KLASIFIKASI BAHASA ISYARAT DENGAN INPUT KAMERA PADA RASPBERRY PI MENGGUNAKAN ALGORITMA RANDOM FOREST DAN LONG SHORT-TERM MEMORY

PRAKTIK KERJA LAPANGAN

OLFAT HARITS ALATAS
312210018

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI DAN DESAIN
UNIVERSITAS MA CHUNG
MALANG
2025

LEMBAR PENGESAHAN

PRAKTIK KERJA LAPANGAN

KLASIFIKASI BAHASA ISYARAT DENGAN INPUT KAMERA PADA RASPBERRY PI MENGGUNAKAN ALGORITMA RANDOM FOREST DAN LONG SHORT-TERM MEMORY

Oleh:

OLFAT HARITS ALATAS

NIM.312210018

dari:

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI dan DESAIN

UNIVERSITAS MA CHUNG

Dosen Pembimbing,

Prof. Dr.Eng. Romy Budhi, ST., MT., M.Pd.
NIP.20070035

Dekan Fakultas Teknologi dan Desain,

Prof. Dr.Eng. Romy Budhi, ST., MT., M.Pd.
NIP.20070035

KATA PENGANTAR

Segala puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa atas limpahan rahmat dan karunia-Nya, sehingga pelaksanaan kegiatan Praktik Kerja Lapangan (PKL) ini dapat berjalan dengan lancar. Laporan ini disusun sebagai salah satu syarat untuk menyelesaikan mata kuliah PKL bagi mahasiswa Program Studi Teknik Informatika, Universitas Ma Chung Malang.

Pada kesempatan ini, penulis menyampaikan rasa terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan bantuan dan dukungan selama proses pelaksanaan PKL maupun dalam penyusunan laporan ini. Ucapan terima kasih secara khusus ditujukan kepada:

1. Bapak Prof. Dr.Eng. Romy Budhi, ST., MT., M.Pd., selaku dosen pembimbing sekaligus kepala peneliti dalam kegiatan PKL ini,
2. Rekan-rekan mahasiswa serta para responden atau subjek penelitian yang turut berpartisipasi dan memberikan dukungan,
3. Bapak Prof. Dr.Eng. Romy Budhi, ST., MT., M.Pd., dalam kapasitasnya sebagai Dekan Fakultas Sains dan Teknologi Universitas Ma Chung,
4. Bapak Hendry Setiawan, ST, M.Kom., selaku Ketua Program Studi Teknik Informatika.

Penulis telah berusaha menyusun laporan ini dengan sebaik mungkin. Namun, penulis menyadari bahwa laporan ini masih memiliki kekurangan. Semoga laporan PKL ini dapat memberikan manfaat dan informasi yang berguna bagi para pembaca.

Malang, 5 Juli 2025

Olfat Harits Alatas

DAFTAR ISI

KATA PENGANTAR i
DAFTAR ISI ii
DAFTAR GAMBAR iv
DAFTAR TABEL v
DAFTAR PERSAMAAN vi
Bab I Pendahuluan 1
1.1. Latar Belakang 1
1.2. Identifikasi Masalah 3
1.3. Batasan Masalah 3
1.4. Rumusan Masalah 3
1.5. Tujuan Penelitian 4
1.6. Manfaat 4
Bab II Gambaran Umum Perusahaan 5
2.1. Universitas Ma Chung 5
2.1.1. Program Studi Teknik Informatika 7
2.2. Pusat Studi Human Interaction 8
Bab III Tinjauan Pustaka 10
3.1. Bahasa Isyarat 10
3.1.1. BISINDO (Bahasa Isyarat Indonesia) 11
3.1.2. SIBI (Sistem Isyarat Bahasa Indonesia) 11
3.2. Python 12
3.2.1. Open CV 13
3.2.2. Numpy 13
3.2.3. Pandas 14
3.2.4. Scikit-learn 14
3.2.5. MediaPipe 15
3.2.6. Tensorflow 16
3.2.7. Joblib 16
3.2.8. Collections 17
3.2.9. Pickle 17
3.2.10. Matplotlib 17

3.2.11. Regular Expression (Regex) 18
3.2.12. OS (Operating System) 18
3.2.13. Imgaug 18
3.2.14. Long Short Term Memory (LSTM) 19
3.2.15. Random Forest 20
3.3. Raspberry Pi 22
Bab IV Deskripsi Data dan Hasil Praktik Kerja Lapangan 27
4.1. Profil Partisipan 27
4.2. Proses Pengerjaan 27
4.3. Pengumpulan Data 29
4.4. Ekstraksi Fitur 31
4.5. Augmentasi Data 33
4.6. Proses Train Model Random Forest 36
4.7. Proses Train Model LSTM 37
4.8. Evaluasi Model 41
4.9. Implementasi Sistem Deteksi Gesture Real-Time 47
4.10. Evaluasi real time 51
Bab V Penutup 55
5.1. Kesimpulan 55
5.2. Saran 55
DAFTAR PUSTAKA 57
Lampiran 60

DAFTAR GAMBAR

Gambar 2. 1 Diagram fokus riset pusat studi *Human-Machine Interaction* 9
Gambar 3. 1 Abjad dalam BISINDO 11
Gambar 3. 2 Abjad dalam SIBI 12
Gambar 3. 3 Peta *MediaPipe Hand Landmarks* 16
Gambar 3. 4 *Rasberry Pi 4 model B* 24
Gambar 4. 1 Tahapan pengerjaan 28
Gambar 4. 2 Skema pengumpulan data dan titik-titik landmark yang akan dikonversi dalam format file CSV 29
Gambar 4. 3 Gestur "Ambil" (Sumber: Alexander, 2023) 31
Gambar 4. 4 Cuplikan Kode Ekstraksi Fitur Landmark Tangan 32
Gambar 4. 5 Kode untuk Augmentasi Data 34
Gambar 4. 6 Dataset Sebelum Di Augmentasi 35
Gambar 4. 7 Dataset Setelah Di Augmentasi 35
Gambar 4. 8 Cuplikan Kode Pelatihan Random Forest 36
Gambar 4. 9 Cuplikan Kode Pelatihan LSTM 38
Gambar 4. 10 Confusion Matrix Random Forest 43
Gambar 4. 11 Confusion Matrix Model LTSM 45
Gambar 4. 12 Kurva Akurasi dan Loss 46
Gambar 4. 13 Flowchart program Hand Tracking 49
Gambar 4.14. a Cuplikan Kode Translasi Gestur 49
Gambar 4.14. b Cuplikan Kode Translasi Gestur 50
Gambar 4.15 Contoh *Output* Implementasi *Real Time* 50

DAFTAR TABEL

Tabel 3. 1 Versi *Rasberry Pi* 23
Tabel 4. 1 Tabel data subjek 27
Tabel 4. 2 Daftar 40 korpus harian 30
Tabel 4. 3 Gestur yang diuji oleh Random Forest 42
Tabel 4. 4 Gestur yang diuji oleh LSTM 44
Tabel 4. 5 Hasil classification report dari model Random Forest dan LSTM 44
Tabel 4. 6 Hasil pengujian real time 52

DAFTAR PERSAMAAN

Persamaan (1) Rumus *Input Gate* 19
Persamaan (2) Rumus *Forget Gate* 20
Persamaan (3) Rumus *Output Gate* 20
Persamaan (4) Rumus *Random Forest* 21

Bab I
Pendahuluan

1.1. Latar Belakang

Bahasa Isyarat Indonesia (BISINDO) merupakan bentuk komunikasi utama yang digunakan oleh penyandang tunarungu dan tunawicara di Indonesia. Sebagai bahasa visual, BISINDO disampaikan melalui kombinasi gerakan tangan, ekspresi wajah, dan posisi tubuh, sehingga sering kali sulit dipahami oleh masyarakat umum yang tidak terbiasa menggunakannya. Kondisi ini menimbulkan hambatan komunikasi antara penyandang disabilitas dengan lingkungan sosial di sekitarnya. Di Indonesia, dikenal dua sistem bahasa isyarat, yaitu Sistem Isyarat Bahasa Indonesia (SIBI) dan Bahasa Isyarat Indonesia (BISINDO). Namun demikian, BISINDO lebih banyak digunakan dalam kehidupan sehari-hari oleh komunitas tunarungu karena bersifat alami, berkembang secara kultural, serta tidak terikat pada struktur bahasa Indonesia formal. Dengan mempertimbangkan aspek penggunaannya yang lebih luas dan praktis, penelitian ini menggunakan BISINDO sebagai dasar dalam pengembangan sistem penerjemah bahasa isyarat otomatis guna mendukung komunikasi yang lebih inklusif.

Perkembangan teknologi kecerdasan buatan, khususnya *machine learning*, telah memungkinkan sistem komputer untuk belajar mengenali pola dari data. Salah satu penerapannya adalah dalam sistem pengenalan bahasa isyarat berbasis citra. Beberapa penelitian telah menunjukkan keberhasilan penggunaan algoritma *machine learning* dalam klasifikasi gerakan tangan. Misalnya, algoritma *Random Forest* dan *Long Short-Term Memory* digunakan untuk mengenali isyarat tangan statis berdasarkan fitur geometri atau hasil ekstraksi gambar dari kamera. Studi oleh Fadlilah et al. (2022) berhasil membangun sistem pengenal isyarat dasar BISINDO menggunakan kamera dan Raspberry Pi, di mana sistem mampu menerjemahkan gestur huruf dan angka menjadi teks yang ditampilkan secara real-time pada layar monitor.

Namun, bahasa isyarat tidak hanya terdiri dari gerakan statis, tetapi juga gerakan dinamis yang menyusun kata atau kalimat. Untuk itu, dibutuhkan pendekatan yang

mampu mengenali urutan gerakan dalam dimensi waktu. Model *Long Short-Term Memory* (LSTM), yang merupakan bagian dari arsitektur *Recurrent Neural Network* (RNN), terbukti efektif dalam menangani data sekuensial. Penelitian oleh Aljabar dan Suharjito (2020) menggabungkan CNN dan LSTM untuk mengenali isyarat BISINDO secara *real-time* menggunakan perangkat desktop., dengan hasil akurasi mencapai 96% dalam pengenalan kata-kata tertentu

Selain itu, penggunaan perangkat Raspberry Pi sebagai media pengembangan sistem memberikan keuntungan dalam hal biaya, ukuran, dan portabilitas. Sistem ini dapat diintegrasikan dengan kamera dan dipasang secara fleksibel di berbagai lokasi, menjadikannya cocok untuk alat bantu komunikasi yang praktis dan mobile.

Penelitian yang dilakukan oleh Alexander dkk., (2023) menunjukkan bahwa algoritma Random Forest memiliki kinerja paling unggul dibandingkan model klasifikasi lain seperti *K-Nearest Neighbor* (KNN), *Support Vector Machine* (SVM), dan *Decision Tree* dalam mengenali gestur bahasa isyarat BISINDO. *Random Forest* mencatat nilai akurasi, presisi, *f1-score*, dan *recall* sebesar 97,9% pada data pengujian, serta 84% presisi dalam pengujian *real-time*, menjadikannya model yang paling stabil dan akurat dalam klasifikasi gestur statis. Keunggulan ini juga tercermin dari waktu klasifikasi paling singkat dibandingkan model lain, yaitu mencapai 34,6 kata per menit, menjadikan *Random Forest* sangat sesuai untuk aplikasi *real-time* berbasis kamera.

Berdasarkan hasil tersebut, penelitian ini mengadopsi model *Random Forest* sebagai komponen utama untuk mengklasifikasi gestur statis. Namun, karena *Random Forest* tidak secara optimal menangani data sekuensial atau temporal yang terdapat pada gestur dinamis, maka pada penelitian ini dikembangkan pendekatan gabungan dengan menambahkan model LSTM (*Long Short-Term Memory*). LSTM digunakan untuk mengenali pola-pola gerakan berurutan yang khas pada gestur dinamis, sementara *Random Forest* tetap digunakan untuk menangani klasifikasi gestur statis yang bersifat spasial. Pendekatan gabungan ini diharapkan mampu meningkatkan akurasi sistem secara keseluruhan dalam mendeteksi bahasa isyarat BISINDO, baik pada gestur statis maupun dinamis.

1.2. Identifikasi Masalah

Permasalahan yang diangkat dalam penelitian ini adalah masih terbatasnya pemahaman masyarakat terhadap BISINDO, belum tersedianya sistem penerjemah bahasa isyarat yang praktis dan *real-time* pada perangkat portabel seperti Raspberry Pi, serta perlunya pemisahan metode klasifikasi untuk gerakan statis dan dinamis yang masing-masing memerlukan algoritma yang sesuai seperti *Random Forest* dan LSTM.

1.3. Batasan Masalah

a) Sistem ini dirancang untuk mengenali alfabet, angka, dan 40 kosakata sehari-hari dari bahasa isyarat BISINDO.
b) Bahasa pemrograman yang digunakan dalam pengembangan sistem adalah Python.
c) Deteksi tangan dilakukan menggunakan MediaPipe untuk menangkap posisi telapak tangan dan jari.
d) Klasifikasi gerakan statis dilakukan dengan algoritma *Random Forest*.
e) Untuk gerakan dinamis, digunakan algoritma *Long Short-Term Memory* (LSTM) untuk mengenali pola urutan gerakan tangan.
f) Prototipe dikembangkan agar dapat melakukan klasifikasi secara real-time di perangkat *Raspberry Pi 4 Model B*.

1.4. Rumusan Masalah

a) Bagaimana merancang sistem klasifikasi bahasa isyarat BISINDO menggunakan input kamera *webcam Logitech C270* pada Raspberry Pi.
b) Bagaimana kinerja algoritma *Random Forest* dalam mengklasifikasikan gerakan statis bahasa isyarat BISINDO.
c) Bagaimana penerapan algoritma *Long Short-Term Memory* (LSTM) dalam mengenali gerakan dinamis bahasa isyarat BISINDO.
d) Apakah sistem dapat melakukan klasifikasi secara *real-time* pada perangkat *Raspberry Pi 4 Model B* dengan hasil yang akurat.

1.5. Tujuan Penelitian

a) Membangun sistem penerjemah bahasa isyarat BISINDO berbasis kamera yang berjalan di perangkat *Raspberry Pi 4 Model B* dan mampu melakukan klasifikasi secara *real-time* secara optimal.
b) Menerapkan algoritma *Random Forest* untuk klasifikasi gerakan statis dan LSTM (*Long Short-Term Memory*) untuk klasifikasi gerakan dinamis dari bahasa isyarat BISINDO.

1.6. Manfaat

a) Memberikan solusi teknologi yang membantu komunikasi antara penyandang tunarungu dan masyarakat umum.
b) Menjadi referensi dalam pengembangan sistem pengenalan bahasa isyarat berbasis Raspberry Pi dan kamera.
c) Memberikan perbandingan performa beberapa algoritma *machine learning* dalam konteks klasifikasi isyarat tangan.
d) Mendorong pemanfaatan model *deep learning* seperti LSTM dalam pengenalan gerakan dinamis berbasis sekuensial.

Bab II
Gambaran Umum Perusahaan

2.1. Universitas Ma Chung

Universitas Ma Chung adalah institusi pendidikan tinggi berstatus swasta yang berlokasi di Malang dan berada di bawah naungan Yayasan Harapan Bangsa Sejahtera. Pendirian universitas ini terjadi pada 7 Juli 2007 dengan alamat di Villa Puncak Tidar N-01, Karang Widoro, Dau, Malang, Jawa Timur. Visi universitas ini adalah untuk mengagungkan Tuhan Yang Maha Esa melalui pembentukan karakter, pengembangan ilmu pengetahuan, dan memberikan kontribusi bermakna sebagai akademisi yang memiliki daya kreativitas tinggi. Misi yang diemban meliputi enam aspek utama:

Pertama, melaksanakan Tri Dharma Perguruan Tinggi yang mencakup kegiatan pendidikan, riset, dan pelayanan masyarakat dengan standar berkualitas, terfokus, dan relevan dengan tuntutan zaman.

Kedua, menciptakan generasi motivator dan pemimpin masyarakat yang memiliki integritas moral tinggi, berjiwa kepemimpinan dan entrepreneurship dengan penekanan pada pembentukan karakter mulia, sikap rendah hati, jiwa pelayanan, dan kontribusi sebagai pribadi yang sempurna.

Ketiga, menumbuhkan sikap dan pola pikir kritis yang berprinsip serta kreatif-realistis berdasarkan kepekaan nurani yang mulia.

Keempat, mencetak lulusan berkualitas tinggi yang siap terjun ke dunia kerja dan mampu berkompetisi di tingkat global.

Kelima, berpartisipasi aktif dalam memajukan peradaban dunia dengan menghasilkan alumni berwawasan internasional, toleran, cinta damai, dan produktif dalam menciptakan karya yang meningkatkan harkat kemanusiaan universal.

Keenam, menjalankan tata kelola perguruan tinggi berdasarkan prinsip efisiensi ekonomi dan transparansi.

Universitas Ma Chung juga memiliki landasan filosofis berupa 12 nilai dasar:

1. Orisinal

Keyakinan bahwa setiap individu memiliki keunikan tersendiri sehingga harus berani tampil autentik tanpa meniru orang lain. Seluruh civitas akademika didorong untuk mengedepankan keorisinilan dalam setiap karya dan inisiatif.

2. Terpercaya

Menjunjung tinggi kejujuran dalam pemikiran, tindakan, dan perkataan untuk membangun institusi dan komunitas akademik yang kredibel, terhormat, andal, dan dapat dipercaya.

3. Gigih

Keyakinan bahwa kesabaran dan ketekunan dapat mengatasi segala hambatan dan masalah. Komitmen untuk membangun budaya pantang menyerah, tekun, tidak mengenal lelah, dan tidak mudah putus asa.

4. Kreatif

Mendorong terciptanya budaya kerja yang inovatif, produktif, dan imajinatif untuk senantiasa mengembangkan pendekatan dan metode baru dalam bekerja dan meraih kesuksesan.

5. Dinamis

Komitmen menciptakan lingkungan kerja dan pembelajaran yang selalu hidup, bersemangat, dan aktif sehingga memungkinkan seluruh civitas akademika untuk mengantisipasi, beradaptasi, dan mengakomodasi perubahan.

6. Ramah dan Menyenangkan

Mendorong terciptanya lingkungan kerja dan pembelajaran yang tertib, penuh kegembiraan dan menyenangkan untuk menghasilkan SDM yang ramah, toleran, pembawa kedamaian, dan sukacita.

7. Meritokratik

Menghargai prestasi, kerja keras, dan kontribusi nyata dengan prinsip bahwa pencapaian menentukan posisi seseorang.

8. Profesional

Komitmen membangun etos kerja yang selalu mengutamakan kualitas dan motivasi untuk menjadi yang terbaik dalam setiap upaya.

9. Bertanggung Jawab

Mendorong seluruh civitas akademika untuk mampu mempertanggungjawabkan semua pemikiran, tindakan, dan perkataan secara baik dan benar.

10. Sinergi

Mendorong kemampuan untuk mempertanggungjawabkan semua pemikiran, tindakan, dan perkataan dengan baik dan benar.

11. Rendah Hati

Mendorong sikap untuk selalu mampu mempertanggungjawabkan semua pemikiran, tindakan, dan perkataan dengan baik dan benar.

12. Kewarganegaraan

Berperan proaktif dalam memberikan kontribusi untuk membangun masyarakat dan lingkungan hidup yang aman, sehat, damai, sejahtera, adil, dan makmur.

Saat ini, Universitas Ma Chung memiliki 11 program studi yaitu Manajemen Bisnis, Akuntansi, Magister Manajemen Inovasi, Sastra Inggris, Pendidikan Bahasa Mandarin, Teknik Informatika, Sistem Informasi, Desain Komunikasi Visual, Teknik Industri, Farmasi, Optometri dan Profesi Apoteker.

2.1.1. Program Studi Teknik Informatika

Jurusan Teknik Informatika merupakan salah satu bidang studi yang tersedia di Universitas Ma Chung dengan rancangan khusus untuk menjawab tuntutan sektor industri teknologi informasi masa kini. Jurusan ini menyediakan pembelajaran bermutu tinggi yang ditunjang dengan sarana dan prasarana komprehensif guna mendukung proses pembelajaran dan pengajaran. Program studi ini juga membuka peluang bagi para mahasiswa untuk terlibat dalam kegiatan magang industri di berbagai perusahaan atau berpartisipasi dalam program Merdeka Belajar Kampus Merdeka (MBKM). Inisiatif ini dirancang dengan tujuan memberikan eksposur praktis dan pengalaman langsung kepada para mahasiswa.

Dalam Program Studi Teknik Informatika di Universitas Ma Chung, tersedia dua track spesialisasi utama yaitu spesialisasi sistem cerdas dan spesialisasi sistem komputer. Spesialisasi sistem cerdas menitikberatkan pada kajian mengenai artificial

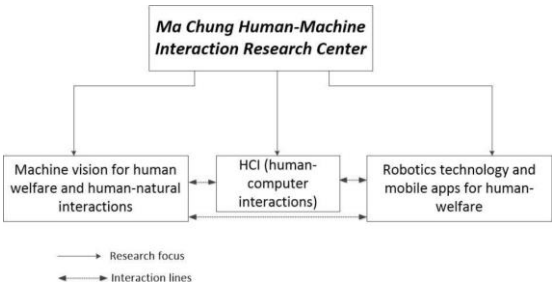
intelligence atau kecerdasan buatan, sedangkan spesialisasi sistem komputer mengutamakan pembelajaran tentang interconnected networks dan smart devices yang memiliki kemampuan untuk saling berkoneksi dan melakukan pertukaran data.

Kegiatan Praktik Kerja Lapangan menjadi komponen integral dari kedua spesialisasi - baik sistem cerdas maupun sistem komputer. Program ini juga merupakan manifestasi dari aktivitas riset yang termasuk dalam framework Merdeka Belajar Kampus Merdeka (MBKM) internal yang diselenggarakan oleh Universitas Ma Chung.

2.2. Pusat Studi Human Interaction

Pusat Studi Human-Machine Interaction didirikan pada 11 September 2019 dan berada di bawah naungan Program Studi Teknik Informatika Universitas Ma Chung. Pusat studi ini memiliki misi untuk mengembangkan teknologi beserta aplikasinya dalam ranah interaksi antara manusia dan mesin. Gambar 2.1 memperlihatkan skema area fokus penelitian dari Pusat Studi Human-Machine Interaction. Pusat studi ini berlokasi di gedung Research and Development lantai 6 Universitas Ma Chung. Terdapat beberapa area kajian yang menjadi fokus penelitian di pusat studi ini:

a) *Machine vision for human welfare and human-natural interactions* Area ini mengkaji penerapan teknologi visi mesin untuk mendukung kesejahteraan manusia serta interaksi natural manusia.
b) *Human-computer interactions (HCI)* Area ini melakukan riset mengenai perancangan, implementasi, dan evaluasi perangkat atau *User Interface* (UI) dan *User Experience* (UX).
c) *Robotics technology and mobile apps for human welfare* Area ini meneliti implementasi teknologi robotika dalam meningkatkan kesejahteraan manusia.



Gambar 2. 1 Diagram fokus riset pusat studi *Human-Machine Interaction*

Bab III

Tinjauan Pustaka

3.1. Bahasa Isyarat

Bahasa isyarat merupakan sistem komunikasi visual-manual yang digunakan oleh komunitas Tuli untuk menyampaikan informasi dan menjalin interaksi sosial. Bahasa ini memiliki struktur linguistik yang kompleks yang mencakup fonologi, morfologi, sintaksis, serta semantik, dan tidak bersifat universal karena tiap negara memiliki bahasa isyaratnya masing-masing (Pujiati, 2019). Di Indonesia, terdapat dua sistem yang dikenal luas, yaitu Bahasa Isyarat Indonesia (BISINDO) dan Sistem Isyarat Bahasa Indonesia (SIBI), yang memiliki perbedaan mendasar dalam struktur dan penggunaannya (Pujiati, 2019).

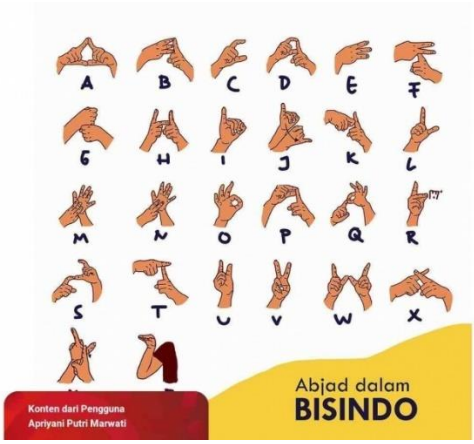
Selain sebagai alat komunikasi, bahasa isyarat juga merupakan bagian dari identitas budaya komunitas Tuli. Penggunaan bahasa isyarat memberikan akses yang lebih luas terhadap pendidikan, pekerjaan, dan kehidupan sosial secara umum. Namun, pengakuan resmi terhadap bahasa isyarat sebagai bahasa nasional dan penerapannya dalam kebijakan publik masih menjadi tantangan di berbagai negara, termasuk Indonesia (Wijaya, 2018).

Dalam konteks pendidikan, pemahaman dan penguasaan bahasa isyarat memiliki peranan penting dalam menciptakan lingkungan belajar yang inklusif bagi siswa Tuli. Implementasi pembelajaran dengan pendekatan bilingual menggunakan bahasa isyarat dan bahasa tulis mampu meningkatkan prestasi belajar dan perkembangan kognitif siswa Tuli (Murni et al., 2024). Selain itu, pelatihan bahasa isyarat bagi guru, teman sebaya, dan keluarga turut memperkuat dukungan sosial yang mereka terima.

Seiring dengan kemajuan teknologi, banyak inovasi digital yang mendukung pembelajaran bahasa isyarat, seperti aplikasi interaktif berbasis multimedia. Aplikasi ini dirancang untuk mengajarkan kosakata bahasa isyarat melalui animasi, audio, dan latihan visual, sehingga menarik minat pengguna dan meningkatkan efektivitas pembelajaran (Assa et al., 2021).

3.1.1. BISINDO (Bahasa Isyarat Indonesia)

BISINDO adalah bahasa yang didorong pengembangannya oleh Gerakan Kesejahteraan Tunarungu Indonesia (Gerkatin) dan dikembangkan secara mandiri oleh komunitas tunarungu. Oleh karena itu, BISINDO menjadi sistem komunikasi yang praktis dan efisien bagi penyandang tunarungu di Indonesia karena bahasa ini memang lahir dari kebutuhan dan pengalaman langsung para tunarungu itu sendiri (Borman et al., 2017). Karena berasal dari komunitas tunarungu itu sendiri, BISINDO lebih mudah dipahami dan diterima dalam interaksi sehari-hari. Dibandingkan dengan SIBI yang muncul belakangan, BISINDO telah digunakan lebih lama dan mencerminkan cara berkomunikasi teman tuli secara alami. Dalam penggunaannya, BISINDO menekankan ekspresi wajah dan gerakan mulut sebagai bagian penting dari makna isyarat. Selain itu, struktur bahasa ini terdiri dari lima unsur utama, yaitu lokasi isyarat, bentuk tangan, orientasi, gerakan tangan, dan ekspresi non-manual (Nugraheni et al., 2021). Visualisasi dari beberapa contoh gestur BISINDO dapat dilihat pada Gambar 3.1, yang memperlihatkan bentuk tangan dan ekspresi khas yang digunakan dalam komunikasi sehari-hari.



Gambar 3. 1 Abjad dalam BISINDO

(Sumber: Kompasiana, 2023)

3.1.2. SIBI (Sistem Isyarat Bahasa Indonesia)

SIBI (Sistem Isyarat Bahasa Indonesia) merupakan alat bantu komunikasi bagi

individu tunarungu yang menggabungkan unsur bahasa lisan, gerakan isyarat, ekspresi wajah, dan gerak tubuh lainnya. Pemerintah menetapkan SIBI sebagai bahasa isyarat resmi yang digunakan di Sekolah Luar Biasa (SLB). Namun, banyak penyandang tunarungu merasa bahwa SIBI tidak sepenuhnya mewakili cara berkomunikasi mereka, karena SIBI menggunakan aturan isyarat yang cenderung menyesuaikan dengan struktur bahasa lisan dalam menyampaikan kosakata (Nugraheni et al., 2021). Gambar 3.2 menyajikan visualisasi dari beberapa contoh gestur dalam SIBI.

Gambar 3. 2 Abjad dalam SIBI

(Sumber: Yayasan Peduli Kasih ABK, 2018)

3.2. Python

Python adalah bahasa pemrograman tingkat tinggi dan serbaguna yang dikembangkan oleh Guido van Rossum pada akhir 1980-an dan pertama kali dirilis pada 1991. Filosofi desainnya menekankan keterbacaan kode dengan penggunaan indentasi yang signifikan, bersifat dinamis dalam pengecekan tipe data, dan mendukung berbagai paradigma pemrograman seperti terstruktur, berorientasi objek, dan fungsional. Sering disebut sebagai bahasa *"batteries included"* karena memiliki pustaka standar yang komprehensif, Python telah berkembang melalui beberapa versi utama (*Python* 2 dan *Python* 3) dan secara konsisten menduduki peringkat sebagai salah satu bahasa pemrograman terpopuler, terutama dalam komunitas machine

learning. Nama "*Python*" diambil dari serial komedi Inggris *"Monty Python's Flying Circus"*, dan kepemimpinan proyeknya beralih dari Van Rossum (yang dijuluki "*benevolent dictator for life*") kepada *Steering Council* beranggotakan lima orang pada 2019 (Van Rossum, G., 2007).

3.2.1. Open CV

Open Source Computer Vision Library (*OpenCV*) merupakan pustaka *open-source* yang dikembangkan oleh Intel pada tahun 1999 sebagai bagian dari inisiatif pengembangan aplikasi pemrosesan visual *real-time* yang efisien. Pustaka ini ditulis dalam bahasa pemrograman C dan dioptimalkan untuk mendukung arsitektur prosesor *multicore*, menjadikannya sangat cocok untuk aplikasi yang membutuhkan kecepatan pemrosesan tinggi (Bradski dan Kaehler, 2008; Kaehler dan Bradski, 2016). *OpenCV* kini telah berkembang menjadi salah satu pustaka paling populer dalam pengembangan sistem *computer vision* karena sifatnya yang fleksibel dan dapat dijalankan di berbagai platform.

Menurut Dawson-Howe (2019), *OpenCV* memiliki beragam fungsi penting, mulai dari pengolahan citra dasar seperti *filtering*, konversi warna, dan deteksi tepi, hingga pemrosesan citra lanjutan seperti deteksi wajah, pelacakan objek, kalibrasi kamera, serta pengenalan pola berbasis *machine learning*. Selain itu, *OpenCV* juga banyak digunakan untuk pengolahan video, rekonstruksi 3D, dan implementasi *augmented reality* serta sistem bantuan pengemudi. Kelengkapan dan kemudahan integrasi pustaka ini membuat *OpenCV* menjadi alat yang sangat bermanfaat untuk riset maupun pengembangan industri.

3.2.2. Numpy

NumPy (*Numerical* Python) adalah pustaka *open-source* dalam Python yang digunakan untuk komputasi numerik, terutama pada array dan matriks berdimensi banyak. Pustaka ini menyediakan fungsi-fungsi efisien untuk operasi matematika, statistik, aljabar linier, transformasi *Fourier*, dan bilangan acak. *NumPy* dibangun di atas kode C yang dioptimalkan, sehingga menawarkan kecepatan tinggi dengan sintaks

Python yang sederhana. Meskipun tidak menyediakan fungsi statistik lanjutan, NumPy menjadi dasar penting bagi pustaka lain seperti *pandas*, terutama dalam pengolahan data tabular (Gupta, P., dan Bagchi, A., 2024).

3.2.3. Pandas

Pandas adalah pustaka *open-source* dalam Python yang dirancang untuk keperluan analisis dan manipulasi data, terutama data dalam bentuk tabel (tabular) seperti spreadsheet dan database. Pustaka ini memungkinkan Python untuk melakukan operasi pemrosesan data secara cepat dan efisien, seperti membaca, membersihkan, menyusun ulang, menyatukan, hingga memodelkan dan menganalisis data. *Pandas* menawarkan struktur data utama berupa *Series* (data satu dimensi) dan *DataFrame* (data dua dimensi), yang memudahkan pengguna dalam mengelola data kompleks.

Pandas pertama kali dikembangkan oleh Wes McKinney pada tahun 2008, dan dibangun di atas *NumPy*. Meskipun tidak menggantikan *NumPy*, *Pandas* memperluas kemampuannya dengan menyediakan alat-alat analisis data yang lebih ekspresif dan terstruktur. Pandas mendukung berbagai format data seperti CSV, Excel, JSON, dan SQL, serta dapat menangani data heterogen, data waktu (*time series*), dan data tidak lengkap (Gupta, P., dan Bagchi, A., 2024).

3.2.4. Scikit-learn

Scikit-learn adalah pustaka *open-source* untuk *machine learning* yang ditulis dalam bahasa *Python*. Pustaka ini memungkinkan integrasi metode *machine learning* secara cepat dan mudah ke dalam kode Python. *Scikit-learn* menyediakan berbagai metode seperti klasifikasi, regresi, estimasi *matriks kovarian*, reduksi dimensi, praproses data, hingga pembuatan dataset uji.

Pustaka ini dapat digunakan di berbagai sistem operasi dan terus dikembangkan secara aktif. *Scikit-learn* banyak digunakan dalam aplikasi komersial, penelitian akademik, dan publikasi ilmiah. Untuk meningkatkan efisiensi, beberapa algoritma dalam *scikit-learn* ditulis dalam bahasa C dan diintegrasikan melalui *Cython*, yang memungkinkan kompilasi *Python* secara lebih cepat. Selain itu, metode seperti SVM

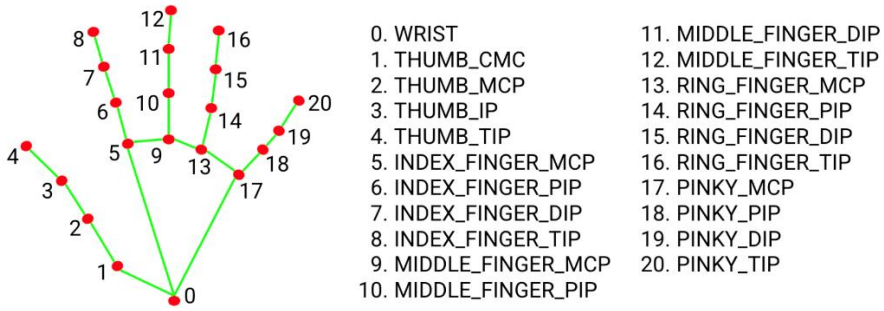
dan *logistic regression* dalam *scikit-learn* menggunakan pustaka LIBSVM dan LIBLINEAR sebagai dasar algoritmanya (Kramer, O., dan Kramer, O., 2016).

3.2.5. MediaPipe

MediaPipe adalah *framework open-source* yang dikembangkan oleh *Google* untuk membangun *pipeline* pemrosesan data sensorik seperti video, audio, dan input dari sensor lainnya secara modular dan *real-time*. *Framework* ini dirancang untuk membantu pengembang dalam membangun sistem pemrosesan visual yang kompleks dengan menyusun proses ke dalam bentuk *graph* kalkulator (*calculator graph*)—yaitu kumpulan komponen modular yang dapat digunakan kembali dan dikombinasikan sesuai kebutuhan.

MediaPipe mendukung berbagai platform, termasuk desktop, mobile (Android/iOS), dan bahkan perangkat *embedded* seperti Raspberry Pi, menjadikannya cocok untuk pengembangan sistem portabel. *Framework* ini dapat berjalan baik di CPU maupun GPU, serta dilengkapi dengan alat bantu seperti *Tracer* dan *Visualizer* untuk memantau performa *pipeline*. Keunggulan lainnya adalah kemampuannya untuk melakukan sinkronisasi waktu yang presisi antar stream data, serta dukungan penuh terhadap integrasi dengan model *machine learning* eksternal.

Salah satu komponen yang paling sering digunakan dalam *MediaPipe* adalah modul *hand tracking*, yang dapat mendeteksi dan melacak 21 titik kunci (*landmark*) pada tangan secara *real-time*. Titik-titik ini mencakup sendi dan ujung jari serta pusat telapak tangan, dan direpresentasikan dalam koordinat tiga dimensi. Informasi landmark ini menjadi sangat penting untuk mengenali bentuk dan pola gerakan tangan, sehingga sangat relevan dalam konteks pengenalan bahasa isyarat, khususnya BISINDO. Gambar 3.3 memperlihatkan peta visual dari 21 titik *landmark* tangan versi *MediaPipe*, yang menjadi acuan dalam proses ekstraksi fitur gerakan untuk klasifikasi gestur secara otomatis



Gambar 3. 3 . Peta *MediaPipe Hand Landmarks* (Sumber: AI Google Dev)

Dalam penelitian ini, *MediaPipe* digunakan untuk mengekstraksi fitur berupa posisi tangan dari input video yang ditangkap oleh kamera. Data koordinat landmark kemudian digunakan sebagai fitur masukan (*input features*) untuk algoritma *machine learning* seperti *Random Forest* dalam mengenali isyarat statis, serta *Long Short-Term Memory* (LSTM) untuk isyarat dinamis. Arsitektur *MediaPipe* yang ringan namun akurat menjadikannya solusi ideal untuk dikombinasikan dengan Raspberry Pi, menciptakan sistem penerjemah bahasa isyarat BISINDO yang portabel, efisien, dan *real-time* (Lugaresi et al., 2019).

3.2.6. Tensorflow

TensorFlow merupakan *framework open-source* dari Google yang dirancang untuk mendukung komputasi numerik berskala besar dalam konteks *machine learning*. *Framework* ini memodelkan perhitungan sebagai struktur data berbasis *computational graph*, yang memungkinkan optimasi eksekusi melalui pemetaan otomatis *node-node* graf ke berbagai unit komputasi, baik itu CPU, GPU, maupun distribusi antar *node* dalam sebuah *cluster* (Shukla, N., dan Fricklas, K., 2018).

3.2.7. Joblib

Joblib adalah pustaka *Python* yang menyediakan mekanisme pemrosesan terstruktur secara efisien melalui konsep *pipelining* yang ringan. Pustaka ini mendukung *caching* fungsi secara otomatis ke disk untuk menghindari komputasi ulang, serta memungkinkan eksekusi paralel yang sederhana, sehingga cocok

digunakan dalam *workflow machine learning* dan pemrosesan data berskala besar (Faouzi, J., dan Janati, H., 2020).

3.2.8. Collections

Modul *collections* menyediakan berbagai tipe data kontainer khusus yang dirancang sebagai alternatif dari struktur data bawaan *Python* seperti *dict, list, set*, dan *tuple*. Tipe-tipe ini menawarkan fungsionalitas tambahan dan efisiensi yang lebih tinggi dalam kasus penggunaan tertentu, seperti pengurutan, penghitungan frekuensi, atau struktur data berorientasi *queue* dan *mapping* (Van Rossum, G. and Drake, F.L., 1995).

3.2.9. Pickle

Pickle adalah pustaka standar *Python* yang digunakan untuk melakukan serialisasi dan deserialisasi objek *Python* ke dalam format biner. Dengan kata lain, modul ini memungkinkan objek *Python* seperti *list, dictionary*, atau bahkan model *machine learning* disimpan ke dalam file dan dimuat kembali di lain waktu tanpa kehilangan struktur dan nilainya. Fitur ini sangat berguna dalam proses penyimpanan model, *caching data*, atau transfer objek antar sistem, terutama dalam *workflow machine learning* di mana hasil pelatihan model sering kali perlu disimpan dan digunakan kembali tanpa perlu dilatih ulang (Rostami et al., 2024).

3.2.10. Matplotlib

Matplotlib adalah pustaka grafik dalam *Python* yang digunakan untuk visualisasi data, dan merupakan bagian penting dalam ekosistem *data science Python*. *Library* ini memungkinkan pembuatan berbagai jenis grafik seperti *line chart, bar chart, scatter plot*, dan lain-lain dengan fleksibilitas tinggi. *Matplotlib* terintegrasi dengan baik bersama pustaka lain seperti *NumPy*, *Pandas*, dan pustaka ilmiah lainnya, sehingga memudahkan proses eksplorasi, analisis, dan presentasi data dalam bentuk visual yang informatif dan interaktif (Sial et al., 2021).

3.2.11. Regular Expression (Regex)

Regular expression (atau *regex*) adalah pola yang digunakan untuk merepresentasikan sekumpulan *string* yang sesuai dengan kriteria tertentu. Modul *re* dalam *Python* menyediakan berbagai fungsi untuk memeriksa apakah suatu *string* cocok dengan pola *regex* yang diberikan, atau sebaliknya, apakah suatu pola *regex* cocok dengan *string* tertentu—yang secara konsep merupakan hal yang sama. *Regular expression* sangat berguna dalam pemrosesan teks, seperti pencarian pola, validasi format *data*, hingga ekstraksi informasi dari teks secara efisien (Van Rossum, G. and Drake, F.L., 1995).

3.2.12. OS (Operating System)

Modul *os* menyediakan antarmuka yang portabel untuk mengakses fungsionalitas yang bergantung pada sistem operasi. Melalui modul ini, pengguna dapat melakukan berbagai operasi sistem seperti mengelola file, direktori, dan variabel lingkungan, tanpa harus bergantung pada detail sistem operasi tertentu.

Jika hanya ingin membaca atau menulis file, bisa langsung menggunakan fungsi *open()*. Untuk manipulasi *path*, tersedia submodul *os.path*. Jika perlu membaca seluruh baris dari banyak *file* yang diberikan melalui *command line*, modul *fileinput* lebih sesuai. Untuk membuat *file* atau direktori sementara, dapat menggunakan modul *tempfile*, dan untuk operasi tingkat tinggi seperti menyalin atau memindahkan file dan folder, disarankan menggunakan modul *shutil* (Van Rossum, G. and Drake, F.L., 1995).

3.2.13. Imgaug

Imgaug merupakan *library open-source* yang populer digunakan untuk melakukan augmentasi citra dan dikembangkan menggunakan bahasa pemrograman *Python*. Pustaka ini memberikan dukungan komprehensif untuk beragam metode augmentasi dan memfasilitasi penggabungan teknik-teknik tersebut dengan mudah, baik secara *random* maupun paralel dengan memanfaatkan *multiple core* CPU. Selain untuk augmentasi citra, *imgaug* juga memiliki kemampuan untuk melakukan

augmentasi pada *keypoint* (*landmark*), *bounding boxes*, *heatmaps*, dan *segmentation maps*.

Library imgaug menawarkan berbagai jenis augmentasi, termasuk augmentasi kondisi cuaca seperti efek hujan, awan, salju, dan kabut. Dibandingkan dengan pustaka augmentasi lainnya, *imgaug* diakui sebagai salah satu yang paling komprehensif dan ekstensif. Metode augmentasi dalam *imgaug* diklasifikasikan menjadi 16 kelompok: *Arithmetic, Artistic, Blend, Blur, Color, Contrast, Convolutional, Edges, Flip, Geometric, ImgCorruptLike, PillLike*, *Pooling, Segmentation, Size*, dan *Weather*. Masing-masing kelompok terdiri dari beberapa metode spesifik. Sebagai ilustrasi, kelompok memuat enam teknik *blur* yang berbeda: *Gaussian, Median, Average, Bilateral, Motion Blur, dan Mean Shift*. Secara total, *imgaug* menyediakan 164 jenis augmentasi yang berbeda. (Amarù et al., 2023).

3.2.14. Long Short Term Memory (LSTM)

Long Short-Term Memory (LSTM) merupakan pengembangan dari arsitektur *Recurrent Neural Network* (RNN) yang dirancang untuk mengatasi kelemahan utama RNN standar dalam memproses data sekuensial, yaitu masalah *vanishing gradient*—di mana informasi dari input awal cenderung hilang atau memudar saat jaringan memproses urutan yang panjang (Graves, A., dan Graves, A,. 2012). Masalah ini menyebabkan jaringan kesulitan dalam mempertahankan konteks jangka panjang, sehingga tidak mampu mengenali pola yang membutuhkan informasi historis yang lebih jauh.

LSTM mengatasi masalah ini dengan memperkenalkan blok memori yang terdiri dari sel memori dan tiga jenis gerbang (*gates*):

1. *Input gate* (mengontrol kapan informasi baru dapat disimpan ke dalam memori.).

$$i^{(t)} = \sigma\left(W_i x^{(t)} + R_i y^{(t-1)} + p_i \odot c^{(t-1)} + b_i\right) \quad (1)$$

2. *Forget gate* (memutuskan informasi mana dari memori sebelumnya yang perlu dilupakan).

$$f^{(t)} = \sigma\left(W_f x^{(t)} + R_f y^{(t-1)} + p_f \odot c^{(t-1)} + b_f\right) \quad (2)$$

3. *Output gate* (mengatur kapan informasi dalam sel memori digunakan sebagai keluaran).

$$o^{(t)} = \sigma\left(W_o x^{(t)} + R_o y^{(t-1)} + p_o \odot c^{(t)} + b_o\right) \quad (3)$$

Ketiga gerbang ini bekerja secara bersamaan untuk memungkinkan jaringan menyimpan, mempertahankan, dan membuang informasi secara selektif, sehingga membuat LSTM sangat efektif dalam mempelajari ketergantungan jangka panjang dalam data sekuensial.

Dalam ulasan komprehensif yang dilakukan oleh Van Houdt et al. (2020), LSTM terbukti sangat efektif dan telah menjadi arsitektur utama dalam berbagai aplikasi, termasuk pengenalan suara, terjemahan otomatis, hingga sistem interaktif berbasis *AI* seperti *Google Translate* dan *Amazon Alexa*. LSTM juga banyak diadopsi dalam *domain computer vision* untuk mengenali pola dalam data visual sekuensial seperti video, gerakan tubuh, dan *gesture recognition*.

Dengan kemampuannya tersebut, LSTM menjadi arsitektur yang sangat tepat untuk digunakan dalam penelitian ini, khususnya dalam mengenali gestur dinamis dalam Bahasa Isyarat Indonesia (BISINDO). Untuk mendukung pengenalan gestur statis, digunakan algoritma *Random Forest*, yang menurut penelitian oleh Alexander dkk. (2023), menunjukkan performa akurasi yang tinggi dan waktu klasifikasi yang efisien dibandingkan dengan algoritma lainnya seperti KNN, SVM, dan *Decision Tree*. Oleh karena itu, dalam penelitian ini dikembangkan pendekatan gabungan *Random Forest* dan LSTM, dengan tujuan mengoptimalkan akurasi sistem penerjemah bahasa isyarat berbasis kamera yang mampu berjalan secara *real-time* di perangkat Raspberry Pi.

3.2.15. Random Forest

Random Forest merupakan metode *ensemble learning* berbasis pohon keputusan yang pertama kali diperkenalkan oleh Breiman (2001). Metode ini

membentuk sekumpulan (*forest*) pohon keputusan yang dilatih menggunakan teknik *bootstrap aggregating* (*bagging*), di mana setiap pohon dibangun dari *subset* data pelatihan yang diambil secara acak dengan pengembalian. Selain itu, pada setiap percabangan (*split*) dalam pohon, hanya sebagian acak dari fitur yang dipertimbangkan, guna menciptakan keragaman struktural antar pohon dalam *ensemble*. Tujuan utama pendekatan ini adalah untuk mengurangi *variance* tanpa meningkatkan bias, sehingga menghasilkan model yang lebih stabil dan memiliki performa yang lebih tinggi dibandingkan model *decision tree*.

Secara formal, misalkan $\mathcal{D}_n = \big((X_1, Y_1), \dots, (X_n, Y_n)\big)$ adalah data pelatihan dengan $X_i \in \mathbb{R}^p$ sebagai vektor fitur dan Y_i sebagai label target (untuk klasifikasi), maka *estimator Random Forest* didefinisikan sebagai rata-rata dari prediksi seluruh pohon:

$$m_{M,n}\big(x;\ominus_1, \dots, \ominus_{M,}\, \mathcal{D}_n\big) = \frac{1}{M}\sum_{j=1}^{M} m_n\big(x;\ominus_j, \mathcal{D}_n\big) \tag{4}$$

Biau dan Scornet (2016) menjelaskan bahwa model ini dapat dikaji lebih lanjut melalui versi yang disederhanakan, seperti *purely random trees*, untuk memahami karakteristik bias dan konvergensinya secara teoritis. Dalam varian ini, pemisahan dilakukan tanpa mempertimbangkan label data, yang memungkinkan analisis sifat konsistensi model terhadap fungsi target dalam regresi.

Keunggulan lain dari *Random Forest* adalah kemampuannya untuk mengevaluasi performa model secara internal menggunakan metode *Out-of-Bag* (OOB) *error*. Teknik ini memanfaatkan data yang tidak terambil selama proses *bootstrap* untuk menguji akurasi model, sehingga tidak memerlukan pembagian eksplisit antara data latih dan data uji. Selain itu, *Random Forest* juga menyediakan metrik *feature importance* yang memungkinkan interpretasi terhadap pengaruh relatif masing-masing fitur dalam proses prediksi. Dua metrik umum yang digunakan adalah *Mean Decrease in Impurity* (MDI) dan *Mean Decrease in Accuracy* (MDA), yang dapat memberikan wawasan penting dalam analisis variabel.

Biau dan Scornet (2016) juga membandingkan *Random Forest* dengan algoritma populer lainnya seperti SVM dan KNN. Mereka menekankan bahwa meskipun SVM unggul dalam masalah klasifikasi *margin* sempit dan KNN menawarkan pendekatan berbasis tetangga terdekat yang intuitif, *Random Forest* lebih unggul dalam hal ketahanan terhadap *noise*, efisiensi pada data berdimensi tinggi, serta kemampuannya menangani fitur numerik dan kategorikal secara bersamaan tanpa perlu normalisasi.

Secara keseluruhan, karakteristik non-parametrik, ketahanan terhadap *overfitting*, serta kemampuannya dalam memberikan estimasi generalisasi dan interpretabilitas menjadikan *Random Forest* sebagai model yang sangat sesuai untuk aplikasi klasifikasi dalam berbagai *domain*, termasuk dalam sistem pengenalan gestur bahasa isyarat berbasis *landmark* tangan.

3.3. Raspberry Pi

Raspberry Pi adalah komputer mini berukuran sebesar kartu kredit yang dikembangkan oleh *Raspberry Pi Foundation* di Inggris, dengan tujuan menyediakan perangkat komputasi murah dan portabel untuk pendidikan dan pengembangan teknologi. Raspberry Pi mendukung berbagai bahasa pemrograman seperti *Python*, C, C++, *Java*, dan lainnya, serta dapat menjalankan sistem operasi berbasis *Linux* seperti *Raspbian* (sekarang *Raspberry Pi OS*), *Debian*, dan lainnya.

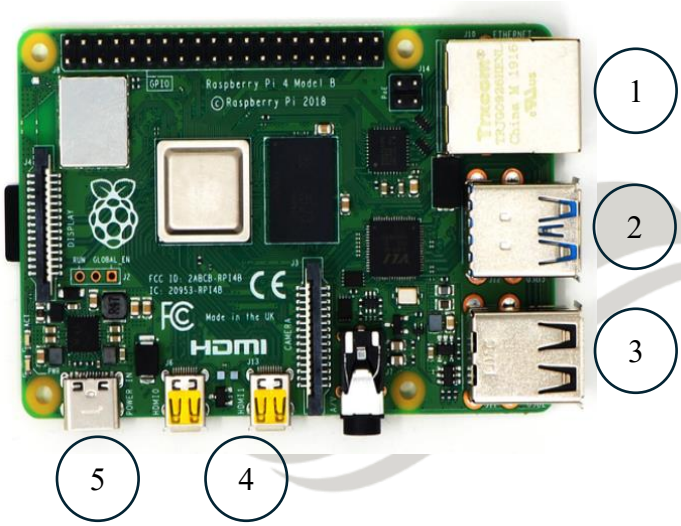
Seiring perkembangan teknologinya, Raspberry Pi telah dirilis dalam berbagai varian model dengan spesifikasi dan fungsi yang disesuaikan untuk kebutuhan yang berbeda. Beberapa di antaranya dirancang untuk penggunaan umum dan edukasi, seperti *Raspberry Pi* 3 dan 4, sementara model lain seperti *Raspberry Pi Zero* dan *Compute Module* lebih ditujukan untuk proyek *embedded* dan aplikasi industri. Tiap model memiliki perbedaan signifikan dalam hal kapasitas RAM, kecepatan prosesor, jumlah *port*, serta dukungan terhadap fitur seperti *Wi-Fi*, *Bluetooth*, dan antarmuka video. Tabel 3.1. berikut menyajikan ringkasan spesifikasi dari berbagai versi Raspberry Pi yang telah dirilis hingga saat ini.

Tabel 3. 1 Versi *Rasberry* Pi

Model	RAM	CPU	Port USB	HDMI / Display	Konektivitas
Raspberry Pi 400	4 GB	4 × 1.8 GHz	4 (2×USB 3.0)	Ya	WiFi & Bluetooth
Raspberry Pi 4 B	1/2/4/8 GB	4 × 1.5 GHz	4 (2×USB 3.0)	2×micro-HDMI	WiFi & Bluetooth
Compute Module 4	1/2/4/8 GB	4 × 1.5 GHz	Tidak ada	Tidak ada	Opsional
Raspberry Pi 3 B+	1 GB	4 × 1.4 GHz	4	Ya	WiFi & Bluetooth
Raspberry Pi 3 A+	512 MB	4 × 1.4 GHz	1	Tidak	WiFi & Bluetooth
Raspberry Pi Zero 2 W	512 MB	4 × 1 GHz	1 (micro)	Tidak	WiFi & Bluetooth
Raspberry Pi Zero W	512 MB	1 × 1 GHz	1 (micro)	Tidak	WiFi & Bluetooth
Raspberry Pi Zero	512 MB	1 × 1 GHz	1 (micro)	Tidak	Tidak ada
Raspberry Pi 2 B	1 GB	4 × 900 MHz	4	Ya	Tidak ada

Versi terbaru seperti *Raspberry Pi 4 Model B* memiliki spesifikasi cukup tinggi, seperti *prosesor quad-core* 1.5GHz, RAM hingga 8GB, konektivitas USB 3.0, Wi-Fi, Bluetooth 5.0, port HDMI ganda, serta slot microSD untuk penyimpanan. Dengan fitur tersebut, Raspberry Pi dapat menjalankan aplikasi komputasi ringan hingga menengah, termasuk sistem berbasis *machine learning* dan *computer vision*.

Meskipun begitu, Monk (2023) juga mengingatkan bahwa Raspberry Pi memiliki keterbatasan, terutama dalam hal manajemen suhu. *Raspberry Pi 4*, misalnya, rentan terhadap *overheating* saat digunakan untuk komputasi intensif dalam jangka waktu lama. Untuk mengatasi hal ini, disarankan penggunaan sistem pendingin aktif seperti kipas (*fan*), *heatsink*, atau casing berpendingin. Selain itu, pemilihan *SD card* berkecepatan tinggi (minimal *Class 10*) dan *power supply* 5V 3A juga penting untuk menjamin kestabilan sistem.



Gambar 3. 4 Rasberry Pi 4 model B (Sumber: Monk, 2022)

Gambar 3.4. memperlihatkan tampilan fisik Raspberry Pi yang digunakan dalam penelitian ini. Pada gambar tersebut, beberapa komponen utama diberi penomoran untuk mempermudah identifikasi dan penjelasan masing-masing fungsinya. Berikut ini adalah uraian dari komponen-komponen penting yang ditandai:

1. Gigabit Ethernet

   Port ini digunakan untuk konektivitas jaringan kabel (LAN) dengan kecepatan hingga 1 Gbps. Gigabit Ethernet sangat berguna untuk keperluan transfer data berkecepatan tinggi, komunikasi antarmuka dengan *server*, atau ketika Raspberry Pi digunakan dalam jaringan lokal yang stabil dan responsif.

   USB 3.0

2. Port USB 3.0 (berwarna biru) menawarkan kecepatan transfer data yang

jauh lebih tinggi dibandingkan USB 2.0, hingga 5 Gbps. Dalam penelitian ini, USB 3.0 dapat dimanfaatkan untuk menghubungkan kamera eksternal (*webcam*), *flashdisk*, atau perangkat penyimpanan lainnya untuk mempercepat pemrosesan dan penyimpanan data.

3. USB 2.0

   Port USB 2.0 digunakan untuk koneksi perangkat seperti *mouse, keyboard*, atau perangkat input-output lainnya yang tidak membutuhkan kecepatan transfer tinggi. Keberadaan USB 2.0 memungkinkan Raspberry Pi berfungsi layaknya komputer desktop dalam skala mini.1

4. Micro HDMI

   Raspberry Pi 4 dilengkapi dua port micro HDMI yang dapat digunakan untuk menampilkan output grafis ke layar monitor atau TV. Port ini mendukung output video hingga resolusi 4K. Dalam konteks pengujian sistem deteksi *gesture*, micro HDMI digunakan untuk menampilkan antarmuka visual dan hasil klasifikasi secara langsung di layar eksternal.

5. USB Type-C (*Power Supply*)

   Port ini berfungsi sebagai jalur utama untuk memasok daya ke Raspberry Pi. Dibandingkan versi sebelumnya yang menggunakan *micro* USB, *port* USB *Type-C* mampu menyediakan arus listrik yang lebih stabil dan cukup untuk mendukung komponen-komponen tambahan seperti kamera, sensor, atau layar tambahan.

Raspberry Pi sangat populer dalam pengembangan sistem tertanam (*embedded systems*), terutama karena harganya yang murah, ukuran kecil, dan fleksibilitas tinggi. Keunggulannya antara lain: dukungan terhadap banyak sensor dan perangkat eksternal melalui GPIO, kompatibilitas dengan berbagai jenis kode, serta dapat difungsikan sebagai komputer portabel. Dalam penelitian ini, *Raspberry Pi* digunakan sebagai platform utama untuk menjalankan sistem klasifikasi bahasa isyarat BISINDO secara *real-time*, karena kemampuannya yang cukup untuk memproses input dari kamera dan menjalankan model *machine learning* secara efisien.

Namun, Raspberry Pi juga memiliki keterbatasan, seperti tidak adanya

penyimpanan internal (mengandalkan SD *card*), kinerja grafis yang terbatas, serta potensi *overheating* jika digunakan dalam waktu lama tanpa pendingin tambahan. Meskipun demikian, kombinasi fleksibilitas, portabilitas, dan kemudahan pemrograman menjadikan Raspberry Pi sangat ideal untuk proyek-proyek inovatif berbasis *AI* dan pengolahan citra (Ghael et al., 2020).

Bab IV

Deskripsi Data dan Hasil Praktik Kerja Lapangan

4.1. Profil Partisipan

Berikut adalah data subjek deteksi bahasa isyarat BISINDO menggunakan input kamera pada Raspberry Pi yang terlampir pada Tabel 4.1.:

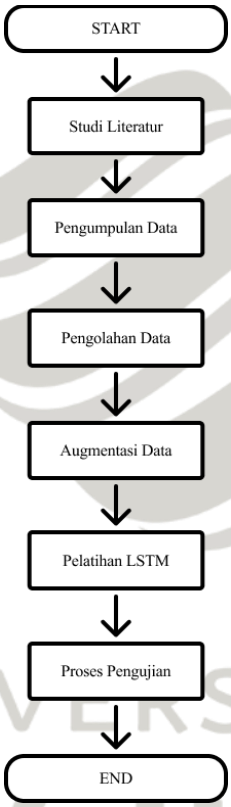
Tabel 4. 1 Tabel data subjek

No	Nama	Umur	Jenis Kelamin	Program Studi	Angkatan	Universitas
1	Benedict Chilwin Punarto	20 Tahun	Laki - Laki	Teknik Informatika	2022	Ma Chung
2	Clint Joy Perez Melody Mokosandi	21 Tahun	Laki - Laki	Teknik Informatika	2022	Ma Chung
3	Devinda Gusmananda	22 Tahun	Perempuan	Farmasi	2021	Ma Chung
4	Michael Gosali Tedjokusuma	20 Tahun	Laki - Laki	Teknik Informatika	2022	Ma Chung
5	Olfat Harits Alatas	20 Tahun	Laki - Laki	Teknik Informatika	2022	Ma Chung
6	Yoachim Yeremia Simatupang	22 Tahun	Laki - Laki	Teknik Informatika	2022	Ma Chung

4.2. Proses Pengerjaan

Praktik Kerja Lapangan dilaksanakan di Pusat Studi *Human-Machine Interaction* Universitas Ma Chung. Proyek PKL ini difokuskan pada pengembangan sistem deteksi bahasa isyarat menggunakan input kamera yang terintegrasi dengan

perangkat *Raspberry Pi 4 Model B*. Sistem ini menggabungkan dua pendekatan algoritma, yaitu *Random Forest* untuk klasifikasi gestur statis dan LSTM (*Long Short-Term Memory*) untuk gestur dinamis, dengan tujuan untuk mendukung pengembangan teknologi berbasis interaksi manusia-komputer, khususnya di bidang asistif dan edukatif.

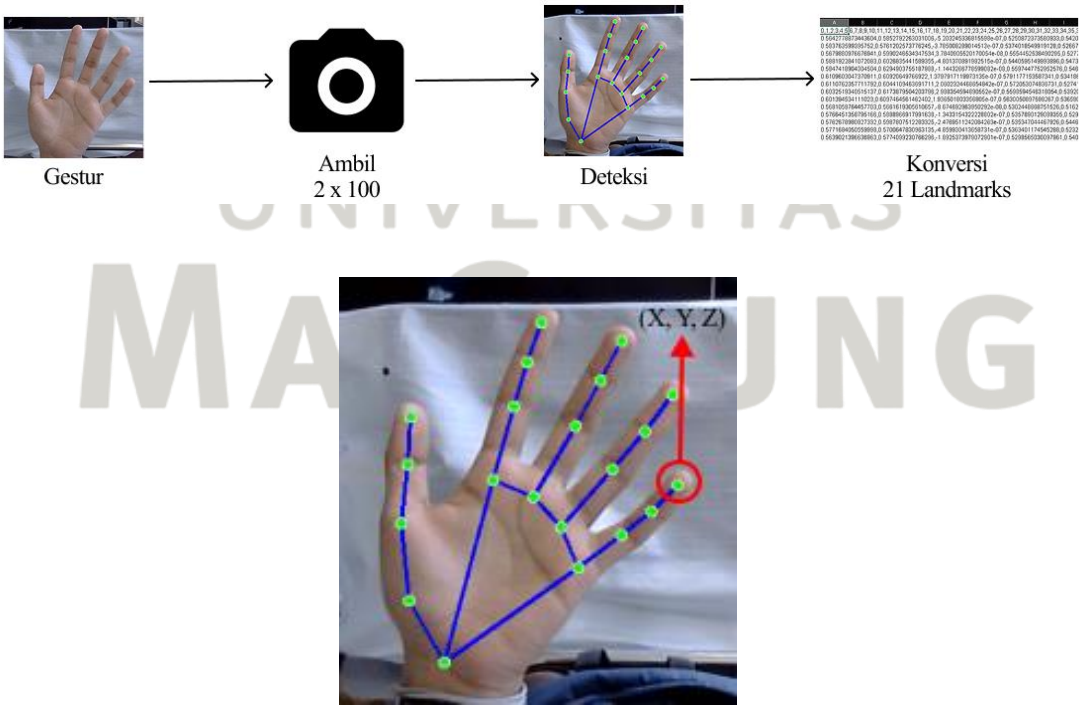


Gambar 4. 1 Tahapan pengerjaan

Tahapan awal pelaksanaan dimulai dengan melakukan studi literatur melalui jurnal ilmiah, buku referensi, serta sumber daring untuk memahami konsep dasar visi komputer, pengolahan citra, serta penerapan *machine learning* pada data gestur. Studi ini menjadi fondasi dalam merancang *pipeline* sistem, mulai dari akuisisi data kamera, ekstraksi fitur menggunakan *MediaPipe*, hingga pelatihan model klasifikasi. Alur keseluruhan tahapan pengerjaan dalam penelitian ini dapat dilihat pada Gambar 4.1., yang menggambarkan *flowchart* dari proses yang dilakukan secara bertahap mulai dari persiapan hingga implementasi sistem.

4.3. Pengumpulan Data

Proses pengumpulan data dilakukan dalam dua sesi pengambilan gambar. Pada setiap sesi, responden diminta untuk melakukan gestur tertentu yang kemudian direkam dalam bentuk citra sebanyak 100 gambar per gestur. Prosedur ini diulang untuk setiap kata yang akan digunakan sebagai kategori dalam sistem klasifikasi bahasa isyarat BISINDO. Pengambilan data dilakukan menggunakan kamera serta didukung oleh perangkat lunak seperti *OpenCV* untuk memfasilitasi proses akuisisi citra. Selama proses ini, partisipan diminta untuk melakukan gerakan bahasa isyarat secara berulang agar menghasilkan kumpulan gambar yang jelas, terang, dan merepresentasikan gerakan secara utuh. Gambar-gambar tersebut selanjutnya digunakan sebagai data latih untuk melatih model *Machine Learning*, sehingga penting untuk memastikan bahwa tiap kategori mencakup variasi gerakan yang realistis dan representatif. Alur proses pengumpulan data ini dapat dilihat pada Gambar 4.2.



Gambar 4. 2 Skema pengumpulan data dan titik-titik *landmark* yang akan dikonversi dalam format file CSV

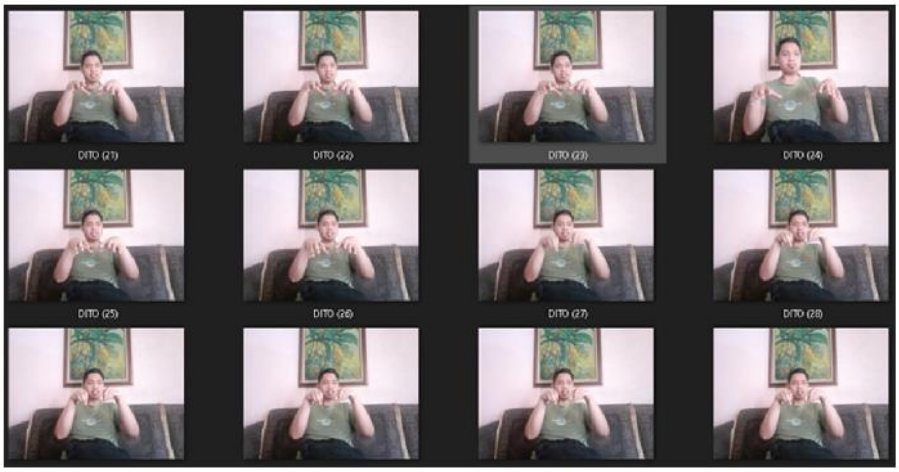
Setelah proses pengambilan gambar selesai, citra yang diperoleh akan diproses lebih lanjut untuk diekstraksi menjadi titik-titik koordinat tangan menggunakan fitur dari *MediaPipe Holistic*. Meskipun *MediaPipe Holistic* menyediakan deteksi pose, wajah, dan tangan secara bersamaan, dalam penelitian ini hanya bagian deteksi tangan yang dimanfaatkan. Sistem secara otomatis mendeteksi posisi tangan dan merepresentasikannya dalam bentuk koordinat tiga dimensi dari titik-titik landmark. Koordinat tersebut terdiri dari nilai x, y, dan z, di mana x menunjukkan posisi horizontal (kiri–kanan), y menunjukkan posisi vertikal (atas–bawah), dan z merepresentasikan kedalaman atau jarak relatif titik terhadap kamera. Nilai z ini bukan berasal dari kamera kedalaman, melainkan merupakan hasil estimasi bawaan dari model *MediaPipe* berdasarkan citra 2D. Gambar 4.2. menunjukkan hasil visualisasi deteksi *landmark* tangan, di mana setiap *node* merepresentasikan satu titik pada struktur tangan dan menyimpan nilai koordinat tiga dimensi tersebut.

Tabel 4. 2 Daftar 40 korpus harian

AIR	INI	ROTI	BERKATA
ANDA KAMU	ITU	SAYA	DALAM
ATAS	JADI	SEPERTI	DAN
ATAU	JIKA KALAU	SIAPA	KITA
BAWAH	KERJA	TAHUN	MEREKA
BELAJAR	LUAR	TAPI	MILK PUNYA
BISA	MAKAN	UNTUK	ORANG
DARI	MINUM	AKAN	SEMUA
DIA	NAIK	AMBIL	TAHU PAHAM
HANYA	PERGI	APA	TURUN

Data pada Tabel 4.2. berasal dari situs https://www.english-corpora.org/coca/, yang memuat hasil penelitian mengenai frekuensi penggunaan kata-kata dalam komunikasi berbahasa Inggris (WordFrequency, n.d.). Selain kata-kata tersebut,

partisipan juga diminta untuk memperagakan gestur untuk kategori angka (0 hingga 10) serta huruf alfabet dari A hingga Z.



Gambar 4. 3 Gestur “Ambil” (Sumber: Alexander, 2023)

Gambar 4.3. menampilkan contoh data yang diperoleh setelah melalui tahap pengumpulan data. Data yang dikumpulkan berupa citra hasil tangkapan kamera dari partisipan, dengan masing-masing partisipan menyumbangkan sebanyak 200 gambar untuk setiap gestur. Total terdapat 77 kategori gestur yang mencakup huruf alfabet dan angka. Dengan melibatkan lima partisipan dan jumlah gambar per gestur yang sama, maka keseluruhan dataset yang dihasilkan mencapai 77.000 citra.

4.4. Ekstraksi Fitur

Pada tahap ini, proses ekstraksi fitur dilakukan untuk memperoleh representasi numerik dari setiap gestur yang nantinya digunakan sebagai input bagi model klasifikasi. Penelitian ini mengembangkan sistem yang sebelumnya telah dibangun oleh Alexander dkk (2023), di mana model *Random Forest* digunakan untuk mengklasifikasikan gestur statis dalam bahasa isyarat BISINDO. Model tersebut telah menunjukkan kinerja yang cukup tinggi untuk sebagian besar gestur yaitu yaitu dengan nilai akurasi, presisi, *f1*, serta *recall* sebesar 97,7% secara program dan 84% presisi secara *real time*., namun penelitian ini bertujuan untuk melakukan improvement

terhadap aspek pengenalan gestur dinamis yang belum dapat ditangani secara optimal oleh model sebelumnya.

Untuk keperluan tersebut, langkah awal yang dilakukan adalah melakukan evaluasi terhadap model *Random Forest* yang dikembangkan oleh Alexander dkk (2023) guna mengidentifikasi jenis gestur apa saja yang belum dapat dikenali secara baik. Berdasarkan hasil pengujian, diketahui bahwa terdapat beberapa gestur statis yang masih mengalami kesalahan klasifikasi, namun sebagian besar yang tidak terdeteksi dengan baik merupakan gestur dinamis, antara lain: AKAN, AMBIL, APA, BERKATA, DALAM, DAN, KITA, MEREKA, MILIK/PUNYA, ORANG, R, SEMUA, TAHU/PAHAM, dan TURUN.

Dataset yang memuat gestur-gestur tersebut kemudian dipisahkan untuk ditangani secara khusus. Proses selanjutnya adalah melakukan ekstraksi fitur menggunakan pustaka *MediaPipe*, melalui modul *Holistic*, yang meskipun dirancang untuk mendeteksi pose tubuh secara menyeluruh, dalam penelitian ini hanya digunakan untuk mengambil koordinat tangan. *MediaPipe Holistic* menghasilkan 21 titik landmark tangan (masing-masing dengan koordinat x, y, z) dari setiap gambar. Data *landmark* ini kemudian dikonversi ke dalam format numerik dan disimpan dalam bentuk CSV agar dapat digunakan sebagai input dalam pelatihan model berbasis *Long Short-Term Memory* (LSTM).

```
results holistic.process(image_rgb)
frame_landmarks = []
if results.left_hand_landmarks:
for lm in results.left_hand_landmarks.landmark:
frame_landmarks.extend([lm.x, lm.y, lm.z])
if results.right_hand_landmarks:
for lm in results.right_hand_landmarks.landmark:
frame_landmarks.extend([lm.x, lm.y, lm.z])
```

Gambar 4. 4 Cuplikan Kode Ekstraksi Fitur Landmark Tangan

Kode ini berfungsi untuk mengambil koordinat *landmark* tangan dari gambar menggunakan *MediaPipe Holistic*. Setelah gambar diubah ke format RGB, fungsi

holistic.process digunakan untuk memproses gambar tersebut. Jika tangan kiri atau kanan berhasil terdeteksi, maka setiap titik dari 21 *landmark* akan diekstrak nilai koordinatnya (x, y, z) dan disimpan dalam *list frame_landmarks*. Proses ini menghasilkan 126 fitur numerik dari satu gambar, yang menggambarkan posisi tangan dan nantinya digunakan sebagai input untuk melatih model klasifikasi gestur.

Namun, saat dilakukan pelatihan awal menggunakan data asli sebanyak 200 gambar per gestur, model LSTM menunjukkan gejala *overfitting*, ditandai dengan perbedaan signifikan antara akurasi data latih dan validasi. Hal ini diduga kuat disebabkan oleh keterbatasan jumlah data, khususnya karena gestur dinamis memerlukan representasi sekuensial yang lebih bervariasi dan kompleks dibandingkan gestur statis.

Untuk mengatasi hal tersebut, dilakukan proses augmentasi data menggunakan pustaka *imgaug*. Proses augmentasi meliputi rotasi, pergeseran posisi, perubahan skala, hingga transformasi *afine* ringan yang tetap mempertahankan struktur logis dari gestur tangan. Hasil augmentasi berhasil meningkatkan jumlah dataset secara signifikan, yaitu menjadi 30.000 gambar per gestur, sehingga lebih representatif untuk pelatihan model LSTM.

Setelah augmentasi, dilakukan kembali proses ekstraksi fitur dari gambar hasil augmentasi menggunakan metode yang sama. Dataset hasil augmentasi yang telah dikonversi ke dalam bentuk koordinat landmark ini kemudian digunakan dalam proses pelatihan model LSTM.

4.5. Augmentasi Data

Pada tahap sebelumnya, proses pelatihan model LSTM untuk mengenali gestur dinamis menunjukkan gejala *overfitting*. Hal ini disebabkan oleh terbatasnya jumlah data untuk setiap kategori gestur, yaitu sekitar 200 gambar per gestur, yang tidak cukup untuk menangkap variasi gerakan tangan dinamis secara representatif. Untuk mengatasi permasalahan tersebut, dilakukan augmentasi data, yaitu proses memperbanyak dataset secara buatan dengan memodifikasi gambar asli menggunakan transformasi visual.

Augmentasi dilakukan menggunakan pustaka *imgaug*, yang menyediakan berbagai teknik augmentasi citra yang fleksibel dan efisien. Teknik augmentasi yang diterapkan mencakup *Gaussian blur*, penambahan *noise*, perubahan pencahayaan (*brightness dan contrast*), rotasi dan translasi posisi gambar, skala (*zoom in/out*), shearing, dan pembalikan horizontal (*flipping*). Tujuan dari variasi ini adalah untuk meniru kondisi pengambilan gambar nyata yang bervariasi, sehingga model yang dilatih menjadi lebih *robust* terhadap perubahan posisi tangan, sudut pandang, atau kualitas gambar.

```
1 augmenter = iaa.Sequential([
2 iaa.Sometimes(0.5, iaa.GaussianBlur(sigma=(0.0,2.0))),
3 iaa.Sometimes(0.5,iaa.AdditiveGaussianNoise(scale=(5,2
  0))),
4 iaa.Sometimes(0.5, iaa.Multiply((0.7, 1.3))),
5 iaa.Sometimes(0.5, iaa.LinearContrast((0.6, 1.4))),
6 iaa.Affine(
7 rotate=(-25, 25),
8 translate_percent={"x": (-0.1, 0.1), "y": (-0.1,
  0.1)},
9 scale=(0.85, 1.15),
10  shear=(-10, 10)),
11  iaa.Fliplr(0.5)])
```

Gambar 4. 5 Kode untuk Augmentasi Data

Cuplikan kode tersebut merupakan konfigurasi augmentasi citra menggunakan pustaka *imgaug*, yang disusun dalam urutan *Sequential*. Beberapa transformasi yang diterapkan antara lain: *blur* (baris 2), penambahan *noise* (baris 3), perubahan kecerahan dan kontras (baris 4–5), serta transformasi geometris seperti rotasi, translasi, skala, dan *shear* melalui *Affine* (baris 6–10). Selain itu, gambar juga dapat diputar horizontal secara acak (*Fliplr*, baris 11). Masing-masing transformasi diterapkan secara acak dengan probabilitas tertentu, sehingga menciptakan variasi visual yang luas.

Setiap gambar dalam dataset asli di-augmentasi sebanyak lima kali, menghasilkan tambahan data sebanyak 30.000 gambar per gestur setelah proses augmentasi selesai. Proses ini dilakukan untuk setiap kategori gestur dinamis yang telah dipisahkan sebelumnya dari model *Random Forest* yang tidak mampu mendeteksinya secara akurat. Gambar 4.6. memperlihatkan kondisi dataset sebelum dilakukan augmentasi, yang jumlahnya masih terbatas, sedangkan Gambar 4.7. menampilkan peningkatan jumlah data secara signifikan setelah proses augmentasi dilakukan, memperlihatkan variasi visual baru yang dihasilkan dari transformasi citra. Proses ini bertujuan untuk memperkaya data latih dan meningkatkan kemampuan generalisasi model LSTM terhadap gestur dinamis.

Gambar 4. 6 Dataset Sebelum Di Augmentasi

Gambar 4. 7 Dataset Setelah Di Augmentasi

Dengan bertambahnya jumlah data hasil augmentasi, diharapkan model LSTM dapat dilatih dengan lebih baik dan mampu menggeneralisasi gestur dinamis secara lebih efektif. Dataset hasil augmentasi ini kemudian digunakan kembali dalam proses

ekstraksi fitur menggunakan *MediaPipe*, sebagaimana dijelaskan pada sub bab sebelumnya.

4.6. Proses Train Model Random Forest

Sebelum pelatihan model LSTM untuk mengenali gestur dinamis dilakukan, sistem terlebih dahulu menggunakan model *Random Forest* untuk mengklasifikasikan gestur statis. Model ini dirancang untuk mengenali gestur-gestur yang tidak melibatkan pergerakan dalam waktu (seperti angka, huruf, dan kata-kata statis). Proses pelatihan dilakukan dengan membagi dataset menjadi 80% data latih dan 20% data uji untuk memastikan evaluasi model dapat dilakukan secara objektif terhadap data yang belum pernah dilihat sebelumnya.

Model *Random Forest* dibangun menggunakan parameter default dari pustaka *Scikit-learn*, dengan arsitektur sebagai berikut:

- **n_estimators = 100**
  Jumlah pohon keputusan yang digunakan dalam ensemble.
- **criterion = "gini"**
  Kriteria pemilihan split pada setiap node berdasarkan indeks Gini.
- **max_depth = None**
  Kedalaman pohon tidak dibatasi.
- **min_samples_split = 2**
  Minimal dua sampel diperlukan untuk memisahkan node.
- **min_samples_leaf = 136**
  Minimal jumlah sampel agar sebuah node menjadi leaf.
- **max_features = "sqrt"**
  Jumlah fitur yang dipertimbangkan pada tiap split diambil akar dari total fitur.

```
1   from sklearn.ensemble import RandomForestClassifier
2   random_forest = RandomForestClassifier()
3   random_forest.fit(X_train, y_train)
```

Gambar 4. 8 Cuplikan Kode Pelatihan Random Forest

Baris pertama berfungsi mengimpor kelas *RandomForestClassifier* dari modul *ensemble* pada pustaka *Scikit-learn*, yang digunakan khusus untuk klasifikasi. Pada baris kedua, dibuat objek *random_forest* sebagai instansiasi dari model *Random Forest* tanpa menyetel parameter eksplisit, sehingga seluruh pengaturan menggunakan nilai default. Kemudian, model dilatih melalui metode *fit*(), yang menerima *X_train* sebagai fitur dan *y_train* sebagai label untuk menyesuaikan model terhadap pola data yang tersedia. Setelah model berhasil dilatih, hasilnya disimpan dalam file bernama *RandForest.pkl*. File ini berisi objek model yang telah dilatih dan siap digunakan kembali dalam sistem klasifikasi bahasa isyarat secara real-time, tanpa perlu dilakukan pelatihan ulang.

Model *Random Forest* ini digunakan secara khusus untuk mengenali gestur statis, dan kemudian dikombinasikan dengan model LSTM untuk menangani gestur dinamis, sehingga sistem dapat mengklasifikasikan berbagai jenis gestur BISINDO secara lebih komprehensif.

4.7. Proses Train Model LSTM

Setelah fitur landmark tangan berhasil diekstraksi, data tersebut digunakan untuk melatih model klasifikasi berbasis *Long Short-Term Memory* (LSTM). Model ini dirancang untuk mempelajari urutan waktu dari koordinat tangan yang merepresentasikan gestur dinamis, menjadikannya sangat relevan untuk mengenali urutan gerakan tangan dalam Bahasa Isyarat Indonesia (BISINDO).

Dataset yang digunakan telah melalui proses augmentasi dan dikonversi ke dalam bentuk sekuens berdimensi (samples, 30, 126), yaitu 30 *frame* per gestur dengan 126 fitur (koordinat x, y, z dari 21 titik di masing-masing tangan). Data kemudian dinormalisasi menggunakan *StandardScaler* dan labelnya dienkode menggunakan *LabelEncoder*, lalu diubah ke format *one-hot encoding* untuk keperluan klasifikasi multikelas.

Model LSTM dibangun menggunakan arsitektur bertingkat dua, dengan lapisan *dropout* di antaranya untuk mencegah *overfitting*, dan dua lapisan *dense* di bagian akhir. Model dilatih menggunakan fungsi aktivasi *softmax* dan *loss function*

categorical_crossentropy, dengan optimasi *Adam*. Untuk memaksimalkan hasil pelatihan, digunakan *EarlyStopping* untuk menghentikan pelatihan lebih awal jika performa validasi tidak meningkat, serta *ModelCheckpoint* untuk menyimpan bobot model terbaik berdasarkan nilai *val_loss*.

Berikut adalah cuplikan kode penting yang merepresentasikan arsitektur model dan pelatihan:

```
1 model = Sequential([
2 LSTM(128,return_sequences=True,input_shape=(30,126)),
3 Dropout(0.3),
4 LSTM(64),
5 Dropout(0.3),
6 Dense(64, activation='relu'),
7  Dense(y_categorical.shape[1], activation='softmax')])
8  model.compile(optimizer='adam',loss='categorical_cros
   sentropy', metrics=['accuracy'])
9  checkpoint = ModelCheckpoint('best_model_lstm.h5',
   monitor='val_loss', save_best_only=True)
10 history = model.fit(
11 X_train, y_train,
12 epochs=100,
13 batch_size=32,
14 validation_data=(X_val, y_val),
15 callbacks=[EarlyStopping(monitor='val_loss',patience=
   10, restore_best_weights=True),checkpoint]
```

Gambar 4. 9 Cuplikan Kode Pelatihan LSTM

Cuplikan kode tersebut merupakan proses pelatihan model *Long Short-Term Memory* (LSTM) menggunakan arsitektur berlapis. Model dibangun secara bertahap dimulai dari dua lapisan LSTM dengan 128 dan 64 *unit*, masing-masing diselingi

dengan *Dropout* sebesar 30% untuk mencegah *overfitting*. Setelah itu, output diarahkan ke lapisan *Dense* berisi 64 neuron dengan aktivasi *ReLU*, dan diakhiri dengan lapisan output *Dense* yang menggunakan aktivasi *softmax* untuk klasifikasi multikelas sesuai jumlah kategori dalam *y_categorical*. Model dikompilasi dengan optimizer *Adam* dan fungsi *loss categorical_crossentropy*, serta diukur menggunakan metrik akurasi. Proses pelatihan dijalankan hingga 100 *epoch* dengan *batch size* 32 dan validasi terhadap data *X_val*. Untuk meningkatkan stabilitas pelatihan, diterapkan *EarlyStopping* yang otomatis menghentikan pelatihan jika tidak terjadi peningkatan pada *val_loss* selama 10 *epoch*, serta *ModelCheckpoint* yang menyimpan bobot terbaik ke dalam file *best_model_lstm.h5*.

Hasil pelatihan menunjukkan performa yang sangat baik, dengan akurasi pengujian mencapai 98%. Berdasarkan *classification report*, sebagian besar gestur memiliki nilai *precision*, *recall*, dan *f1-score* di atas 0.90. Hal ini menunjukkan bahwa model mampu mengklasifikasikan gestur dinamis dengan sangat baik meskipun terdapat perbedaan variasi antar partisipan dan frame. Pelatihan model ini menghasilkan beberapa file penting:

1. **best_model_lstm.h5**: Merupakan model LSTM terbaik yang disimpan selama pelatihan dan digunakan dalam sistem hand tracking untuk klasifikasi *real-time* gestur dinamis.
2. **label_encoder.pkl**: Digunakan untuk mengonversi output numerik model kembali ke bentuk label teks (misalnya “AKAN”, “APA”, “TURUN”).
3. **scaler.pkl**: Digunakan untuk menormalkan input koordinat saat sistem dijalankan, agar skala nilai sesuai dengan kondisi saat model dilatih.

Model *best_model_lstm.h5* akan dimuat oleh sistem hand tracking pada saat dijalankan. Ketika pengguna memperagakan gestur, sistem akan menangkap 30 *frame*, mengekstrak titik-titik koordinat tangan menggunakan *MediaPipe*, melakukan normalisasi menggunakan *scaler.pkl*, lalu mengirim data ke model LSTM untuk diprediksi. Output dari model akan berupa probabilitas kelas, yang dikonversi kembali

ke nama gestur menggunakan *label_encoder.pkl*, dan ditampilkan sebagai hasil pengenalan gerakan dinamis.

Dalam pelatihan model LSTM, sejumlah *hyperparameter* digunakan untuk mengontrol proses pembelajaran dan memengaruhi performa akhir model. *Hyperparameter* ini ditentukan sebelum pelatihan dimulai dan tidak diubah selama proses *training*. Berikut adalah penjelasan masing-masing *hyperparameter* yang digunakan dalam penelitian ini:

- **input_shape = (30, 126)**

  Input model adalah urutan 30 *frame* dengan 126 fitur per *frame*. Ini mencerminkan 30 langkah waktu (*frame*), masing-masing berisi 21 titik *landmark* pada dua tangan (kiri dan kanan), dengan 3 koordinat (x, y, z) untuk setiap titik.
- **LSTM(128, return_sequences=True)**

  Lapisan LSTM pertama memiliki 128 *unit* memori dan mengembalikan *output* sekuensial untuk setiap *timestep* agar bisa diteruskan ke LSTM berikutnya. Ukuran 128 dipilih untuk menyeimbangkan antara kapasitas pembelajaran dan efisiensi komputasi.
- **LSTM(64)**

  Lapisan LSTM kedua memiliki 64 unit. Tidak mengembalikan sekuens karena hasilnya langsung diteruskan ke lapisan *dense* untuk klasifikasi.
- **Dropout(0.3)**

  Dropout dengan rasio 30% diterapkan setelah masing-masing lapisan LSTM untuk mencegah *overfitting* dengan cara mengabaikan sebagian unit selama pelatihan.
- **Dense(64, activation='relu')**

  Lapisan *fully connected* dengan 64 *neuron* dan fungsi aktivasi *ReLU* digunakan untuk memperkuat non-linearitas sebelum tahap klasifikasi akhir.
- **Dense(output, activation='softmax')**

  Lapisan *output* menggunakan jumlah *neuron* sesuai jumlah kelas, dengan fungsi aktivasi *softmax* untuk mengubah skor menjadi probabilitas tiap kelas.

- **optimizer='adam'**

  Algoritma optimasi *Adam* dipilih karena efisien dan adaptif terhadap pembaruan bobot.
- **loss='categorical_crossentropy'**

  Fungsi loss ini digunakan untuk masalah klasifikasi multikelas dengan label *one-hot encoding*.
- **metrics=['accuracy']**

  Akurasi digunakan sebagai metrik utama untuk mengevaluasi kinerja model selama pelatihan dan validasi.
- **epochs=100**

  Maksimum pelatihan dijalankan selama 100 *epoch*. Namun, training dapat dihentikan lebih awal oleh *EarlyStopping*.
- **batch_size=32**

  Setiap iterasi pelatihan dilakukan pada batch berukuran 32 sampel.
- **EarlyStopping(patience=10)**

  Pelatihan akan berhenti secara otomatis jika tidak terjadi penurunan *loss* validasi selama 10 *epoch* berturut-turut.
- **ModelCheckpoint(monitor='val_loss')**

  Model terbaik disimpan berdasarkan nilai *loss* validasi terendah selama pelatihan, dan disimpan dalam file *best_model_lstm.h5*.

Hyperparameter ini dirancang untuk memberikan keseimbangan antara kapasitas model, generalisasi, dan efisiensi pelatihan, terutama mengingat bahwa sistem akan diimplementasikan pada perangkat seperti *Raspberry Pi* dengan keterbatasan sumber daya. Jika diperlukan, tuning hyperparameter lebih lanjut dapat dilakukan pada pengembangan sistem tahap berikutnya.

4.8. Evaluasi Model

Evaluasi model dilakukan untuk mengukur performa sistem dalam mengklasifikasikan gestur Bahasa Isyarat Indonesia (BISINDO), khususnya pada

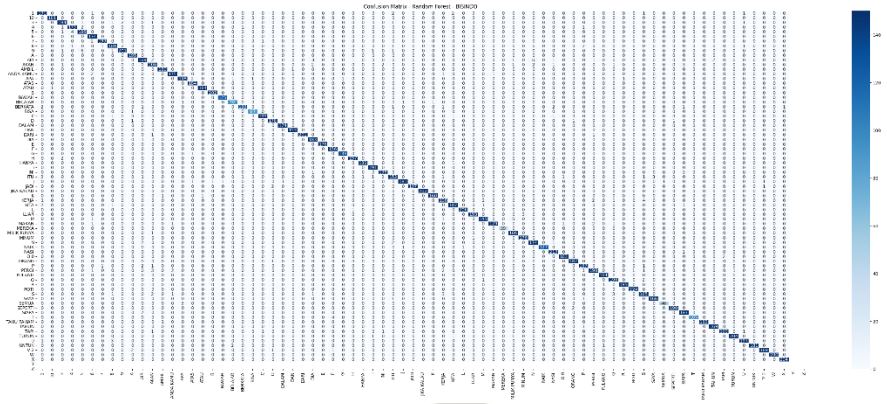
kategori gestur dinamis yang sebelumnya tidak berhasil dikenali secara akurat oleh model *Random Forest*. Dua pendekatan yang digunakan dalam penelitian ini adalah model *Random Forest* untuk gestur statis dan model *Long Short-Term Memory* (LSTM) untuk gestur dinamis.

Berdasarkan hasil pengujian, model *Random Forest* yang digunakan oleh penelitian sebelumnya (Alexander, 2023) menunjukkan akurasi yang sangat tinggi dengan nilai 0.98 untuk semua metrik evaluasi utama, yaitu *precision*, *recall*, dan *f1-score*. Ini menunjukkan bahwa model *Random Forest* sangat andal dalam mengenali gestur-gestur statis yang bentuknya cenderung konsisten dari satu frame seperti yang dilampirkan pada Tabel 4.3.

Tabel 4. 3 Gestur yang diuji oleh Random Forest

AIR	INI	ROTI
ANDA KAMU	ITU	SAYA
ATAS	JADI	SEPERTI
ATAU	JIKA KALAU	SIAPA
BAWAH	KERJA	TAHUN
BELAJAR	LUAR	TAPI
BISA	MAKAN	UNTUK
DARI	MINUM	DIA
HANYA	NAIK	PERGI

Evaluasi model *Random Forest* dilakukan dengan mengacu pada *confusion matrix* pada Gambar 4.10. yang dihasilkan setelah model diterapkan pada *dataset gesture* BISINDO. Dalam evaluasi ini, fokus analisis dikhususkan pada *gesture-gesture* yang bersifat statis, karena sejumlah gesture dinamis seperti "AKAN", "APA", "MEREKA", dan sejenisnya telah dialihkan ke model LSTM sehingga tidak dibahas dalam konteks evaluasi *Random Forest*.



Gambar 4. 10 Confusion Matrix Random Forest

Secara umum, hasil evaluasi menunjukkan bahwa model *Random Forest* memiliki performa klasifikasi yang baik terhadap sebagian besar gesture statis. Hal ini terlihat dari dominasi nilai prediksi benar pada elemen diagonal *confusion matrix*, yang mencerminkan tingkat akurasi yang tinggi. *Gesture* alfabet seperti "A", "B", "C", hingga "Z" mampu dikenali dengan konsisten, dan *gesture* dengan bentuk tangan yang khas seperti "MAKAN", "MINUM", atau "SAYA" juga diklasifikasikan secara akurat oleh model. Ini menunjukkan bahwa *Random Forest* cukup andal dalam mengenali pola-pola visual statis yang kuat dan konsisten antar pengguna.

Meskipun demikian, terdapat beberapa gesture yang kerap mengalami kekeliruan dalam klasifikasi, khususnya *gesture-gesture* yang memiliki bentuk tangan yang sangat mirip. Misalnya, gesture alfabet seperti "I" dan "J" atau "V" dan "U" cenderung sulit dibedakan oleh model karena hanya memiliki perbedaan kecil dalam representasi posisi jari. Selain itu, gesture angka seperti "1", "7", dan "9" juga menunjukkan adanya kebingungan klasifikasi karena kesamaan visual yang tinggi, terutama ketika dilakukan oleh subjek yang berbeda atau dalam pencahayaan yang kurang ideal.

Kesalahan klasifikasi yang terjadi dalam *confusion matrix* tersebar secara merata di berbagai *gesture* dan tidak terpusat hanya pada satu atau dua kelas. Hal ini mencerminkan bahwa model *Random Forest* memiliki ketahanan yang cukup merata terhadap *gesture-gesture* statis, meskipun tetap ada ruang perbaikan pada *gesture* tertentu yang menunjukkan konsistensi kesalahan lebih tinggi.

Penting pula dicatat bahwa dengan dialihkannya *gesture-gesture* dinamis ke model LSTM, beban klasifikasi gesture menjadi lebih terdistribusi dengan baik. *Random Forest* dapat lebih fokus pada gesture statis yang sesuai dengan karakternya, sementara *gesture* dinamis yang membutuhkan konteks waktu dan urutan gerakan lebih tepat ditangani oleh model sekuensial seperti LSTM. Pendekatan ini tidak hanya membantu meningkatkan akurasi klasifikasi tetapi juga memperkuat efektivitas sistem secara keseluruhan.

Secara keseluruhan, evaluasi menunjukkan bahwa model *Random Forest* sangat potensial untuk digunakan dalam klasifikasi gesture statis dalam bahasa isyarat BISINDO, terutama jika didukung dengan distribusi data yang seimbang, pencahayaan yang baik, dan proses pelabelan yang konsisten. Model ini memberikan landasan kuat bagi integrasi sistem pengenalan gestur berbasis kamera yang responsif dan akurat, khususnya dalam lingkungan yang membutuhkan interaksi pengguna secara *real-time*.

Tabel 4. 4 Gestur yang diuji oleh LSTM

AKAN	AMBIL	APA	BERKATA
MEREKA	KITA	DAN	DALAM
MILIK PUNYA	ORANG	R	SEMUA
TAHU PAHAM	TURUN		

Sementara itu, model LSTM yang dikembangkan dalam penelitian ini untuk mendeteksi gestur dinamis yang di perlihatkan di Tabel 4.4. berhasil mencapai akurasi keseluruhan sebesar 0.98, dengan nilai *precision* dan *recall* yang seimbang, sebagaimana ditunjukkan pada Tabel 4.5. berikut:

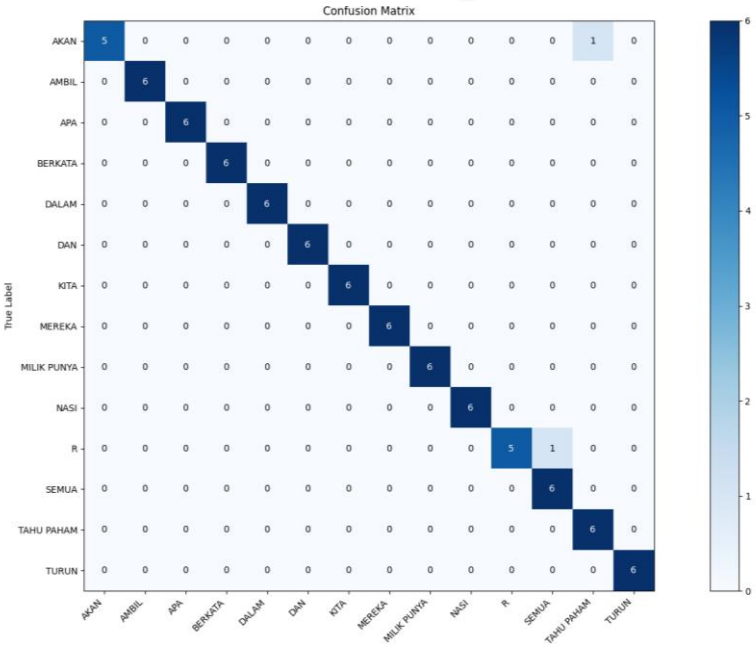
Tabel 4. 5 Hasil classification report dari model Random Forest dan LSTM

Model	f1	recall	precission	accuracy
Random Forest	0.98	0.98	0.98	0.98
LSTM	0.98	0.98	0.98	0.98

Berdasarkan *classification report* tersebut, dapat disimpulkan bahwa sebagian besar gestur berhasil dikenali dengan sangat baik, ditunjukkan oleh nilai *f1-score* yang tinggi (≥ 0.90) pada hampir semua kategori. Beberapa gestur seperti APA, BERKATA, MILIK/PUNYA, dan TURUN bahkan memperoleh nilai sempurna (1.00) untuk *precision*, *recall*, dan *f1-score*. Hal ini menunjukkan bahwa model LSTM sangat efektif dalam mengenali pola gerakan dinamis dari gestur-gestur tersebut.

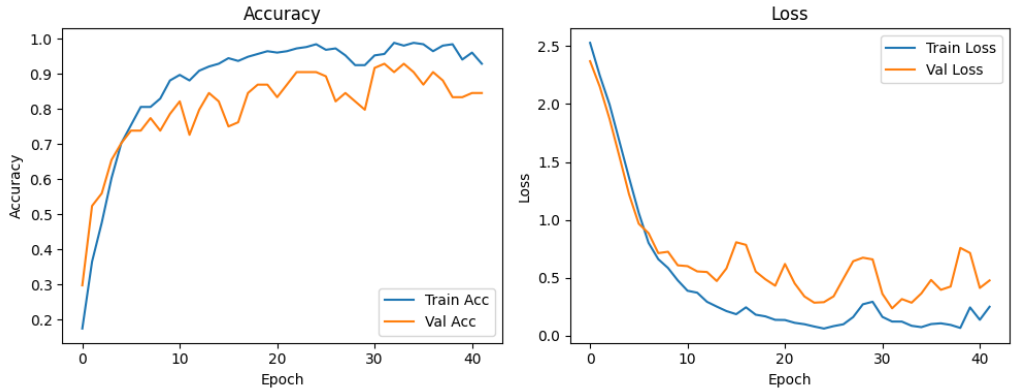
Namun, terdapat beberapa gestur yang performanya sedikit lebih rendah, seperti DAN dan KITA, yang memiliki *f1-score* di bawah 0.90. Rendahnya nilai ini umumnya disebabkan oleh kesalahan klasifikasi yang kemungkinan terjadi akibat kemiripan visual antar gestur atau variasi gerakan antar responden. Meskipun demikian, nilai rata-rata tetap berada di angka 0.98, yang menandakan bahwa secara keseluruhan model telah berhasil mencapai performa yang sangat baik dalam klasifikasi.

Untuk mendukung evaluasi kuantitatif yang telah dijelaskan sebelumnya, dilakukan juga analisis visual melalui dua jenis grafik untuk model LSTM, yaitu *confusion matrix* (Gambar 4.11.) dan kurva akurasi serta *loss* selama pelatihan (Gambar 4.12.).



Gambar 4. 11 Confusion Matrix Model LTSM

Gambar 4.11 menampilkan *confusion matrix* dari hasil prediksi model LSTM terhadap data uji. Sebagian besar nilai prediksi berada pada diagonal utama, yang menunjukkan bahwa model berhasil mengklasifikasikan gestur dengan benar sesuai label aslinya. Beberapa gestur seperti AMBIL, APA, BERKATA, DALAM, DAN, KITA, MEREKA, MILIK PUNYA, NASI, SEMUA, TAHU PAHAM, dan TURUN dikenali secara sempurna dengan akurasi 100%. Namun, terdapat sedikit misklasifikasi, seperti gestur AKAN yang salah diklasifikasikan sebanyak satu kali ke kelas R, serta gestur R yang satu kali diklasifikasikan sebagai TAHU PAHAM. Hal ini mengindikasikan bahwa terdapat kemiripan bentuk visual pada beberapa gestur tertentu, yang menyebabkan kebingungan dalam prediksi. Meskipun demikian, secara keseluruhan performa model tetap sangat baik karena distribusi kesalahan sangat kecil dan tidak memengaruhi mayoritas prediksi benar.



Gambar 4. 12 Kurva Akurasi dan Loss

Gambar 4.12 menunjukkan grafik akurasi dan *loss* selama proses pelatihan model LSTM selama sekitar 42 *epoch*. Kurva akurasi pelatihan (*Train Accuracy*) menunjukkan peningkatan yang konsisten dan stabil hingga mendekati 100%, sementara kurva akurasi validasi (*Val Accuracy*) cenderung mengikuti pola yang serupa namun dengan sedikit fluktuasi. Hal ini mengindikasikan bahwa model mampu mempelajari pola dari data latih dengan baik, dan juga menunjukkan performa yang stabil pada data validasi.

Sementara itu, grafik *loss* menunjukkan bahwa *Train Loss* menurun tajam di awal pelatihan dan terus mendekati nol, menandakan bahwa kesalahan prediksi pada data pelatihan berhasil ditekan secara signifikan. *Validation Loss* juga menurun di awal, namun mulai berfluktuasi setelah titik tertentu menunjukkan gejala *overfitting* ringan, yaitu ketika model mulai terlalu menyesuaikan diri dengan data latih dan menurun kemampuannya dalam generalisasi.

Untuk mengatasi hal ini, digunakan teknik *early stopping* dengan parameter *patience*=10, yang secara otomatis menghentikan pelatihan jika tidak ada peningkatan pada *val_loss* selama 10 *epoch* berturut-turut. Dengan demikian, meskipun pelatihan dijadwalkan hingga 100 *epoch*, model terbaik disimpan pada saat *val_loss* mencapai titik minimum, yaitu sekitar *epoch* ke-33 hingga 35, sebelum fluktuasi signifikan mulai terjadi.

4.9. Implementasi Sistem Deteksi Gesture Real-Time

Sistem deteksi *gesture* ini dijalankan secara *real-time* menggunakan kamera eksternal (USB *webcam Logitech C270*) yang terhubung ke perangkat *Raspberry Pi 4 Model B*. Sistem dirancang untuk mengenali Bahasa Isyarat Indonesia (BISINDO) dengan menggabungkan dua model klasifikasi, yaitu *Random Forest* untuk mendeteksi gestur statis, dan *Long Short-Term Memory* (LSTM) untuk mendeteksi gestur dinamis.

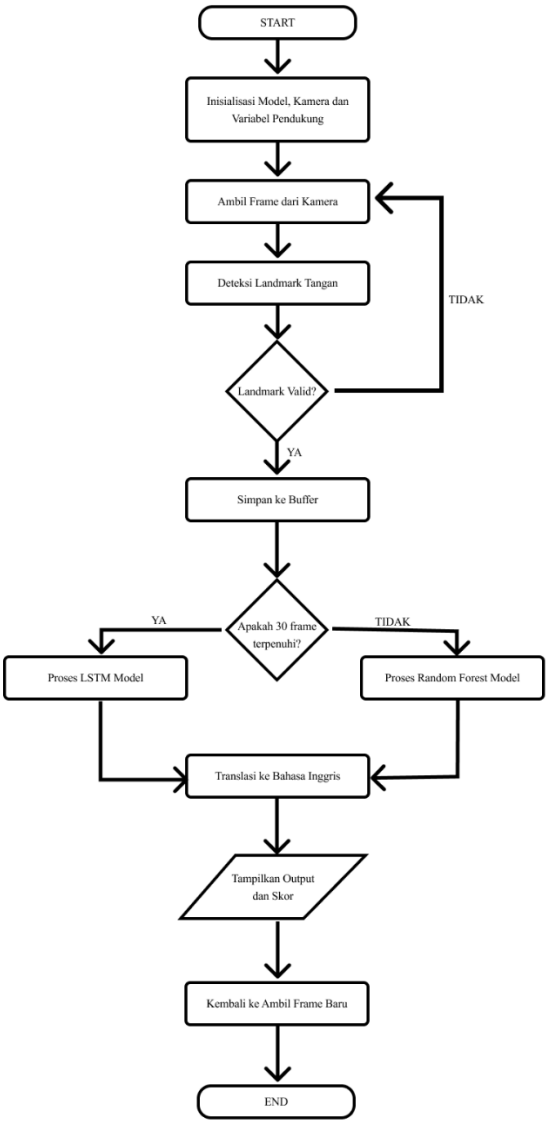
Ketika kamera aktif dan tangan pengguna berhasil terdeteksi, sistem akan mengekstraksi 126 fitur numerik dari titik-titik *landmark* tangan yang diperoleh menggunakan *MediaPipe Holistic*. Masing-masing tangan menghasilkan 21 titik koordinat dalam tiga dimensi (x, y, z), sehingga total setiap *frame* mencakup 126 fitur untuk dua tangan. Meskipun kamera yang digunakan hanyalah *webcam* biasa, *MediaPipe* dapat memperkirakan koordinat z melalui pemodelan spasial berbasis pembelajaran mesin yang telah dilatih sebelumnya, sehingga tidak memerlukan kamera *depth* khusus. Nilai z ini bersifat relatif dan digunakan untuk membantu mengenali posisi dan bentuk tangan secara tiga dimensi dalam ruang citra.

Data dari setiap *frame* ini kemudian disimpan dalam dua buffer berbeda, yaitu *static buffer* dan *sequence buffer*. *Static buffer* berfungsi untuk menyimpan satu frame

terkini dan langsung digunakan sebagai input bagi model *Random Forest* yang memang dirancang untuk mengenali pola gestur yang bersifat diam atau tidak berubah secara signifikan antar *frame*. Sebaliknya, *sequence buffer* menyimpan rangkaian 30 *frame* secara berurutan, yang kemudian digunakan sebagai *input* untuk model LSTM guna mendeteksi pola gerakan dalam gestur dinamis.

Pengambilan keputusan mengenai model mana yang digunakan untuk memproses *input* dilakukan secara adaptif oleh sistem. Jika *sequence buffer* telah berhasil mengumpulkan 30 *frame* dan telah melewati jeda waktu tertentu sejak prediksi LSTM terakhir, maka sistem akan mengaktifkan model LSTM untuk melakukan klasifikasi gestur dinamis. Jeda waktu ini ditetapkan untuk menghindari pemrosesan berulang pada gestur yang sama dalam waktu yang terlalu cepat, dan memberikan kesempatan sistem untuk mengumpulkan informasi gerakan yang cukup. Namun, jika kondisi tersebut belum terpenuhi misalnya jumlah frame belum cukup atau jeda waktu belum selesai maka sistem akan memanfaatkan model *Random Forest* secara langsung untuk memprediksi gestur statis menggunakan satu *frame* terkini. Untuk visualisasi cara kerja program nya terdapat pada Gambar 4.13.

Hasil klasifikasi gestur ditampilkan secara *real-time* di layar dalam bentuk teks, yang merepresentasikan hasil prediksi sistem terhadap isyarat tangan yang diperagakan oleh pengguna. Sebagaimana ditunjukkan pada Gambar 4.15., kata "THAT" muncul di sisi kiri atas tampilan sebagai hasil dari proses klasifikasi terhadap salah satu gestur BISINDO. Sistem secara otomatis menerjemahkan label gestur dari Bahasa Indonesia ke Bahasa Inggris dengan menggunakan padanan kata (*translation dictionary*) yang telah ditentukan dalam kode program. Proses translasi ini dapat dilihat pada Gambar 4.14., yang menampilkan cuplikan kode berisi daftar padanan kata. Dengan demikian, pengguna dapat memperoleh keluaran sistem dalam Bahasa Inggris secara langsung, yang mendukung fleksibilitas penggunaan untuk konteks internasional atau multilingual.



Gambar 4. 13 *Flowchart* program *Hand Tracking*

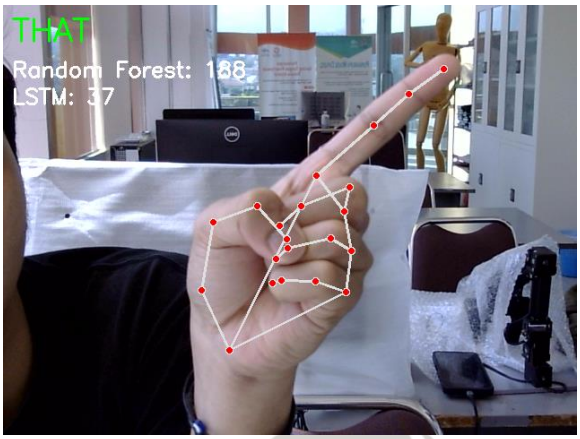
```
1  ind_to_eng = {
2  "AIR": "WATER", "AKAN": "WILL", "AMBIL": "TAKE",
   "ANDA KAMU": "YOU",
3  "APA": "WHAT", "ATAS": "UP", "ATAU": "OR", "BAWAH":
   "DOWN", "BELAJAR": "LEARN", "BERKATA": "SAY", "BISA":
   "CAN", "DALAM": "IN",
```

Gambar 4.14. a Cuplikan Kode Translasi Gestur

```
4  "DAN": "AND", "DARI": "FROM",
5  "DIA": "HE/SHE", "HANYA": "ONLY", "INI": "THIS",
   "ITU": "THAT", "JADI": "SO",
6  "JIKA KALAU": "IF", "KERJA": "WORK", "KITA": "WE",
   "LUAR": "OUTSIDE",
7  "MAKAN": "EAT", "MEREKA": "THEY", "MILIK PUNYA":
   "OWN/HAVE", "MINUM": "DRINK",
8  "NAIK": "UP/GO UP", "NASI": "RICE", "PERGI": "GO",
   "PULANG": "GO HOME",
9  "ROTI": "BREAD", "SAYA": "I/ME", "SEMUA": "ALL",
   "SEPERTI": "LIKE", "SIAPA": "WHO",
10 "TAHU PAHAM": "KNOW/UNDERSTAND", "TAHUN": "YEAR",
   "TAPI": "BUT", "TURUN": "DOWN/GO DOWN",
11 "UNTUK": "FOR"
12 }
```

Gambar 4.14. b Cuplikan Kode Translasi Gestur

Selain hasil klasifikasi dalam bentuk teks, sistem juga menampilkan skor dari kedua model klasifikasi yang digunakan, yaitu *Random Forest* dan LSTM. Angka seperti “Random Forest: 188” dan “LSTM: 37” mencerminkan jumlah prediksi yang telah dilakukan oleh masing-masing model sejak sistem dijalankan. Informasi ini berfungsi untuk memantau kontribusi model dalam proses klasifikasi: *Random Forest* digunakan untuk mengenali gestur statis, sedangkan LSTM untuk gestur dinamis. Dengan pendekatan kombinasi model ini, sistem mampu menangani berbagai jenis gestur secara fleksibel dan otomatis berdasarkan jenis *input* yang diterima dari kamera.



Gambar 4.15. Contoh *Output* Implementasi *Real Time*

4.10. Evaluasi real time

Setelah model LSTM terbaik diperoleh melalui proses pelatihan, tahap selanjutnya adalah melakukan evaluasi performa sistem secara *real-time*. Evaluasi ini bertujuan untuk menguji sejauh mana model yang telah dilatih mampu mengenali gestur dinamis BISINDO ketika dijalankan secara langsung melalui kamera menggunakan sistem *hand tracking* berbasis *Raspberry Pi*.

Pengujian dilakukan dengan mengundang enam subjek, yang seluruhnya merupakan mahasiswa aktif Universitas Ma Chung, sebagai partisipan uji coba sistem. Setiap subjek diminta untuk memperagakan seluruh kategori gestur yang terdiri dari:

- Gestur angka 0–10,
- Gestur alfabet A–Z,
- 14 gestur kata dinamis yang sebelumnya digunakan dalam pelatihan model LSTM, seperti AKAN, AMBIL,APA, BERKATA, DALAM, DAN, KITA, MEREKA, MILIK PUNYA, ORANG, SEMUA, TAHU PAHAM, TURUN, R,
- 27 gestur kata statis yang digunakan dalam pelatihan model *Random Forest*, seperti AIR, INI, ROTI, ANDA KAMU, ITU, SAYA, ATAS, JADI, SEPERTI, ATAU, JIKA KALAU, SIAPA, BAWAH, KERJA, TAHUN, BELAJAR, LUAR, TAPI, BISA, MAKAN, UNTUK, DARI, MINUM, DIA, HANYA, NAIK, PERGI.

Proses pengujian dilakukan dengan menggunakan USB *webcam Logitech C270* yang dipasang pada *Raspberry Pi 4 Model B*, yang berfungsi sebagai perangkat utama untuk menangkap gerakan tangan pengguna secara *real-time*. Kamera tersebut mendeteksi gerakan gestur, kemudian sistem mengekstrak titik-titik koordinat *landmark* tangan, melakukan normalisasi, dan memprosesnya melalui model LSTM yang telah dilatih sebelumnya. Selama pengujian *real-time* berlangsung, subjek berada pada jarak sekitar 1,5 meter dari kamera *webcam*, yang merupakan jarak optimal untuk memastikan seluruh tangan dapat terdeteksi secara utuh dan jelas oleh sistem, tanpa kehilangan presisi pada koordinat landmark yang dihasilkan. Jarak antara tangan dan kamera berpotensi memengaruhi akurasi deteksi, terutama jika tangan berada terlalu dekat (sehingga sebagian tangan terpotong dari *frame*) atau terlalu jauh (sehingga resolusi visual berkurang), sehingga sistem tidak dapat menangkap detail posisi jari dengan baik.

Untuk setiap gestur, masing-masing subjek melakukan 5 kali percobaan (*trial*), sehingga sistem dapat diuji terhadap konsistensi dan keakuratan dalam mengenali berbagai jenis *input*. Setelah setiap *trial*, sistem akan mencatat apakah prediksi berhasil atau gagal, dan jika gagal, dan akan dihitung rata rata keberhasilan pada setiap gestur dari setiap model. Data hasil pengujian ini kemudian dikompilasi dalam bentuk tabel untuk dianalisis lebih lanjut pada Tabel 4.6.

Tabel 4. 6 Hasil pengujian *real time**

Rata - Rata	
Random Forest	LSTM
76%	34%

*Data lengkap berada di lampiran B.2 dan B.3

Selama evaluasi real-time, sistem dijalankan menggunakan USB *webcam* yang terhubung ke *Raspberry Pi 4*. Meskipun *Raspberry Pi* memiliki keunggulan dari sisi portabilitas dan biaya rendah, hasil pengujian menunjukkan bahwa performa deteksi mengalami penurunan dibandingkan saat pengujian program secara offline. Salah satu

faktor utama penyebabnya adalah keterbatasan komputasi pada *Raspberry Pi*, yang membuat tampilan kamera tidak berjalan dengan frame rate tinggi. Hal ini menyebabkan model LSTM yang sensitif terhadap urutan gerakan menjadi kurang optimal dalam mengenali pola sekuensial secara tepat waktu.

Selain faktor teknis perangkat keras, beberapa gestur tidak terdeteksi secara akurat kemungkinan juga disebabkan oleh:

- Fleksibilitas dan struktur tangan setiap subjek yang berbeda, menyebabkan model kesulitan mencocokkan pola dengan data pelatihan yaitu mayoritas dalam gestur angka “9”.
- Beberapa gestur tertentu memang memiliki karakteristik gerakan yang kompleks, sehingga sulit ditiru dengan konsisten oleh subjek baru.
- Khusus untuk gestur "MILIK PUNYA" dan "DAN", hasil pengujian menunjukkan tingkat deteksi yang rendah. Hal ini disebabkan oleh bentuk gestur yang melibatkan kedua tangan saling bertumpuk, sedangkan MediaPipe sering kali hanya mendeteksi satu tangan secara dominan, terutama jika tangan yang satu menutupi tangan lainnya. Akibatnya, fitur landmark yang diperoleh menjadi tidak lengkap, sehingga model gagal mengenali gestur dengan akurat.

Hasil pengujian real-time yang dilakukan terhadap 77 gestur menunjukkan bahwa tingkat keberhasilan deteksi sistem secara keseluruhan berada pada angka 69%, berdasarkan lima kali percobaan untuk masing-masing gestur oleh enam subjek berbeda. Hasil ini mengindikasikan bahwa sistem bekerja cukup baik, namun performanya sangat bergantung pada karakteristik masing-masing gestur.

Beberapa gestur seperti angka 0–5, 7, 8, 10, serta alfabet seperti B, E, I, K, L, M, N, O, Q, S, U, V, W, X, Y, dan Z, dan kosakata seperti ATAS, ATAU, BAWAH, BELAJAR, DARI, KERJA, LUAR, dan SEPERTI, menunjukkan tingkat akurasi deteksi yang sangat tinggi, bahkan mencapai 100%. Hal ini dikarenakan gestur-gestur tersebut cenderung statis, memiliki bentuk tangan yang khas dan konsisten, serta dapat

dikenali dengan baik oleh algoritma *MediaPipe* dan model *Random Forest* tanpa ketergantungan pada sekuens gerakan.

Sebaliknya, sejumlah gestur lain menunjukkan tingkat deteksi yang sangat rendah. Misalnya, gestur AMBIL, JIKA KALAU, DAN, dan KITA sama sekali tidak terdeteksi (0%), sedangkan MILIK PUNYA, SAYA, MAKAN, MINUM, TAHU PAHAM, dan APA hanya terdeteksi dalam kurang dari 20% percobaan. Gestur-gestur ini umumnya merupakan gestur dinamis atau menggunakan dua tangan, yang menimbulkan tantangan dalam proses deteksi. Misalnya, MILIK PUNYA dan DAN dilakukan dengan kedua tangan saling bertumpuk, sehingga *MediaPipe* sering kali hanya menangkap satu tangan dominan, menyebabkan informasi fitur yang dikirim ke model menjadi tidak lengkap. Sementara itu, gestur seperti AMBIL, SAYA, atau MAKAN, meskipun tidak terlalu kompleks, dapat mengalami misklasifikasi karena kecepatan atau bentuk gerakan yang tidak konsisten antar subjek, serta keterbatasan *Raspberry Pi* dalam memproses data sekuensial secara stabil dengan model LSTM.

Selain itu, hasil deteksi untuk huruf alfabet seperti A, R, dan J juga cukup rendah, dengan akurasi masing-masing hanya sebesar 3%, 30%, dan 57%. Kemungkinan hal ini disebabkan oleh bentuk tangan yang kurang kontras, atau kemiripan fitur dengan gestur lain yang menyebabkan model salah klasifikasi, seperti huruf A yang sering terdeteksi sebagai gestur TAPI atau X.

Faktor-faktor seperti ketidakstabilan *frame* kamera saat menggunakan *Raspberry Pi*, variabilitas gaya gerakan antar subjek, serta kemungkinan landmark tidak terdeteksi sepenuhnya menjadi penyebab utama penurunan performa pada pengujian *real-time* ini. Hal ini menegaskan bahwa penerapan sistem pada perangkat dengan sumber daya terbatas perlu disesuaikan dengan batas kemampuan deteksi dan pengolahan model agar tetap memberikan hasil optimal.

Bab V
Penutup

5.1. Kesimpulan

Penelitian ini telah berhasil membangun sebuah sistem penerjemah Bahasa Isyarat Indonesia (BISINDO) berbasis kamera yang berjalan di perangkat *Raspberry Pi* dengan memanfaatkan dua pendekatan klasifikasi yang berbeda, yaitu algoritma *Random Forest* untuk gestur statis dan *Long Short-Term Memory* (LSTM) untuk gestur dinamis. Sistem dirancang untuk mengenali gestur dari data koordinat *landmark* tangan yang diperoleh melalui *MediaPipe Holistic*. Dari hasil pelatihan dan pengujian *offline*, kedua model yang digunakan menunjukkan performa yang sangat baik, dengan akurasi yang sama-sama mencapai 98%. Hal ini menunjukkan bahwa data yang digunakan mampu merepresentasikan gestur dengan cukup baik, dan model mampu mengenalinya secara konsisten pada kondisi terkontrol.

Namun demikian, hasil evaluasi secara *real-time* menunjukkan adanya penurunan performa yang cukup signifikan, dengan rata-rata keberhasilan deteksi sebesar 69%. Penurunan ini disebabkan oleh beberapa hal, antara lain keterbatasan performa komputasi dari *Raspberry Pi* yang menyebabkan pemrosesan data menjadi kurang lancar, serta kualitas tangkapan kamera yang tidak selalu stabil dan memengaruhi konsistensi *input* sekuens untuk model LSTM. Selain itu, faktor dari pengguna juga turut memengaruhi, terutama pada gestur yang sulit atau memerlukan koordinasi dua tangan. Dalam beberapa gestur, *MediaPipe* hanya mendeteksi satu tangan secara dominan, sehingga sistem tidak mendapatkan informasi penuh untuk mengenali gestur secara akurat. Beberapa gestur yang kompleks atau kurang umum juga terbukti sulit untuk ditiru secara tepat oleh subjek pengujian yang bukan merupakan pengguna aktif bahasa isyarat, sehingga menurunkan akurasi pada kondisi nyata.

5.2. Saran

Berdasarkan hasil yang diperoleh, terdapat beberapa hal yang dapat disarankan untuk pengembangan selanjutnya. Perlu dilakukan peningkatan pada aspek perangkat

keras, khususnya penggunaan kamera dengan kualitas *frame rate* yang lebih baik serta perangkat komputasi yang memiliki kapasitas lebih tinggi dari *Raspberry Pi*, agar proses deteksi *real-time* dapat berjalan lebih lancar dan responsif. Selain itu, pengumpulan data dari subjek yang lebih beragam, khususnya dari pengguna aktif bahasa isyarat, sangat dianjurkan agar model memiliki kemampuan generalisasi yang lebih kuat terhadap variasi bentuk dan gaya gerakan gestur yang berbeda. Dengan dua perbaikan tersebut, diharapkan sistem dapat dikembangkan lebih lanjut untuk digunakan secara lebih luas, termasuk dalam konteks aplikasi dunia nyata seperti alat bantu komunikasi untuk penyandang tunarungu.

DAFTAR PUSTAKA

Alexander, N., Widodo, R.B., & Swastika, W. Penggunaan Machine Learning Dalam Klasifikasi Bahasa Isyarat BISINDO Menggunakan Kamera. *Prosiding Seminar Nasional Universitas Ma Chung(SENAM)*, 2023, (pp. 11-26).

Aljabar, A., Suryani, D., and Prasetyo, E., 2020, 'BISINDO Sign Language Recognition Using CNN and LSTM', *Proceedings of the International Conference on Computer Engineering, Network and Intelligent Multimedia*, pp. 1–6.

Amarù, S., Marelli, D., Ciocca, G. and Schettini, R., 2023, 'DALib: A curated repository of libraries for data augmentation in Computer Vision', *Journal of Imaging*, 9(10), p. 232.

Assa, M.C., Kaunang, S.T.G. and Sugiarso, B.A., 2021, 'Aplikasi Interaktif Belajar Bahasa Isyarat Indonesia', *Jurnal Teknik Elektro dan Komputer*, 10(2), pp. 89–97.

Biau, G. and Scornet, E., 2016, A random forest guided tour. Test, 25(2), pp.197-227.

Borman, R.I., Priopradono, B. and Syah, A.R., 2017, 'Klasifikasi objek kode tangan pada pengenalan isyarat alphabet bahasa isyarat indonesia (bisindo)', *Seminar Nasional Informatika dan Aplikasinya (SNIA)*, pp. 1–4.

Bradski, G. and Kaehler, A., 2008, *Learning OpenCV: Computer vision with the OpenCV library*. Sebastopol, CA: O'Reilly Media, Inc.

Bradski, G. and Kaehler, A., 2016, *Learning OpenCV 3: Computer vision in C++ with the OpenCV library*. Sebastopol, CA: O'Reilly Media, Inc.

Breiman, L., 2001, 'Random Forests', *Machine Learning*, 45(1), pp. 5–32.

Dawson-Howe, K., 2014, *A practical introduction to computer vision with OpenCV*. John Wiley and Sons.

Fadlilah, N., Suryani, D., and Prasetyo, E., 2022, 'Modelling of Basic Indonesian Sign Language Translator Based on Raspberry Pi Technology', *Journal of Physics: Conference Series*, 1(1), pp. 1–6.

Faouzi, J. and Janati, H., 2020, 'pyts: A python package for time series classification', *Journal of Machine Learning Research*, 21(46), pp. 1–6.

Ghael, H.D., Solanki, L. and Sahu, G., 2020, 'A review paper on raspberry pi and its applications', *International Journal of Advances in Engineering and Management (IJAEM)*, 2(12).

Graves, A. and Graves, A., 2012, 'Long short-term memory', in *Supervised sequence labelling with recurrent neural networks*, pp. 37–45.

Gupta, P. and Bagchi, A., 2024, 'Introduction to NumPy', in *Essentials of Python for Artificial Intelligence and Machine Learning*, Cham: Springer Nature Switzerland, pp. 127–159.

Kramer, O., and Kramer, O., 2016, Scikit-learn. Machine learning for evolution strategies, 45-53.

Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., ... and Grundmann, M., 2019, Mediapipe: A framework for building perception pipelines.

Monk, S., 2022, Raspberry pi cookbook. " O'Reilly Media, Inc.".

Murni, R.K., Padlurrahman and Murcahyanto, H., 2024, 'Peran Vital Bahasa Isyarat Indonesia dalam Membangun Komunikasi dan Integrasi Sosial Anak Tuli', *Jurnal KIBASP*, 8(1), pp. 1–12.

Nugraheni, A.S., Husain, A.P. and Unayah, H., 2021, 'Optimalisasi penggunaan bahasa isyarat dengan sibi dan bisindo pada mahasiswa difabel tunarungu di prodi pgmi uin sunan kalijaga', *Jurnal Holistika*, 5(1), pp. 28–33.

Pujiati, D., 2019, 'Perbandingan Struktur antara Sistem Isyarat Bahasa Indonesia (SIBI) dengan Bahasa Isyarat Indonesia (BISINDO): Kajian Sintaksis', *Skripsi*, Universitas Pendidikan Indonesia.

Rostami, A.H., Haghighi, S., Sabouri, S. and Zolanvari, A., 2024, 'PyMilo: A Python Library for ML I/O

Shukla, N. and Fricklas, K., 2018, *Machine learning with TensorFlow*. Greenwich: Manning.

Sial, A.H., Rashdi, S.Y.S. and Khan, A.H., 2021, 'Comparative analysis of data visualization libraries Matplotlib and Seaborn in Python', *International Journal*, 10(1), pp. 277–281.

Van Houdt, G., Mosquera, C. and Nápoles, G., 2020, 'A review on the long short-term memory model', *Artificial Intelligence Review*, 53(8), pp. 5929–5955.

Van Rossum, G. (2007) 'Python programming language', *USENIX Annual Technical Conference*, 41(1), pp. 1–36.

Van Rossum, G. and the Python Development Team, (2017). *The Python Library Reference*, Release 3.6.1. Python Software Foundation.

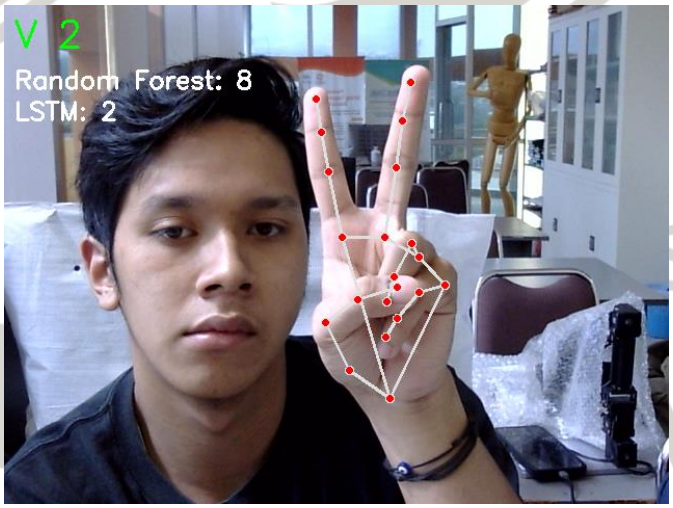
Wijaya, L.L. (2018) *Bahasa Isyarat Indonesia sebagai Panduan Kehidupan bagi Tuli*. Jakarta: Badan Pengembangan dan Pembinaan Bahasa.

Lampiran

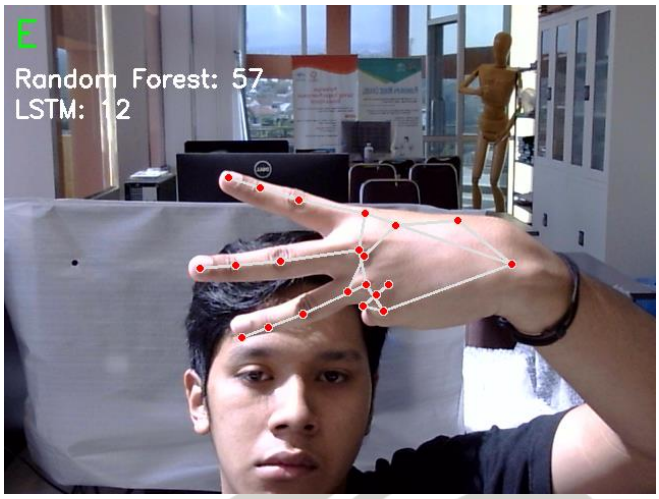
Lampiran A. Data Kegiatan PKL

Lampiran A.1 Dokumentasi Pengambilan Data Subjek

(a)



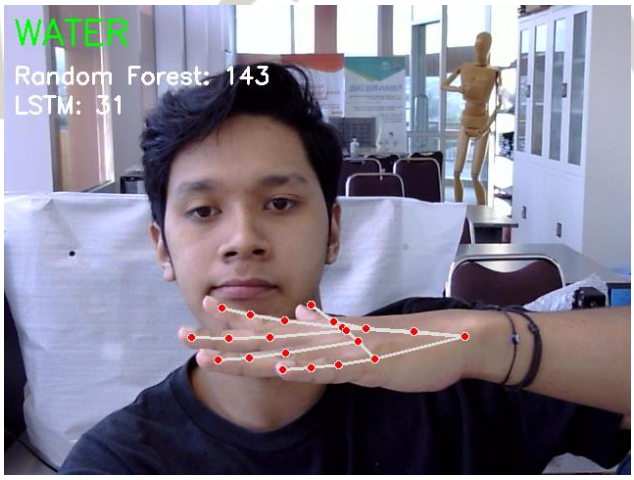
(b)


E
Random Forest: 57
LSTM: 12

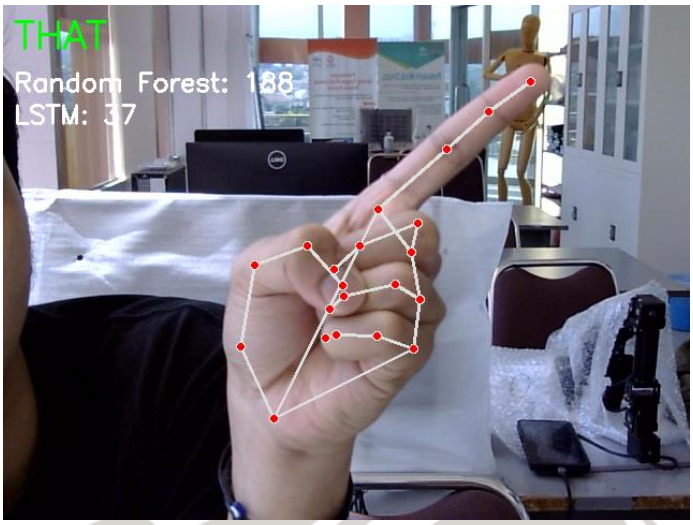
(c)


Z
Random Forest: 133
LSTM: 30

(d)

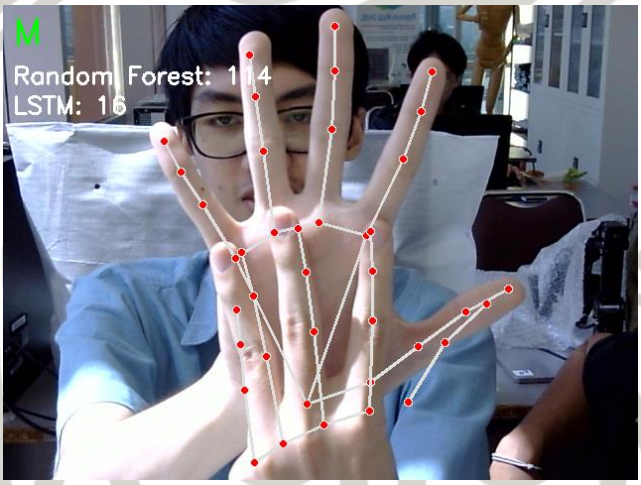

WATER
Random Forest: 143
LSTM: 31

(e)



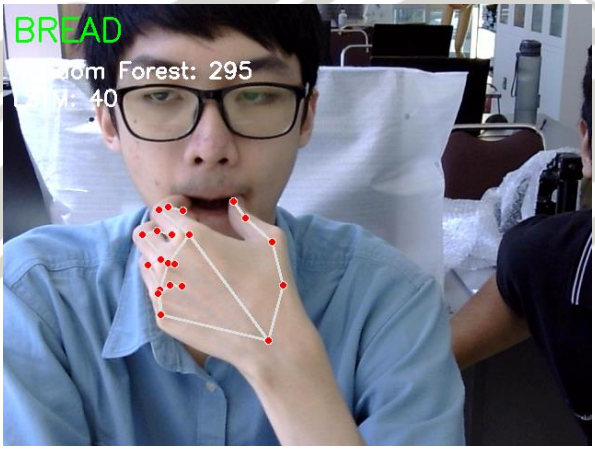
(f)

Gambar A.1.1 Data Subjek Olfat (a) Gestur “1” (b) Gestur “2” (c) Gestur “3” (d) Gestur “Z” (e) Gestur “AIR” (f) Gestur “ITU”.

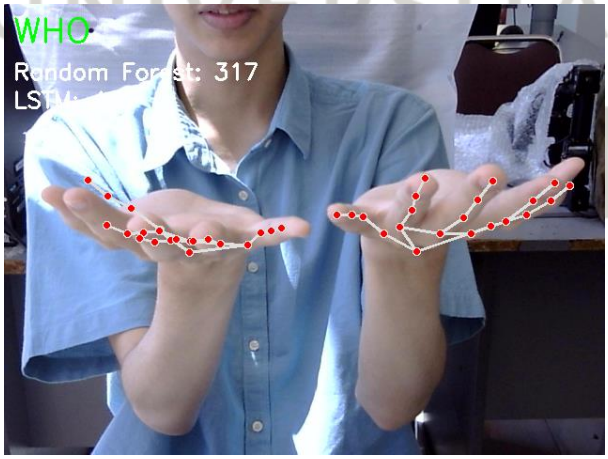


(a)

(b)

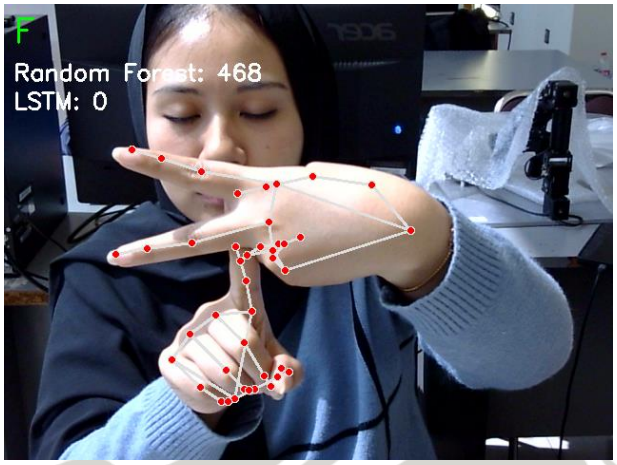


(c)

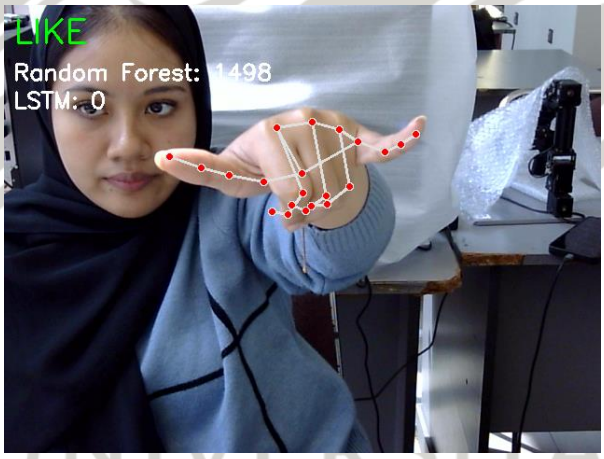


(d)

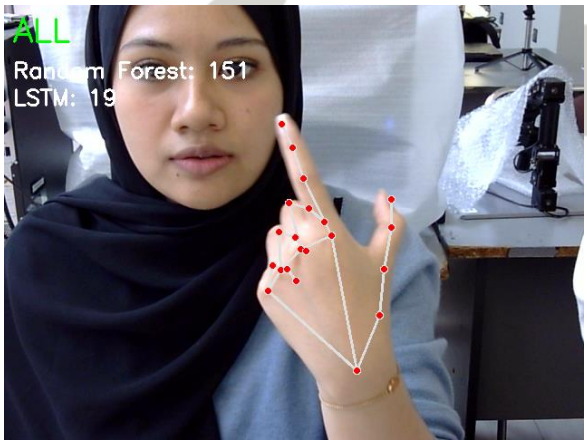
Gambar A.1.2 Data Subjek Chilwin (a) Gestur “M” (b) Gestur “W” (c) Gestur “ROTI” (d) Gestur “SIAPA”.



(a)



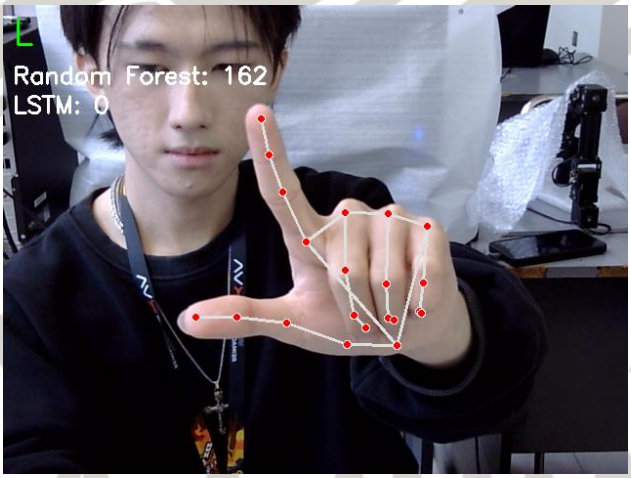
(b)



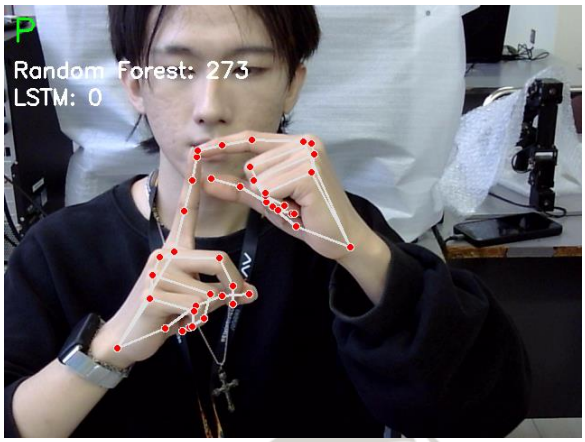
(c)

(d)

Gambar A.1.3 Data Subjek Devinda (a) Gestur “F” (b) Gestur “SEPERTI” (c) Gestur “SEMUA” (d) Gestur “TURUN”.

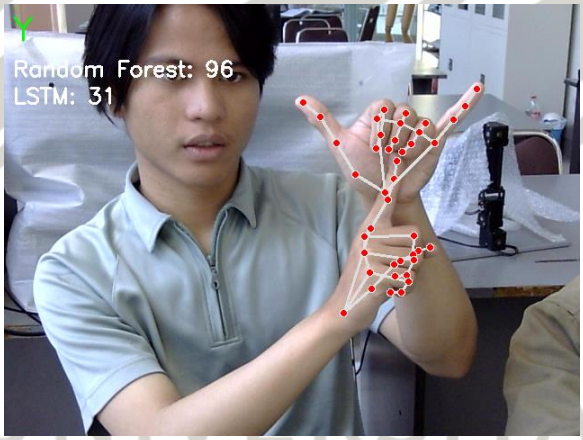


(a)

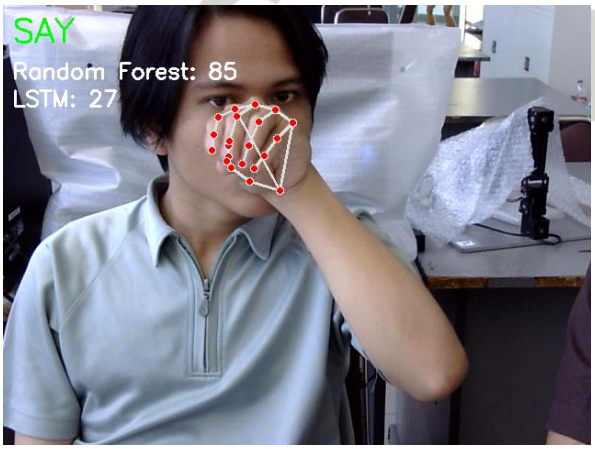


(b)

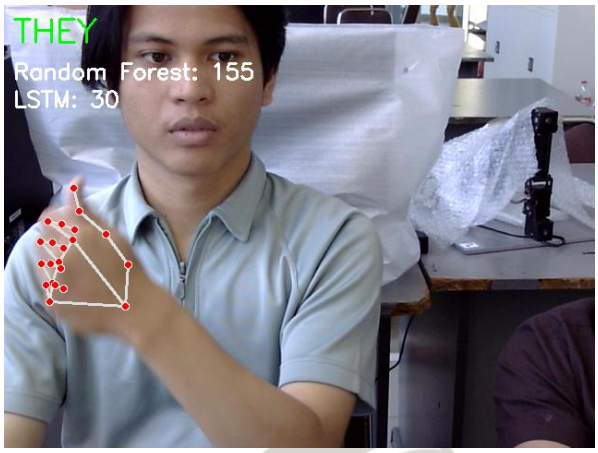
Gambar A.1.4 Data Subjek Michael (a) Gestur “L” (b) Gestur “P”.



(a)



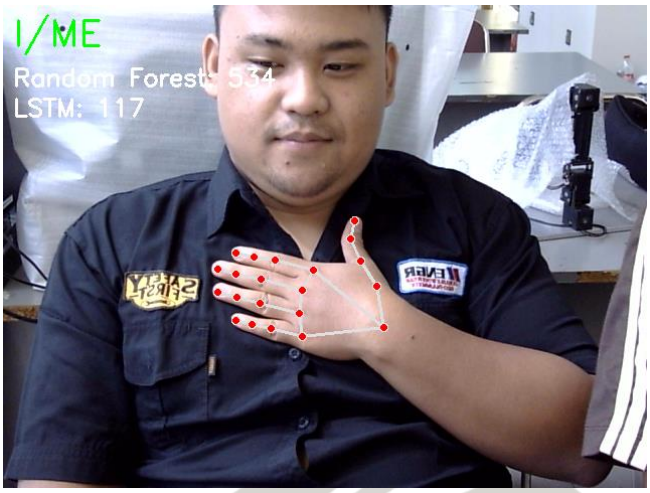
(b)



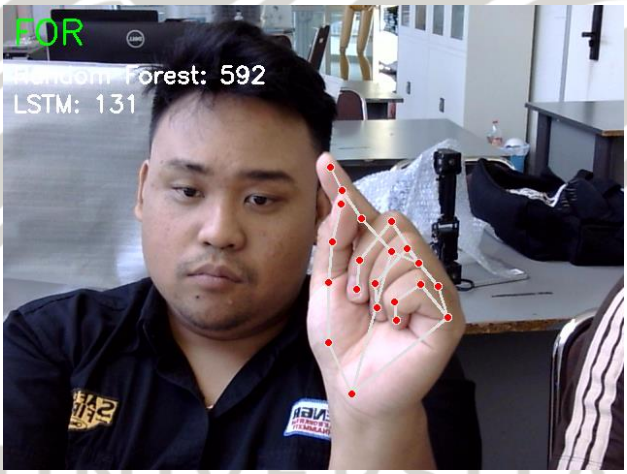
(c)

Gambar A.1.5 Data Subjek Clint (a) Gestur “Y” (b) Gestur “BERKATA” (c) Gestur “MEREKA”.

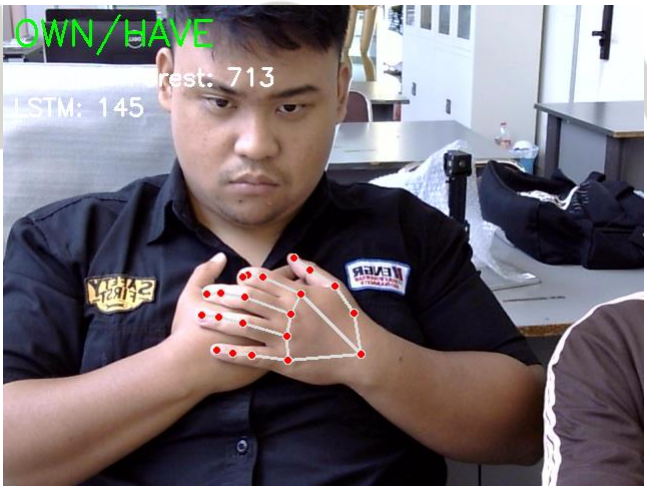
(a)



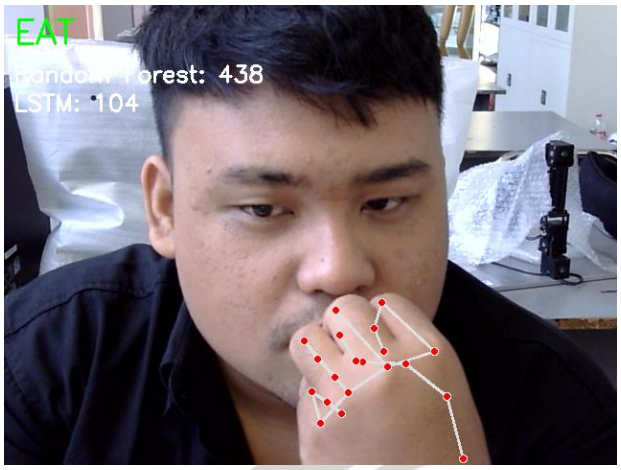
(b)



(c)



(d)



(e)

Gambar A.1.6 Data Subjek Yoachim (a) Gestur “PERGI” (b) Gestur “SAYA” (c) Gestur “UNTUK” (d) Gestur “MILIK PUNYA” (e) Gestur “MAKAN”.

Lampiran B. Hasil Kegiatan PKL

Lampiran B.1 Tabel Persentase Pengujian Presisi secara *Real Time*

Gestur	Persentase
0 O	100%
1	100%
2	100%
3	100%
4	100%
5	100%
6	53%
7	100%
8	100%
9	13%
10	100%
A	3%
B	93%
C	70%
D	60%
E	100%
F	77%

G	93%
H	67%
I	100%
J	57%
K	100%
L	100%
M	100%
N	100%
O	100%
P	90%
Q	100%
R	30%
S	100%
T	83%
U	93%
V	100%
W	87%
X	100%
Y	100%
Z	100%
AIR	93%
ANDA KAMU	50%
ATAS	100%
ATAU	100%
BAWAH	100%
BELAJAR	93%
BISA	63%
DARI	93%
DIA	47%
HANYA	67%
INI	100%
ITU	17%
JADI	100%
JIKA KALAU	0%
KERJA	93%
LUAR	97%
MAKAN	7%
MINUM	7%

NAIK	53%
PERGI	40%
ROTI	67%
SAYA	13%
SEPERTI	97%
SIAPA	10%
TAHUN	50%
TAPI	70%
UNTUK	83%
AKAN	73%
AMBIL	0%
APA	17%
BERKATA	37%
DALAM	50%
DAN	0%
KITA	0%
MEREKA	67%
MILIK PUNYA	13%
ORANG	83%
SEMUA	57%
TAHU PAHAM	13%
TURUN	33%
Rata - Rata	**69%**

Lampiran B.2 Tabel Persentase Pengujian Presisi secara *Real Time Random Forest*

Gestur	Persentase Random Forest
0 O	100%
1	100%
2	100%
3	100%
4	100%
5	100%
6	53%

7	100%
8	100%
9	13%
10	100%
A	3%
B	93%
C	70%
D	60%
E	100%
F	77%
G	93%
H	67%
I	100%
J	57%
K	100%
L	100%
M	100%
N	100%
O	100%
P	90%
Q	100%
S	30%
T	100%
U	83%
V	93%
W	100%
X	87%
Y	100%
Z	100%
AIR	100%
ANDA KAMU	93%
ATAS	50%
ATAU	100%
BAWAH	100%
BELAJAR	100%
BISA	93%
DARI	63%
DIA	93%

HANYA	47%
INI	67%
ITU	100%
JADI	17%
JIKA KALAU	100%
KERJA	0%
LUAR	93%
MAKAN	97%
MINUM	7%
NAIK	7%
PERGI	53%
ROTI	40%
SAYA	67%
SEPERTI	13%
SIAPA	97%
TAHUN	10%
TAPI	50%
UNTUK	70%
Rata - Rata	**76%**

Lampiran B.3 Tabel Persentase Pengujian Presisi secara *Real Time* LSTM

Gestur	**Persentase LSTM**
R	30%
AKAN	73%
AMBIL	0%
APA	17%
BERKATA	37%
DALAM	50%
DAN	0%
KITA	0%
MEREKA	67%
MILIK PUNYA	13%
ORANG	83%
SEMUA	57%
TAHU PAHAM	13%

TURUN	33%
Rata - Rata	**34%**

Lampiran B.4 Tabel Persentase Pengujian Presisi secara *Real Time* Subjek 1-6

* 0=Gagal, 1=Berhasil

GESTURE	SUBJECT	TRIAL					AVERAGE (%)	ACCURACY (%)
		1	2	3	4	5		
O 0	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
1	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
2	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
3	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
4	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	

5	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
6	1	1	1	1	1	1	100%	53%
	2	1	0	0	0	0	20%	
	3	0	0	0	0	0	0%	
	4	1	1	1	1	1	100%	
	5	0	0	0	0	0	0%	
	6	1	1	1	1	1	100%	
7	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
8	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
9	1	0	0	0	0	0	0%	13%
	2	0	0	0	0	0	0%	
	3	0	0	0	0	0	0%	
	4	1	0	0	0	0	20%	
	5	0	0	0	0	0	0%	
	6	1	1	0	1	0	60%	
10	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
A	1	0	0	0	0	0	0%	3%
	2	0	0	0	0	0	0%	

	3	0	0	0	0	0	0%	
	4	1	0	0	0	0	20%	
	5	0	0	0	0	0	0%	
	6	0	0	0	0	0	0%	
B	1	1	1	1	1	1	100%	93%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	0	0	60%	
C	1	1	1	1	1	1	100%	70%
	2	1	1	1	1	1	100%	
	3	0	0	0	0	1	20%	
	4	0	0	1	0	1	40%	
	5	1	1	1	1	1	100%	
	6	1	1	1	0	0	60%	
D	1	1	1	1	1	1	100%	60%
	2	0	0	1	0	0	20%	
	3	1	0	1	1	1	80%	
	4	0	0	0	0	1	20%	
	5	1	1	1	1	1	100%	
	6	1	1	0	0	0	40%	
E	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
F	1	1	1	1	1	1	100%	77%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	0	1	0	1	60%	
	5	0	0	0	0	0	0%	
	6	1	1	1	1	1	100%	
G	1	1	1	1	1	1	100%	93%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	

	5	1	0	0	1	1	60%	
	6	1	1	1	1	1	100%	
H	1	1	1	1	1	1	100%	67%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	0	0	0	0	0	0%	
	6	0	0	0	0	0	0%	
I	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
J	1	1	1	1	1	1	100%	57%
	2	1	1	1	1	1	100%	
	3	0	0	1	0	0	20%	
	4	1	1	1	1	1	100%	
	5	0	0	0	0	0	0%	
	6	0	0	0	0	1	20%	
K	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
L	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
M	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	

N	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
O	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
P	1	0	1	1	1	1	80%	90%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	0	0	1	60%	
Q	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
R	1	1	1	1	1	1	100%	30%
	2	0	0	0	0	0	0%	
	3	0	0	0	0	0	0%	
	4	1	0	1	0	0	40%	
	5	0	0	0	0	0	0%	
	6	0	0	0	1	1	40%	
S	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
T	1	1	1	1	1	1	100%	83%
	2	1	1	1	1	1	100%	

	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	0	0	0	0	0	0%	
U	1	0	0	1	1	1	60%	93%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
V	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
W	1	1	1	1	1	1	100%	87%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	0	0	1	0	0	20%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
X	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
Y	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
Z	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	

	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
AIR	1	1	1	1	1	1	100%	93%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	0	0	1	1	60%	
	6	1	1	1	1	1	100%	
ANDA KAMU	1	1	1	1	1	1	100%	50%
	2	0	0	0	0	0	0%	
	3	1	0	0	1	1	60%	
	4	0	0	0	0	0	0%	
	5	1	1	1	1	1	100%	
	6	0	0	1	1	0	40%	
ATAS	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
ATAU	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
BAWAH	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
BELAJAR	1	1	1	1	1	1	100%	93%
	2	1	1	1	1	1	100%	
	3	1	0	1	0	1	60%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	

BISA	1	0	0	1	1	1	60%	63%
	2	1	1	1	1	1	100%	
	3	1	0	0	1	0	40%	
	4	1	0	1	1	1	80%	
	5	0	0	0	0	0	0%	
	6	1	1	1	1	1	100%	
DARI	1	1	1	1	1	1	100%	93%
	2	1	1	1	1	1	100%	
	3	1	1	0	0	1	60%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
DIA	1	0	0	0	1	1	40%	47%
	2	1	1	1	1	1	100%	
	3	0	0	1	1	0	40%	
	4	0	0	0	0	0	0%	
	5	1	1	1	1	1	100%	
	6	0	0	0	0	0	0%	
HANYA	1	1	1	1	1	1	100%	67%
	2	1	1	1	1	1	100%	
	3	1	0	1	1	0	60%	
	4	1	0	0	0	0	20%	
	5	0	0	0	0	1	20%	
	6	1	1	1	1	1	100%	
INI	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
ITU	1	1	1	1	1	1	100%	17%
	2	0	0	0	0	0	0%	
	3	0	0	0	0	0	0%	
	4	0	0	0	0	0	0%	
	5	0	0	0	0	0	0%	
	6	0	0	0	0	0	0%	
JADI	1	1	1	1	1	1	100%	100%
	2	1	1	1	1	1	100%	

	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
JIKA KALAU	1	0	0	0	0	0	0%	0%
	2	0	0	0	0	0	0%	
	3	0	0	0	0	0	0%	
	4	0	0	0	0	0	0%	
	5	0	0	0	0	0	0%	
	6	0	0	0	0	0	0%	
KERJA	1	0	0	1	1	1	60%	93%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
LUAR	1	1	1	1	1	1	100%	97%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	0	1	1	1	1	80%	
	6	1	1	1	1	1	100%	
MAKAN	1	0	0	0	0	0	0%	7%
	2	0	0	0	0	0	0%	
	3	0	0	0	0	0	0%	
	4	0	0	1	0	1	40%	
	5	0	0	0	0	0	0%	
	6	0	0	0	0	0	0%	
MINUM	1	0	0	0	0	0	0%	7%
	2	0	0	0	0	0	0%	
	3	0	0	0	0	0	0%	
	4	0	0	1	0	0	20%	
	5	0	0	0	0	1	20%	
	6	0	0	0	0	0	0%	
NAIK	1	1	1	1	1	1	100%	53%
	2	1	1	1	1	1	100%	
	3	1	0	0	0	0	20%	
	4	1	0	0	0	0	20%	

	5	0	0	0	1	1	40%	
	6	0	1	0	1	0	40%	
PERGI	1	1	1	1	1	1	100%	40%
	2	1	0	1	0	0	40%	
	3	0	0	0	0	0	0%	
	4	1	0	1	0	1	60%	
	5	0	0	0	0	0	0%	
	6	1	1	0	0	0	40%	
ROTI	1	0	0	0	0	1	20%	67%
	2	1	1	1	1	1	100%	
	3	1	0	0	1	1	60%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	0	0	0	0	1	20%	
SAYA	1	0	0	0	0	0	0%	13%
	2	0	0	0	0	0	0%	
	3	0	0	0	0	0	0%	
	4	1	0	1	0	1	60%	
	5	0	0	0	0	0	0%	
	6	1	0	0	0	0	20%	
SEPERTI	1	1	1	1	1	1	100%	97%
	2	1	1	1	1	1	100%	
	3	0	1	1	1	1	80%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	1	1	1	1	1	100%	
SIAPA	1	0	0	0	0	0	0%	10%
	2	0	0	0	0	0	0%	
	3	0	0	0	0	0	0%	
	4	0	0	0	0	0	0%	
	5	0	0	1	1	1	60%	
	6	0	0	0	0	0	0%	
TAHUN	1	0	1	0	1	0	40%	50%
	2	1	1	1	1	1	100%	
	3	0	0	0	0	0	0%	
	4	0	0	1	0	0	20%	
	5	0	0	0	1	1	40%	
	6	1	1	1	1	1	100%	

TAPI	1	0	0	1	1	1	60%	70%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	0	0	1	1	1	60%	
	6	0	0	0	0	0	0%	
UNTUK	1	1	1	1	1	1	100%	83%
	2	1	1	1	1	1	100%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	1	1	1	1	1	100%	
	6	0	0	0	0	0	0%	
AKAN	1	0	1	0	1	1	60%	73%
	2	1	1	1	1	1	100%	
	3	1	0	1	0	1	60%	
	4	1	1	1	1	1	100%	
	5	1	0	1	0	0	40%	
	6	1	0	1	1	1	80%	
AMBIL	1	0	0	0	0	0	0%	0%
	2	0	0	0	0	0	0%	
	3	0	0	0	0	0	0%	
	4	0	0	0	0	0	0%	
	5	0	0	0	0	0	0%	
	6	0	0	0	0	0	0%	
APA	1	0	0	0	0	0	0%	17%
	2	1	1	1	1	1	100%	
	3	0	0	0	0	0	0%	
	4	0	0	0	0	0	0%	
	5	0	0	0	0	0	0%	
	6	0	0	0	0	0	0%	
BERKATA	1	0	0	0	0	0	0%	37%
	2	0	0	0	0	0	0%	
	3	1	0	1	0	0	40%	
	4	0	0	0	0	1	20%	
	5	0	1	1	1	1	80%	
	6	1	1	0	1	1	80%	
DALAM	1	0	0	1	1	1	60%	50%
	2	1	0	0	0	0	20%	

	3	0	0	0	0	0	0%	
	4	1	1	1	1	1	100%	
	5	0	0	1	1	1	60%	
	6	0	0	1	1	1	60%	
DAN	1	0	0	0	0	0	0%	0%
	2	0	0	0	0	0	0%	
	3	0	0	0	0	0	0%	
	4	0	0	0	0	0	0%	
	5	0	0	0	0	0	0%	
	6	0	0	0	0	0	0%	
KITA	1	0	0	0	0	0	0%	0%
	2	0	0	0	0	0	0%	
	3	0	0	0	0	0	0%	
	4	0	0	0	0	0	0%	
	5	0	0	0	0	0	0%	
	6	0	0	0	0	0	0%	
MEREKA	1	1	1	1	1	1	100%	67%
	2	1	0	1	0	0	40%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	0	0	0	0	0	0%	
	6	1	1	1	0	0	60%	
MILIK PUNYA	1	0	0	0	0	0	0%	13%
	2	0	0	1	0	0	20%	
	3	0	0	0	0	0	0%	
	4	0	1	0	0	1	40%	
	5	0	0	0	0	0	0%	
	6	0	0	0	0	1	20%	
ORANG	1	1	1	1	1	1	100%	83%
	2	1	0	0	1	0	40%	
	3	1	1	1	1	1	100%	
	4	1	1	1	1	1	100%	
	5	0	0	1	1	1	60%	
	6	1	1	1	1	1	100%	
SEMUA	1	0	0	0	1	1	40%	57%
	2	1	1	1	1	1	100%	
	3	1	1	0	1	0	60%	
	4	1	1	1	1	1	100%	

	5	0	0	0	0	1	20%	
	6	0	0	0	1	0	20%	
TAHU PAHAM	1	0	0	0	0	0	0%	13%
	2	0	0	0	1	0	20%	
	3	0	0	0	0	0	0%	
	4	1	0	1	0	0	40%	
	5	0	0	0	0	0	0%	
	6	1	0	0	0	0	20%	
TURUN	1	0	0	0	0	1	20%	33%
	2	0	1	0	1	0	40%	
	3	1	0	1	0	1	60%	
	4	1	1	1	1	0	80%	
	5	0	0	0	0	0	0%	
	6	0	0	0	0	0	0%	

Lampiran C. Source Code

Lampiran C.1 Source Code Ekstraksi Landmark

```
import os
import re
import cv2
import mediapipe as mp
import numpy as np
import pandas as pd
from tqdm import tqdm

# Path ke folder dataset
DATASET_PATH = 'E:/Dataset Penelitian Bahasa Isyarat Olfat 2025/Data
Nico/GABISA DETECT KATA/'
FRAMES_PER_GESTURE = 30

# Inisialisasi MediaPipe Holistic (static_image_mode=True agar tiap
gambar diproses independen)
mp_holistic = mp.solutions.holistic
holistic = mp_holistic.Holistic(static_image_mode=True)

data = []
labels = []
responden_list = []
```

```
# Loop setiap responden
for responden in tqdm(os.listdir(DATASET_PATH), desc="Responden"):
    responden_path = os.path.join(DATASET_PATH, responden)
    if not os.path.isdir(responden_path):
        continue

    # Loop setiap folder gestur
    for folder in os.listdir(responden_path):
        gesture_path = os.path.join(responden_path, folder)
        if not os.path.isdir(gesture_path):
            continue

        # Bersihkan nama label gesture: hapus "_AUG" dan angka di akhir
        cleaned_name = folder.replace('_AUG', '')
        gesture_label = re.sub(r'\d+$', '', cleaned_name)

        # Ambil gambar (maks 30 frame per gesture)
        images = sorted(os.listdir(gesture_path))[:FRAMES_PER_GESTURE]

        for img_name in images:
            img_path = os.path.join(gesture_path, img_name)
            image = cv2.imread(img_path)
            if image is None:
                continue

            # Convert BGR ke RGB
            image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
            results = holistic.process(image_rgb)
            frame_landmarks = []

            # Ekstrak landmark tangan kiri (21 titik x 3 koordinat)
            if results.left_hand_landmarks:
                for lm in results.left_hand_landmarks.landmark:
                    frame_landmarks.extend([lm.x, lm.y, lm.z])
            else:
                frame_landmarks.extend([0] * (21 * 3))

            # Ekstrak landmark tangan kanan (21 titik x 3 koordinat)
            if results.right_hand_landmarks:
                for lm in results.right_hand_landmarks.landmark:
                    frame_landmarks.extend([lm.x, lm.y, lm.z])
            else:
```

```
                frame_landmarks.extend([0] * (21 * 3))

            # Pastikan jumlah fitur = 126 (21*3*2)
            if len(frame_landmarks) == 126:
                data.append(frame_landmarks)
                labels.append(gesture_label)
                responden_list.append(responden)

holistic.close()

# Buat DataFrame dari data
df = pd.DataFrame(data)
df['label'] = labels
df['responden'] = responden_list

# Simpan ke CSV
df.to_csv('output_dataset_kata_lstm_126fitur_new.csv', index=False)
print('✅ Ekstraksi selesai. Data disimpan ke
output_dataset_kata_lstm_126fitur.csv')
```

Lampiran C.2 Source Code Augmentasi Data

```
import os
import cv2
from tqdm import tqdm
import imgaug.augmenters as iaa
import numpy as np

# Path dataset utama
DATASET_PATH = 'F:/Data Nico/GABISA DETECT KATA/'

# Augmentasi baru yang lebih bervariasi
augmenter = iaa.Sequential([
    iaa.Sometimes(0.5, iaa.GaussianBlur(sigma=(0.0, 2.0))),
    iaa.Sometimes(0.5, iaa.AdditiveGaussianNoise(scale=(5, 20))),
    iaa.Sometimes(0.5, iaa.Multiply((0.7, 1.3))),
    iaa.Sometimes(0.5, iaa.LinearContrast((0.6, 1.4))),
    iaa.Affine(
        rotate=(-25, 25),
        translate_percent={"x": (-0.1, 0.1), "y": (-0.1, 0.1)},
        scale=(0.85, 1.15),
        shear=(-10, 10)
    ),
```

```
    iaa.Fliplr(0.5)
])

def augment_folder(input_folder, output_folder, num_augments=3):
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    for file in os.listdir(input_folder):
        if not file.lower().endswith(('.jpg', '.jpeg', '.png')):
            continue

        img_path = os.path.join(input_folder, file)
        img = cv2.imread(img_path)
        if img is None:
            continue

        for i in range(num_augments):
            aug_img = augmenter(image=img)
            aug_filename = f"{os.path.splitext(file)[0]}_aug{i}.jpg"
            cv2.imwrite(os.path.join(output_folder, aug_filename),
aug_img)

# Loop tiap responden dan gestur
for responden in tqdm(os.listdir(DATASET_PATH), desc="Responden"):
    responden_path = os.path.join(DATASET_PATH, responden)
    if not os.path.isdir(responden_path):
        continue

    for gesture in os.listdir(responden_path):
        gesture_path = os.path.join(responden_path, gesture)
        if not os.path.isdir(gesture_path) or "_AUG" in gesture:
            continue

        output_path = os.path.join(responden_path, f"{gesture}_AUG5")
        augment_folder(gesture_path, output_path)
```

Lampiran C.3 Source Code Train LSTM

```
import numpy as np
import pandas as pd
import pickle
import matplotlib.pyplot as plt
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import classification_report, confusion_matrix

from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

# ------------------------------------
# CONFIG
# ------------------------------------
CSV_PATH = 'output_dataset_kata_lstm_126fitur_new.csv'
MODEL_SAVE_PATH = 'lstm_model_bisindo.h5'
ENCODER_PATH = 'label_encoder.pkl'
SCALER_PATH = 'scaler.pkl'
BEST_MODEL_PATH = 'best_model_lstm.h5'
FRAMES_PER_GESTURE = 30

# ------------------------------------
# Load dan siapkan data
# ------------------------------------
print("📂 Loading dataset...")
df = pd.read_csv(CSV_PATH)

# Ambil hanya fitur (exclude label dan responden)
X = df.iloc[:, :-2].values
y = df['label'].values

# Pastikan jumlah data bisa dibagi 30 frame per sample
num_samples = len(X) // FRAMES_PER_GESTURE
X = X[:num_samples * FRAMES_PER_GESTURE]
y = y[:num_samples * FRAMES_PER_GESTURE]

# Reshape ke (samples, 30, 126)
X = X.reshape(num_samples, FRAMES_PER_GESTURE, -1)
y = y[::FRAMES_PER_GESTURE]

# ------------------------------------
# Normalisasi
# ------------------------------------
print("🧹 Normalizing data...")
```

```
scaler = StandardScaler()
X_flat = X.reshape(-1, X.shape[2])
X_scaled = scaler.fit_transform(X_flat).reshape(X.shape)

# Simpan scaler
with open(SCALER_PATH, 'wb') as f:
    pickle.dump(scaler, f)

# -------------------------------------
# Encode label
# -------------------------------------
le = LabelEncoder()
y_encoded = le.fit_transform(y)
y_categorical = to_categorical(y_encoded)

# Simpan label encoder
with open(ENCODER_PATH, 'wb') as f:
    pickle.dump(le, f)

# -------------------------------------
# Split data
# -------------------------------------
print("🛠️ Splitting data...")
X_temp, X_testval, y_temp, y_testval = train_test_split(
    X_scaled, y_categorical, test_size=0.4, random_state=42,
stratify=y_encoded)

X_val, X_test, y_val, y_test = train_test_split(
    X_testval, y_testval, test_size=0.5, random_state=42)

X_train, y_train = X_temp, y_temp

# -------------------------------------
# Model
# -------------------------------------
print("🧠 Building model...")
model = Sequential([
    LSTM(128, return_sequences=True, input_shape=(FRAMES_PER_GESTURE,
X.shape[2])),
    Dropout(0.3),
    LSTM(64),
    Dropout(0.3),
    Dense(64, activation='relu'),
```

```
    Dense(y_categorical.shape[1], activation='softmax')
])
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
model.summary()

# -------------------------------------
# Training
# -------------------------------------
early_stop = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)
checkpoint = ModelCheckpoint(BEST_MODEL_PATH, monitor='val_loss',
save_best_only=True, verbose=1)

print("🚀 Training started...")
history = model.fit(
    X_train, y_train,
    epochs=100,
    batch_size=32,
    validation_data=(X_val, y_val),
    callbacks=[early_stop, checkpoint]
)

# -------------------------------------
# Evaluasi
# -------------------------------------
print("📊 Evaluating on test set...")
test_loss, test_acc = model.evaluate(X_test, y_test)
print(f"✅ Test Accuracy: {test_acc:.4f} | Test Loss: {test_loss:.4f}")

# -------------------------------------
# Confusion Matrix & Classification Report
# -------------------------------------
y_pred = model.predict(X_test)
y_pred_labels = np.argmax(y_pred, axis=1)
y_true_labels = np.argmax(y_test, axis=1)

print("\n📋 Classification Report:")
print(classification_report(y_true_labels, y_pred_labels,
target_names=le.classes_))

# -------------------------------------
# Simpan model akhir
```

```
# -------------------------------------
model.save(MODEL_SAVE_PATH)
print(f"💾 Model saved to: {MODEL_SAVE_PATH}")

# -------------------------------------
# Visualisasi
# -------------------------------------
plt.figure(figsize=(10, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Acc')
plt.plot(history.history['val_accuracy'], label='Val Acc')
plt.title('Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.title('Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()

# -------------------------------------
# Confusion Matrix (Matplotlib-only)
# -------------------------------------
cm = confusion_matrix(y_true_labels, y_pred_labels)
labels = le.classes_

plt.figure(figsize=(12, 10))
plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.colorbar()

tick_marks = np.arange(len(labels))
plt.xticks(tick_marks, labels, rotation=45, ha='right')
plt.yticks(tick_marks, labels)

# Tampilkan nilai di dalam kotak
```

```
thresh = cm.max() / 2.
for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        plt.text(j, i, format(cm[i, j], 'd'),
                 ha="center", va="center",
                 color="white" if cm[i, j] > thresh else "black")

plt.tight_layout()
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.show()
```

Lampiran C.4 Source Code Train Random Forest

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report, f1_score, recall_score, precision_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

df = pd.read_csv('filtered_output_string.csv', low_memory=False)
df.columns = [i for i in range(df.shape[1])]
df

df = df.rename(columns={126: 'Output'})
df

X = df.iloc[:, :-1]
print("Features shape =", X.shape)

Y = df.iloc[:, -1]
print("Labels shape =", Y.shape)

x_train, x_test, y_train, y_test = train_test_split(X, Y,
test_size=0.2, random_state=0)
random_forest = RandomForestClassifier()
random_forest.fit(x_train, y_train)
print(random_forest)
```

```
y_pred = random_forest.predict(x_test)
y_pred

cf_matrix = confusion_matrix(y_test, y_pred)
f1 = f1_score(y_test, y_pred, average='micro')
recall = recall_score(y_test, y_pred, average='micro')
precision = precision_score(y_test, y_pred, average='micro')
f1, recall, precision

labels = sorted(list(set(df['Output'])))
labels = [str(x).upper() for x in labels]

fig, ax = plt.subplots(figsize=(40, 15))

ax.set_title("Confusion Matrix - Random Forest - BISINDO")

maping = sns.heatmap(cf_matrix,
                     annot=True,
                     cmap = plt.cm.Blues,
                     linewidths=.2,
                     xticklabels=labels,
                     yticklabels=labels, vmax=150,
                     fmt='g',
                     ax=ax
                    )
maping

import pickle

# save model
with open('RandForest.pkl','wb') as f:
    pickle.dump(random_forest,f)
```

Lampiran C.5 Source Code Hand Tracking

```
import cv2
import mediapipe as mp
import numpy as np
import joblib
import tensorflow as tf
from collections import deque, Counter
from sklearn.preprocessing import LabelEncoder
import time
```

```
# ====== Load Model & Scaler ======
rf_model = joblib.load('RandForest.pkl')
lstm_model = tf.keras.models.load_model('best_model_lstm.h5')
le = joblib.load('label_encoder.pkl')
scaler = joblib.load('scaler.pkl')

# Mapping label ke bahasa Inggris
ind_to_eng = {
    "AIR": "WATER", "AKAN": "WILL", "AMBIL": "TAKE", "ANDA KAMU":
"YOU",
    "APA": "WHAT", "ATAS": "UP", "ATAU": "OR", "BAWAH": "DOWN",
"BELAJAR": "LEARN",
    "BERKATA": "SAY", "BISA": "CAN", "DALAM": "IN", "DAN": "AND",
"DARI": "FROM",
    "DIA": "HE/SHE", "HANYA": "ONLY", "INI": "THIS", "ITU": "THAT",
"JADI": "SO",
    "JIKA KALAU": "IF", "KERJA": "WORK", "KITA": "WE", "LUAR":
"OUTSIDE",
    "MAKAN": "EAT", "MEREKA": "THEY", "MILIK PUNYA": "OWN/HAVE",
"MINUM": "DRINK",
    "NAIK": "UP/GO UP", "NASI": "RICE", "PERGI": "GO", "PULANG": "GO
HOME",
    "ROTI": "BREAD", "SAYA": "I/ME", "SEMUA": "ALL", "SEPERTI": "LIKE",
"SIAPA": "WHO",
    "TAHU PAHAM": "KNOW/UNDERSTAND", "TAHUN": "YEAR", "TAPI": "BUT",
"TURUN": "DOWN/GO DOWN",
    "UNTUK": "FOR"
}

# ====== MediaPipe setup ======
mp_hands = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils
hands = mp_hands.Hands(static_image_mode=False, max_num_hands=2,
min_detection_confidence=0.7)

# ====== Buffer dan variabel ======
static_buffer = deque(maxlen=15)
sequence_buffer = deque(maxlen=30)
kalimat = ""
prev_gesture = ""

MIN_SEQUENCE_LENGTH = 30
```

```
MIN_FRAME_DELAY = 5
frame_counter = MIN_FRAME_DELAY

last_prediction_time = 0
prediction_delay = 5

lstm_count = 0
rf_count = 0

# ====== Inisialisasi Webcam ======
cap = cv2.VideoCapture(0)  # ganti index kalau kamu pakai lebih dari
satu kamera

if not cap.isOpened():
    print("❌ Tidak dapat mengakses webcam.")
    exit()

print("⏳ Menunggu 5 detik sebelum mulai... Siapkan tangan!")
time.sleep(5)
print("✅ Mulai deteksi gesture...")

while True:
    ret, frame = cap.read()
    if not ret:
        print("❌ Gagal menangkap frame dari webcam.")
        break

    frame = cv2.flip(frame, 1)
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    result = hands.process(rgb)

    all_landmarks = []
    model_used = None

    if result.multi_hand_landmarks:
        hand_landmarks_sorted = sorted(result.multi_hand_landmarks,
key=lambda hl: hl.landmark[0].x)
        for hand_landmarks in hand_landmarks_sorted:
            mp_drawing.draw_landmarks(frame, hand_landmarks,
mp_hands.HAND_CONNECTIONS)
            for lm in hand_landmarks.landmark:
                all_landmarks.extend([lm.x, lm.y, lm.z])
```

```
        if len(result.multi_hand_landmarks) == 1:
            all_landmarks.extend([0.0] * 63)

        if len(all_landmarks) == 126:
            sequence_buffer.append(all_landmarks)

            if len(sequence_buffer) == MIN_SEQUENCE_LENGTH and
frame_counter >= MIN_FRAME_DELAY:
                input_seq = np.array(list(sequence_buffer))
                input_seq_scaled =
scaler.transform(input_seq).reshape(1, MIN_SEQUENCE_LENGTH, 126)
                pred_proba = lstm_model.predict(input_seq_scaled,
verbose=0)[0]
                pred_label =
le.inverse_transform([np.argmax(pred_proba)])[0]
                gesture = pred_label
                model_used = "lstm"
                frame_counter = 0
            else:
                features_scaled = scaler.transform([all_landmarks])
                gesture = rf_model.predict(features_scaled)[0]
                model_used = "rf"

            frame_counter += 1
            static_buffer.append(gesture)

            if len(sequence_buffer) < MIN_SEQUENCE_LENGTH:
                gesture = Counter(static_buffer).most_common(1)[0][0]

            current_time = time.time()
            if gesture != prev_gesture and (current_time -
last_prediction_time) > prediction_delay:
                translated = ind_to_eng.get(gesture, gesture).upper()
                kalimat = translated
                prev_gesture = gesture
                last_prediction_time = current_time

                sequence_buffer.clear()
                static_buffer.clear()

                if model_used == "lstm":
                    lstm_count += 1
                elif model_used == "rf":
```

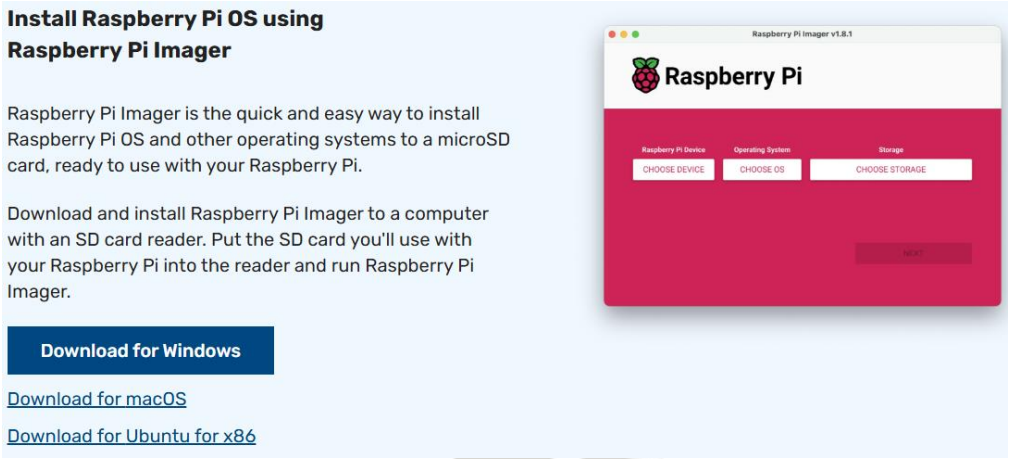
```
                    rf_count += 1

                print(f"→ Deteksi: {translated} via
{model_used.upper()}")

    cv2.putText(frame, kalimat, (10, 40), cv2.FONT_HERSHEY_SIMPLEX,
1.2, (0, 255, 0), 2)
    cv2.putText(frame, f"Random Forest: {rf_count}", (10, 80),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 255, 255), 2)
    cv2.putText(frame, f"LSTM: {lstm_count}", (10, 110),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 255, 255), 2)

    cv2.imshow("Gesture Recognition", frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# ====== Bersihkan ======
cap.release()
cv2.destroyAllWindows()
```

Lampiran D Tabel Resource yang Dipakai

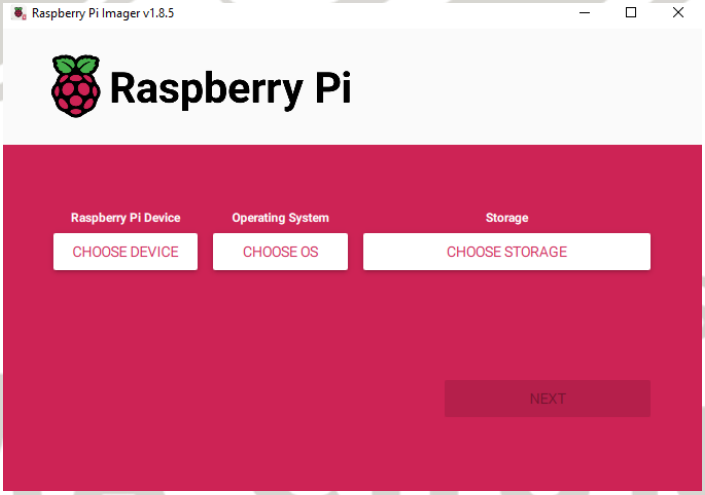
No	Nama Resource	Spesifikasi / Keterangan
1	Raspberry Pi 4 Model B	RAM 4GB, 4× Cortex-A72 1.5GHz CPU, USB 3.0, micro HDMI.
2	Sistem Operasi	Raspberry Pi OS 64-bit, versi 6.12
3	Media Penyimpanan	SanDisk MicroSDHC UHS-I, kapasitas 32GB
4	Kamera Eksternal	Logitech C270 USB Webcam, resolusi 720p

Lampiran E Langkah - Langkah Instalasi *Raspberry Pi OS*

1. Unduh dan instal aplikasi *Raspberry Pi Imager* di komputer Anda.



2. Masukkan *MicroSD* card ke *card reader* dan hubungkan ke komputer.
3. Buka *Raspberry Pi Imager* dan klik:
    - *Choose* OS → pilih *Raspberry Pi OS* (64-*bit*)
    - *Choose Storage* → pilih *MicroSD* yang terdeteksi



4. Klik *Write*, tunggu hingga proses selesai
5. Setelah selesai, *eject MicroSD* dan masukkan ke slot *Raspberry Pi*.
6. Sambungkan *Raspberry Pi* ke:
    - *Monitor* (via kabel *micro* HDMI)
    - *Keyboard & mouse* (via USB)
    - *Power supply* (5V 3A USB-C)

7. Nyalakan *Raspberry Pi.* Ikuti *wizard* instalasi awal (*setting* bahasa, Wi-Fi, update, dll.).
8. *Raspberry Pi OS* siap digunakan.