin', 'xmax', 'ymax']
    xml_df = pd.DataFrame(xml_list, columns=column_name)
    return xml_df


def class_text_to_int(row_label):
    return label_map_dict[row_label]


def split(df, group):
    data = namedtuple('data', ['filename', 'object'])
    gb = df.groupby(group)
    return [data(filename, gb.get_group(x)) for filename, x in zip(gb.groups.keys(), gb.groups)]
```



```python
def create_tf_example(group, path):
    with tf.gfile.GFile(os.path.join(path, '{}'.format(group.filename)), 'rb') as fid:
        encoded_jpg = fid.read()
    encoded_jpg_io = io.BytesIO(encoded_jpg)
    image = image.open(encoded_jpg_io)
    width, height = image.size

    filename = group.filename.encode('utf8')
    image_format = b'jpg'
    xmins = []
    xmaxs = []
    ymins = []
    ymaxs = []
    classes_text = []
    classes = []

    for index, row in group.object.iterrows():
        xmins.append(row['xmin'] / width)
        xmaxs.append(row['xmax'] / width)
        ymins.append(row['ymin'] / height)
        ymaxs.append(row['ymax'] / height)
        classes_text.append(row['class'].encode('utf8'))
        classes.append(class_text_to_int(row['class']))

    tf_example = tf.train.Example(features=tf.train.Features(feature={
        'image/height': dataset_util.int64_feature(height),
        'image/width': dataset_util.int64_feature(width),
        'image/filename': dataset_util.bytes_feature(filename),
        'image/source_id': dataset_util.bytes_feature(filename),
        'image/encoded': dataset_util.bytes_feature(encoded_jpg),
        'image/format': dataset_util.bytes_feature(image_format),
        'image/object/bbox/xmin': dataset_util.float_list_feature(xmins),
        'image/object/bbox/xmax': dataset_util.float_list_feature(xmaxs),
        'image/object/bbox/ymin': dataset_util.float_list_feature(ymins),
        'image/object/bbox/ymax': dataset_util.float_list_feature(ymaxs),
        'image/object/class/text': dataset_util.bytes_list_feature(classes_text),
```

```
        'image/object/class/label': dataset_util.int64_list_feature(classes),
    }))
    return tf_example


def main(_):

    writer = tf.python_io.TFRecordWriter(args.output_path)
    path = os.path.join(args.image_dir)
    examples = xml_to_csv(args.xml_dir)
    grouped = split(examples, 'filename')
    for group in grouped:
        tf_example = create_tf_example(group, path)
        writer.write(tf_example.SerializeToString())
    writer.close()
    print('Successfully created the TFRecord file: {}'.format(args.output_path))
    if args.csv_path is not None:
        examples.to_csv(args.csv_path, index=None)
        print('Successfully created the CSV file: {}'.format(args.csv_path))


if __name__ == '__main__':
    tf.app.run()
```

Ln 19, Col 10 Spaces: 4 UTF-8 CRLF Python

2. Cam Output

labelImg.py cam_fps.py 7

D: > _RealTimeObjectDetection > cam_fps.py > ...

```
import tensorflow as tf
import cv2
import numpy as np
import time
import os
from object_detection.utils import config_util
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder

# Setup model and config
WORKSPACE_PATH = 'Tensorflow/workspace'
MODEL_PATH = WORKSPACE_PATH + '/models'
CUSTOM_MODEL_NAME = 'my_ssd_mobnet'
CONFIG_PATH = MODEL_PATH + '/' + CUSTOM_MODEL_NAME + '/pipeline.config'
CHECKPOINT_PATH = MODEL_PATH + '/' + CUSTOM_MODEL_NAME + '/'

configs = config_util.get_configs_from_pipeline_file(CONFIG_PATH)
detection_model = model_builder.build(model_config=configs['model'], is_training=False)

ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(CHECKPOINT_PATH, 'ckpt-11')).expect_partial()

@tf.function
def detect_fn(image):
    image, shapes = detection_model.preprocess(image)
    prediction_dict = detection_model.predict(image, shapes)
    detections = detection_model.postprocess(prediction_dict, shapes)
    return detections

category_index = label_map_util.create_category_index_from_labelmap('Tensorflow/workspace/annotations/label_map.pbtxt')

# Setup capture
cap = cv2.VideoCapture(0)

# Variabel untuk memulai perhitungan waktu
```

labelImg.py cam_fps.py 7

D: > _RealTimeObjectDetection > cam_fps.py > ...

```python
# Variabel untuk memulai perhitungan waktu
detection_started = False
start_time = 0
elapsed_time = 0
prev_gray = None  # Variabel untuk menyimpan gambar sebelumnya
frame_threshold = 5000  # Threshold untuk mendeteksi pergerakan tangan

# Fungsi untuk format waktu fleksibel
def format_time(ms):
    """Mengubah milidetik menjadi ms, s, m, atau h sesuai durasi"""
    if ms < 1000:
        return f"{ms:.2f} ms"  # Menampilkan dalam milidetik
    elif ms < 60000:
        return f"{ms / 1000:.2f} s"  # Menampilkan dalam detik
    elif ms < 3600000:
        return f"{ms / 60000:.2f} m"  # Menampilkan dalam menit
    else:
        return f"{ms / 3600000:.2f} h"  # Menampilkan dalam jam

while True:
    ret, frame = cap.read()
    image_np = np.array(frame)
    image_np_rgb = cv2.cvtColor(image_np, cv2.COLOR_BGR2RGB)

    # Menghitung perbedaan gambar untuk mendeteksi gerakan tangan
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    if prev_gray is None:
        prev_gray = gray
        continue

    # Menghitung perbedaan antara frame saat ini dan frame sebelumnya
    frame_diff = cv2.absdiff(prev_gray, gray)
    non_zero_count = np.count_nonzero(frame_diff)  # Hitung jumlah piksel yang berubah
```

labelImg.py cam_fps.py 7

D: > _RealTimeObjectDetection > cam_fps.py > ...

```python
    # Deteksi gerakan tangan berdasarkan perbedaan frame
    hand_movement_detected = non_zero_count > frame_threshold  # Sesuaikan threshold sesuai kebutuhan

    # Jika gerakan tangan terdeteksi dan waktu belum mulai, mulai hitung waktu
    if hand_movement_detected:
        if not detection_started:
            start_time = time.time()  # Mulai waktu saat pertama kali gerakan terdeteksi
            detection_started = True
    # Jika gerakan tangan tidak terdeteksi, berhenti menghitung waktu
    else:
        if detection_started:
            elapsed_time = time.time() - start_time  # Hitung waktu sejak mulai deteksi tangan
            detection_started = False  # Waktu berhenti jika tangan hilang

    # Hitung durasi yang telah berlalu jika tangan terdeteksi
    if detection_started:
        elapsed_time = time.time() - start_time

    # Konversi ke ms
    elapsed_ms = elapsed_time * 1000  # Mengonversi detik menjadi milidetik
    formatted_time = format_time(elapsed_ms)

    # Menampilkan waktu di layar
    cv2.putText(image_np, f"Time: {formatted_time}", (20, 40), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv2.LINE_AA)

    # Deteksi objek menggunakan model SSD
    input_tensor = tf.convert_to_tensor(np.expand_dims(image_np_rgb, 0), dtype=tf.float32)
    detections = detect_fn(input_tensor)
    num_detections = int(detections.pop('num_detections'))
    detections = {key: value[0, :num_detections].numpy() for key, value in detections.items()}
    detections['num_detections'] = num_detections
    detections['detection_classes'] = detections['detection_classes'].astype(np.int64)

    # Hanya menampilkan deteksi objek tangan (kelas ID 1)
    hand_class_id = 1  # Ganti dengan ID kelas yang benar untuk tangan
```

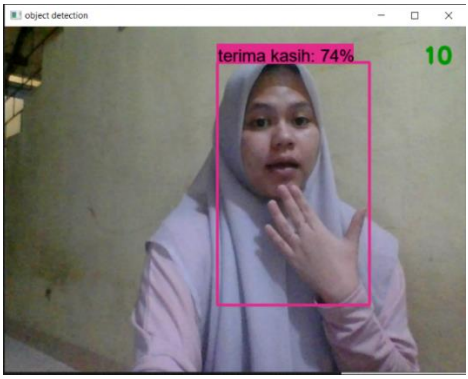
labelImg.py | cam_fps.py 7

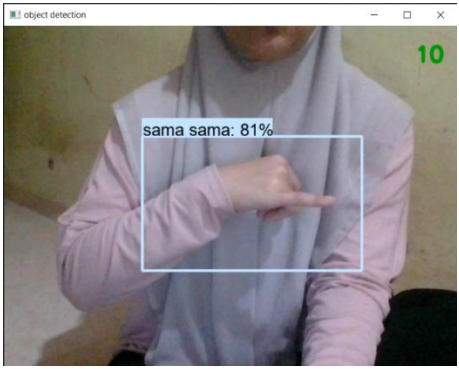
D: > _RealTimeObjectDetection > cam_fps.py > ...

```
103
104     # Hanya menampilkan deteksi objek tangan (kelas ID 1)
105     hand_class_id = 1  # Ganti dengan ID kelas yang benar untuk tangan
106     detection_classes = detections['detection_classes']
107     detected_class = np.isin(detection_classes, [hand_class_id])  # Memfilter hanya kelas tangan
108
109     # Jika kelas tangan terdeteksi, lakukan visualisasi
110     if np.any(detected_class):
111         viz_utils.visualize_boxes_and_labels_on_image_array(
112             image_np,
113             detections['detection_boxes'],
114             detections['detection_classes'] + 1,  # label offset
115             detections['detection_scores'],
116             category_index,
117             use_normalized_coordinates=True,
118             max_boxes_to_draw=200,
119             min_score_thresh=0.5,  # Threshold untuk deteksi tangan
120             agnostic_mode=False)
121
122     # Menampilkan gambar dengan waktu dan deteksi objek
123     cv2.imshow('Object Detection', image_np)
124
125     # Menutup aplikasi dengan menekan 'q'
126     if cv2.waitKey(1) & 0xFF == ord('q'):
127         break
128
129     prev_gray = gray  # Simpan gambar saat ini sebagai gambar sebelumnya
130
131 cap.release()
132 cv2.destroyAllWindows()
133
```

LAMPIRAN D
BEBERAPA HASIL OUTPUT

1. Elsie (Penulis)



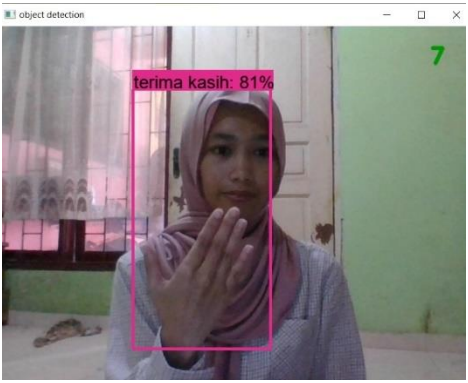



object detection
9
hallo: 91%

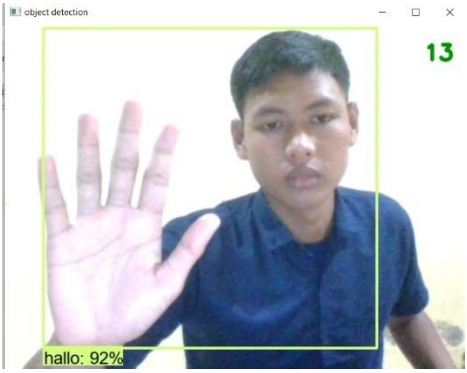
2. Gladys


object detection
7
wah, keren: 94%


object detection
7
bermain: 75%


object detection
terima kasih: 81%
7

3. Isa


object detection
13
hallo: 92%

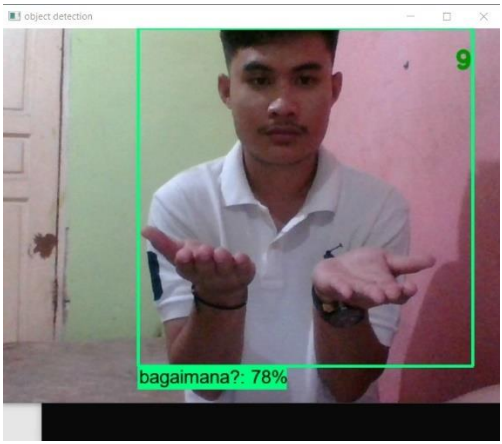

object detection
terima kasih: 80%
8


object detection
8
bagaimana?: 88%

5. Lingga


object detection
10
tolong: 91%


object detection
9
bermain: 89%


object detection
9
bagaimana?: 78%