

**PENGENALAN BAHASA ISYARAT INDONESIA (BISINDO) DINAMIS  
MENGGUNAKAN MEDIPIPE DENGAN ALGORITMA  
TEMPORAL CONVOLUTIONAL NETWORK (TCN)**

**ANGGI YOHANES PARDEDE**

**191402143**



**PROGRAM STUDI S1 TEKNOLOGI INFORMASI  
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI  
UNIVERSITAS SUMATERA UTARA  
MEDAN  
2025**

**PENGENALAN BAHASA ISYARAT INDONESIA (BISINDO) DINAMIS  
MENGGUNAKAN MEDIPIPE DENGAN ALGORITMA  
TEMPORAL CONVOLUTIONAL NETWORK (TCN)**

**SKRIPSI**

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah  
Sarjana Teknologi Informasi

**ANGGI YOHANES PARDEDE**  
**191402143**



**PROGRAM STUDI S1 TEKNOLOGI INFORMASI**  
**FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI**  
**UNIVERSITAS SUMATERA UTARA**  
**MEDAN**  
**2025**

## PERNYATAAN

### PENGENALAN BAHASA ISYARAT INDONESIA (BISINDO) DINAMIS MENGGUNAKAN MEDIPIPE DENGAN ALGORITMA TEMPORAL CONVOLUTIONAL NETWORK (TCN)

#### SKRIPSI

Penulis mengakui bahwa skripsi ini merupakan hasil karya penulis sendiri, kecuali beberapa kutipan dan ringkasan yang masing – masing telah disebutkan sumbernya.

Medan, 10 Januari 2025

ANGGI YOHANES PARDEDE  
191402143

## PERSETUJUAN

Judul : Pengenalan Bahasa Isyarat Indonesia (BISINDO) Dinamis Menggunakan MediaPipe dengan Algoritma Temporal Convolutional Network (TCN)

Kategori : Skripsi

Nama Mahasiswa : Anggi Yohanes Pardede

Nomor Induk Mahasiswa : 191402143

Program Studi : Sarjana (S-1) Teknologi Informasi

Fakultas : Ilmu Komputer dan Teknologi Informasi

Universitas Sumatera Utara

Medan, 10 Januari 2025

Komisi Pembimbing:

Pembimbing 1,

Pembimbing 2,

Ainul Hizriadi, S.Kom., M.Sc  
NIP. 198510272017061001

Mohammad Fadly Syahputra, B.Sc., M.Sc.IT  
NIP. 198301292009121003

Diketahui/disetujui oleh

Program Studi S-1 Teknologi Informasi

Ketua,



Dedy Ansandi S.T., M.Kom.  
NIP. 197908312009121002

## UCAPAN TERIMA KASIH

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa atas segala rahmat, karunia, dan petunjuk-Nya yang melimpah penulis dapat menyelesaikan penyusunan skripsi ini sebagai syarat untuk memperoleh gelar Sarjana Komputer (S.Kom) dari Program Studi S1 Teknologi Informasi Universitas Sumatera Utara.

Dengan segala kerendahan hati, penulis ingin mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan, bimbingan, dan motivasi selama proses penulisan skripsi ini. Oleh karena itu pada kesempatan ini, penulis ingin menyampaikan ucapan terima kasih yang tulus kepada :

1. Keluarga penulis, Orangtua Ayah Henry Pardede, dan Ibu Krislita Sibuea yang selalu mendukung dan mendoakan penulis, serta saudara dan saudari penulis.
2. Ibu Dr. Maya Silvi Lidya, B.Sc., M.Sc. selaku Dekan Fasilkom-TI USU.
3. Bapak Dedy Arisandi, S.T., M.Kom. selaku Ketua Program Studi Teknologi Informasi Universitas Sumatera Utara.
4. Bapak Ivan Jaya S.Si., M.Kom. selaku Sekretaris Program Studi Teknologi Informasi Universitas Sumatera Utara.
5. Bapak Mohammad Fadly Syah Putra M.Sc. selaku Dosen Pembimbing I penulis dan Bapak Ainul Hizriadi S.Kom., M.Sc. yang telah meluangkan waktunya, memberikan bimbingan, dorongan, serta wawasan yang berharga dalam proses penulisan skripsi ini.
6. Seluruh Dosen Pengajar Studi S1 Program Teknologi Informasi Universitas Sumatera Utara.
7. Seluruh Staff, dan Pegawai Studi S1 Program Teknologi Informasi Universitas Sumatera Utara.
8. Teman-teman penulis yang telah membantu penulis selama penyelesaian perkuliahan terutama Sarah, Daniel, Christopher, Geylfedra, Timothy, AR, Zikri, dan warga Andimed.

9. Senior dan Junior yang telah membantu memberikan saran dan masukan dalam pengerjaan tugas akhir.

Semoga kasih dan berkat Tuhan senantiasa menyertai semua pihak yang telah mendukung dan memberikan bantuan serta perhatian kepada penulis selama masa perkuliahan dan proses penyusunan skripsi ini.

Medan, 20 Agustus 2024

Penulis,



Anggi Yohanes Pardede

191402143



## ABSTRAK

Komunikasi merupakan kebutuhan dasar manusia, namun bagi teman tuli-bisu, hal ini menjadi tantangan karena keterbatasan dalam berbicara dan mendengar. Bahasa Isyarat Indonesia (BISINDO) adalah alat komunikasi utama oleh komunitas tuli-bisu, namun belum dipahami secara luas oleh masyarakat umum. Penelitian ini bertujuan untuk mengembangkan sistem pengenalan BISINDO dinamis menggunakan *MediaPipe* dan algoritma *Temporal Convolutional Network* (TCN), aplikasi ini dirancang untuk membantu menerjemahkan gerakan isyarat BISINDO ke dalam bahasa Indonesia dalam bentuk teks dan suara. Dengan menggunakan metode studi literatur, analisis permasalahan, perancangan sistem, implementasi, dan pengujian, penelitian ini menunjukkan bahwa sistem yang dikembangkan mampu mengenali gerakan isyarat dengan akurasi 90,25%, dengan tingkat presisi, *recall*, dan *f1-score* yang tinggi untuk setiap *label* data. Sistem ini efektif dalam mengurangi kompleksitas *pre-processing* data serta mampu mengenali pola gerakan tangan secara akurat. Kesimpulannya, penelitian ini berhasil merancang dan mengimplementasikan sistem pengenalan BISINDO yang dapat diandalkan, dengan saran untuk pengembangan lebih lanjut termasuk penggunaan *dataset* yang lebih besar, integrasi dengan teknologi *real-time*, dan pengembangan aplikasi *mobile* untuk memperluas aksesibilitas.

Kata Kunci: Bahasa Isyarat Indonesia (BISINDO), *MediaPipe*, *Temporal Convolutional Network* (TCN), Pengenalan Isyarat

# **DYNAMIC INDONESIAN SIGN LANGUAGE (BISINDO) RECOGNITION USING MEDIPIPE WITH TEMPORAL CONVOLUTIONAL NETWORK (TCN) ALGORITHM**

## **ABSTRACT**

Communication is a fundamental human need, but for the deaf-mute community, it poses challenges due to limitations in speaking and hearing. Indonesian Sign Language (BISINDO) is the primary communication tool for the deaf-mute community, yet it is not widely understood by the general public. This study aims to develop a dynamic BISINDO recognition system using MediaPipe and the Temporal Convolutional Network (TCN) algorithm, which is expected to assist in translating BISINDO sign language into Indonesian in the form of text and voice. By employing literature review, problem analysis, system design, implementation, and testing methods, this study demonstrates that the developed system is capable of recognizing sign movements with an accuracy of 90.25%, with high precision, recall, and f1-score for each data label. The system is effective in reducing data pre-processing complexity and accurately recognizing movement patterns. In conclusion, this study successfully designed and implemented a reliable BISINDO recognition system, with suggestions for further development, including the use of larger datasets, integration with real-time technology, and the development of mobile applications to enhance accessibility.

Keywords: Indonesian Sign Language (BISINDO), Sign Recognition, MediaPipe,  
Temporal Convolutional Network (TCN)

## DAFTAR ISI

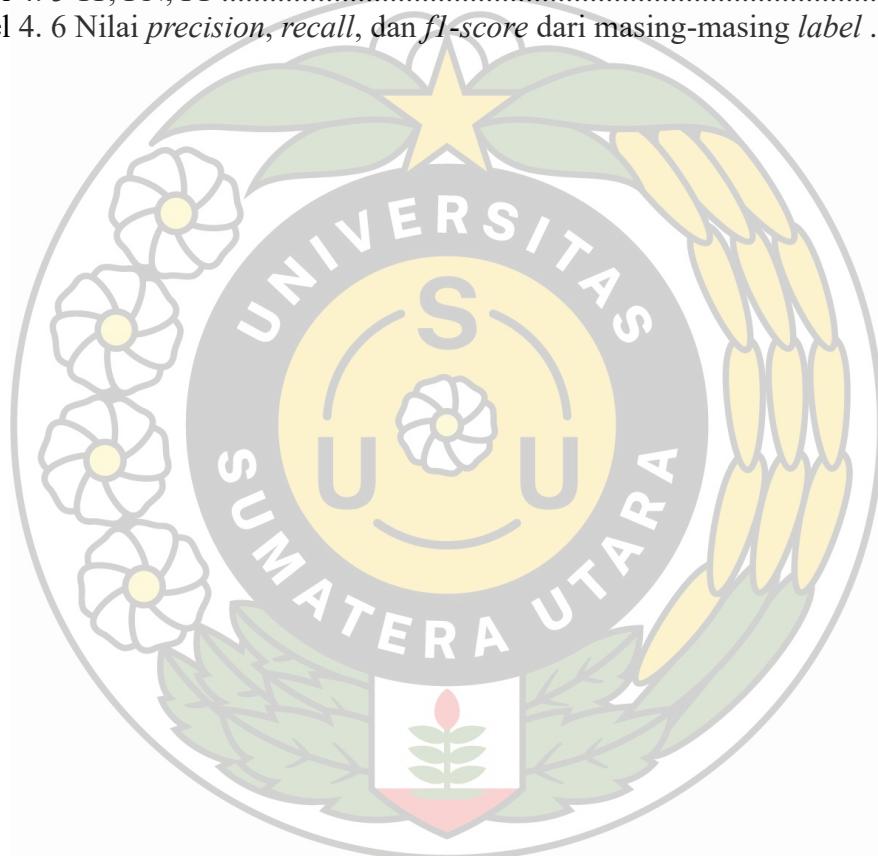
<b>PERNYATAAN.....</b>	<b>ii</b>
<b>UCAPAN TERIMA KASIH .....</b>	<b>iii</b>
<b>ABSTRAK .....</b>	<b>vi</b>
<b>ABSTRACT .....</b>	<b>vii</b>
<b>DAFTAR ISI.....</b>	<b>viii</b>
<b>DAFTAR TABEL .....</b>	<b>xi</b>
<b>DAFTAR GAMBAR.....</b>	<b>xii</b>
<b>BAB 1 PENDAHULUAN .....</b>	<b>1</b>
1.1 Latar Belakang .....	1
1.2 Rumusan Masalah.....	4
1.3 Tujuan Penelitian .....	4
1.4 Batasan Masalah .....	4
1.5 Manfaat Penelitian .....	5
1.6 Metodologi Penelitian.....	5
1.6.1 Studi Literatur.....	5
1.6.2 Analisis Permasalahan.....	5
1.6.3 Perancangan Sistem .....	6
1.6.4 Implementasi.....	6
1.6.5 Pengujian Sistem.....	6
1.6.6 Penyusunan Laporan .....	6
1.7 Sistematika Penulisan .....	6
<b>BAB 2 LANDASAN TEORI .....</b>	<b>8</b>
2.1 Pengolahan Citra Digital.....	8
2.2 Citra Digital .....	8
2.2.2 Citra Grayscale.....	9
2.2.3 Citra Warna .....	9
2.3 Deteksi Objek.....	10
2.4 Bahasa Isyarat .....	10
2.5 Bahasa Isyarat Indonesia (BISINDO).....	10
2.6 MediaPipe .....	11
2.6.1 MediaPipe Holistic.....	12
2.6.2 MediaPipe Face Mesh.....	12

2.6.3 MediaPipe Pose.....	13
2.6.4 MediaPipe .....	13
2.7 Temporal Convolutional Network (TCN).....	14
2.8 Confusion Matrix .....	17
2.9 Penelitian Terdahulu.....	19
<b>BAB 3 ANALISIS DAN PERANCANGAN SISTEM.....</b>	<b>27</b>
3.1 Dataset.....	27
3.2 Arsitektur Umum .....	30
3.2.1     Video Acquisition.....	31
1. Pembuatan Dataset.....	31
2. Edit Data .....	32
3.2.2     Video Pre-processing .....	32
1. Extract Frame .....	33
2. Keyframe Selection.....	33
3. Frame Reduction.....	37
3.2.3     Feature Extraction.....	39
1. Landmark Setup .....	39
2. Extract Keypoint .....	40
3. Data Augmentation .....	41
4. Bounding Box Normalization.....	45
5. Save Keypoint.....	49
3.2.4     Data Pre-processing .....	50
1. Data Padding.....	50
2. Data Labeling.....	51
3. Split Data .....	52
4. Create Features dan Labels Array .....	53
3.2.5     Training.....	55
3.2.6     Testing.....	57
1. Input .....	57
2. Load Model.....	58
3. Pre-processing.....	58
4. Prediction .....	58
5. Output .....	58

3.3 Perancangan Antarmuka Sistem .....	59
3.3.1 Tampilan Utama .....	59
3.3.2 Tampilan Loading Penerjemahan.....	59
3.3.3 Tampilan Hasil Terjemahan .....	60
<b>BAB 4 IMPLEMENTASI DAN PENGUJIAN SISTEM.....</b>	<b>61</b>
4.1 Implementasi Sistem.....	61
4.1.1 Spesifikasi Hardware dan Software .....	61
4.1.2 Spesifikasi Perangkat Lunak.....	61
4.1.3 Implementasi Perancangan Tampilan Antarmuka.....	62
1. Tampilan Utama.....	62
2. Tampilan Loading Penerjemahan.....	62
3. Tampilan Hasil Terjemahan .....	63
4.1.4 Implementasi Data .....	63
4.1.5 Pelatihan Sistem.....	64
4.2 Pengujian Sistem.....	67
4.3 Evaluasi Sistem.....	71
<b>BAB 5 KESIMPULAN DAN SARAN.....</b>	<b>77</b>
5.1 Kesimpulan .....	77
5.2 Saran .....	77
<b>DAFTAR PUSTAKA .....</b>	<b>79</b>

**DAFTAR TABEL**

Tabel 3. 1 Data Kata Isyarat.....	28
Tabel 3. 2 Pembagian Data .....	29
Tabel 3. 3 <i>Hyperparameter Tuning</i> pada TCN.....	57
Tabel 3. 4 Spesifikasi <i>Hardware</i> .....	61
Tabel 3. 5 Spesifikasi <i>Software</i> .....	61
Tabel 4. 1 <i>Hyperparameter Tuning</i> .....	64
Tabel 4. 2 Hasil Percobaan <i>Hyperparameter Tuning</i> .....	64
Tabel 4. 3 Hasil Pengujian Sistem (Indikator Kata).....	67
Tabel 4. 4 Hasil Pengujian Sistem (Indikator Jarak).....	71
Tabel 4. 5 TP, FN, FP .....	72
Tabel 4. 6 Nilai <i>precision</i> , <i>recall</i> , dan <i>f1-score</i> dari masing-masing <i>label</i> .....	74



## DAFTAR GAMBAR

Gambar 2. 1 Contoh Representasi Citra Biner.....	9
Gambar 2. 2 Contoh Representasi Citra <i>Grayscale</i> .....	9
Gambar 2. 3 Contoh Representasi Citra Warna .....	10
Gambar 2. 4 Ilustrasi Bahasa Isyarat Indonesia.....	11
Gambar 2. 5 <i>Mediapipe Holistic</i> .....	12
Gambar 2. 6 <i>Mediapipe Face Landmark Detection</i> .....	13
Gambar 2. 7 <i>Mediapipe Pose Landmark Detection</i> .....	13
Gambar 2. 8 <i>Mediapipe Landmarks Detection</i> .....	14
Gambar 2. 9 Alur Kerja <i>MediaPipe</i> .....	14
Gambar 2. 10 Arsitektur Umum TCN.....	15
Gambar 2. 11 TCN <i>Residual Block</i> .....	16
Gambar 2. 12 <i>Confusion Matrix</i> .....	18
Gambar 3. 1 Contoh data Bahasa Isyarat.....	27
Gambar 3. 2 Arsitektur Umum.....	31
Gambar 3. 3 Contoh data yang benar dan salah.....	32
Gambar 3. 4 Hasil proses <i>Extract Frame</i> .....	33
Gambar 3. 5 Hasil proses <i>Keyframe Selection</i> .....	37
Gambar 3. 6 Hasil proses <i>Reduction Frame</i> .....	39
Gambar 3. 7 <i>Mediapipe Holistic</i> .....	39
Gambar 3. 8 <i>Landmark Mediapipe Holistic</i> (A), <i>Landmark</i> yang Dikonfigurasi (B) .	40
Gambar 3. 9 Bauentwurfslehre oleh Ernst Neufert (1936) .....	46
Gambar 3. 10 Ilustrasi ruang isyarat .....	47
Gambar 3. 11 Contoh <i>Bounding Box</i> pada salah satu <i>frame</i> .....	49
Gambar 3. 12 Contoh <i>array</i> setelah ekstrak <i>keypoint</i> .....	50
Gambar 3. 13 Contoh <i>array</i> setelah <i>padding</i> data .....	51
Gambar 3. 14 Kode dan hasil dari proses <i>labelling</i> data .....	52
Gambar 3. 15 Hasil implementasi proses <i>split data</i> .....	53
Gambar 3. 16 Contoh <i>array features</i> .....	54
Gambar 3. 17 Contoh <i>array labels</i> .....	55
Gambar 3. 18 Contoh <i>array labels</i> setelah proses <i>one-hot encoding</i> .....	55
Gambar 3. 19 Tampilan Utama .....	59
Gambar 3. 20 Tampilan <i>Loading</i> Penerjemahan.....	60
Gambar 3. 21 Tampilan Hasil Penerjemahan.....	60
Gambar 4. 1 Contoh <i>dataset</i> video isyarat BISINDO.....	63
Gambar 4. 2 Contoh <i>dataset frame</i> isyarat BISINDO .....	64

## BAB 1

### PENDAHULUAN

#### 1.1 Latar Belakang

Komunikasi adalah proses pertukaran informasi dan ide antara individu atau kelompok. Komunikasi sangat penting bagi manusia karena memungkinkan kita untuk berinteraksi dengan orang lain, membangun hubungan, dan memperoleh informasi yang diperlukan (Situmorang & Gunawan, 2020). Namun, bagi teman tuli dan bisu, komunikasi bisa menjadi tantangan karena mereka kehilangan kemampuan untuk berbicara dan mendengar. Oleh karena itu, mereka menggunakan bahasa isyarat sebagai alat komunikasi.

Bahasa isyarat adalah bahasa yang menggunakan gerakan tubuh, terutama tangan dan bibir, untuk menyampaikan arti (Pratamasunu et al., 2022). Bahasa isyarat digunakan oleh individu dengan gangguan pendengaran dan bicara sebagai bentuk komunikasi non-verbal untuk mengekspresikan pikiran dan emosi (Bantupalli & Xie, 2018). Menurut World Federation of the Deaf, ada sekitar 70 juta orang tuli dan ada lebih dari 200 bahasa isyarat berbeda di seluruh dunia, sedangkan menurut Pusat Bahasa Isyarat Indonesia terdapat lebih dari 2.5 juta tuli di Indonesia dan terdapat 2 bahasa isyarat, yaitu Sistem Isyarat Bahasa Indonesia (SIBI) dan Bahasa Isyarat Indonesia (BISINDO). BISINDO menjadi bahasa isyarat yang dominan dalam interaksi sehari-hari bagi penyandang disabilitas tuli karena kemudahannya dalam pergaulan (Saraswati et al., 2022). Namun masih banyak masyarakat umum yang belum memahami dan bisa menggunakan bahasa isyarat (Pratamasunu et al., 2022). Hal ini menjadi kendala bagi teman tuli-bisu dalam hal berkomunikasi dengan masyarakat umum. Masyarakat umum memang bisa mempelajari bahasa isyarat dari berbagai sumber seperti buku bacaan, namun pendekatan ini memiliki beberapa keterbatasan. Bahasa isyarat bersifat visual dan dinamis, sehingga sulit dipahami hanya dari gambar statis atau teks tertulis. Selain itu, tidak adanya umpan balik langsung dan interaksi yang terbatas menyulitkan pembelajaran untuk memahami gerakan dengan benar, serta proses menghafal kosakata isyarat memakan waktu yang cukup lama. Oleh karena itu, penggunaan teknologi dalam

penerjemahan bahasa isyarat menjadi solusi yang lebih efektif dan praktis. Teknologi memungkinkan penerjemahan bahasa isyarat secara *real-time*, memberikan interaksi yang lebih intuitif, dan sangat bermanfaat dalam situasi mendesak. Hal tersebut menjadi salah satu ide utama dalam pembuatan penelitian ini.

Pemanfaatan teknologi dapat menjadi solusi dari permasalahan di atas dengan memadukan *computer vision* dan *machine learning*. *Computer vision* adalah cabang ilmu komputer yang mempelajari cara komputer dapat memahami dan menganalisis gambar dan video. *Computer vision* menggunakan teknik *machine learning*, seperti *deep learning*, untuk memproses gambar dan video dan mengidentifikasi objek, wajah, dan aktivitas dalam gambar dan video (Szegedy et al., 2015). *Machine learning* dapat didefinisikan sebagai aplikasi komputer dan algoritma matematika yang diadopsi dengan cara pembelajaran yang berasal dari data dan menghasilkan prediksi di masa yang akan datang (Goldberg & Holland, 1988). *Computer vision* dan *machine learning* berperan penting dalam pembelajaran bahasa isyarat dengan memungkinkan mesin mengenali dan menerjemahkan gerakan secara otomatis. *Computer vision* digunakan untuk melacak dan memahami gerakan tangan serta ekspresi wajah melalui kamera, sementara *machine learning* memungkinkan sistem belajar dari data gerakan isyarat sehingga dapat mengenali pola dan variasi. Dengan integrasi kedua teknologi ini, penerjemahan bahasa isyarat dapat dilakukan secara *real-time*, memfasilitasi komunikasi yang lebih cepat dan efektif. Penelitian ini menggabungkan kedua teknologi di atas untuk membuat *model* komunikasi satu arah antara teman tuli-bisu dengan masyarakat umum yang mengenali gerakan pada Bahasa Isyarat Indonesia dan menerjemahkan menjadi bahasa Indonesia.

Penelitian ini juga melibatkan penelitian-penelitian terdahulu sebagai acuan pembanding penelitian, memperkuat alur penelitian, dan untuk dapat memunculkan inspirasi terbaru dari penelitian sebelumnya. Penelitian terdahulu yang dilakukan oleh (Halder & Tayade, 2021) dalam penelitiannya yang berjudul “*Real-time Vernacular Sign Language Recognition using MediaPipe and Machine Learning*” menyatakan bahwa penggunaan perangkat sensor tambahan dalam pengenalan bahasa isyarat memiliki kekurangan yaitu ketidaknyamanan dan keterbatasan dalam penggunaannya bagi penutur bahasa isyarat dikarenakan perangkat sensor ini tergolong mahal. Penelitian ini merancang sistem yang menggunakan *framework* *MediaPipe* dan algoritma *machine learning* yaitu algoritma SVM yang dimana sistem ini dapat

mempermudah dalam pendekripsi gerakan tubuh dan juga pengimplementasiannya yang sederhana, cepat, portabel dan hemat biaya. Pengujian dari sistem ini mendapatkan hasil akurasi dengan rata-rata sebesar 99% untuk sebagian besar kumpulan data.

Selanjutnya, penelitian terdahulu oleh Putri (2022) dengan judul penelitian “Pendekripsi Bahasa Isyarat Indonesia Secara *Real-time* Menggunakan *Long Short-Term Memory (LSTM)*” menyatakan bahwa penelitian ini bertujuan untuk merancang sistem yang dapat menjadi penghubung komunikasi antar teman tuli-bisu dan teman dengar. Sistem ini dapat melakukan pendekripsi kosakata isyarat BISINDO (Bahasa Isyarat Indonesia) yang akan dikonversi menjadi sebuah teks. Penelitian ini menggunakan *MediaPipe Holistic* sebagai pendekripsi gestur tubuh dan metode yang digunakan ialah metode LSTM. Pengujian dari sistem ini mendapatkan hasil akurasi sebesar 92%.

Kemudian, penelitian yang dilakukan oleh (Tsinganos et al., 2022) yang berjudul “*Real-Time Analysis of Gesture Recognition with Temporal Convolutional Networks*”. Penelitian ini membahas penerapan metode *deep learning*, khususnya Jaringan Konvolusional Temporal (TCNs), pada masalah klasifikasi gestur berbasis elektromiografi (sEMG). Dalam beberapa tahun terakhir, penggunaan TCNs dalam berbagai domain telah memberikan dampak yang signifikan. Penelitian ini mengambil pendekatan untuk mengklasifikasikan gestur tangan berbasis sEMG sebagai masalah klasifikasi urutan, dengan melakukan eksperimen simulasi pada *dataset* sEMG yang direkam untuk mengevaluasi perilaku *real-time* dari *model* TCN yang diusulkan. Hasil penelitian menunjukkan bahwa *model* yang diusulkan mampu meningkatkan sedikit akurasi klasifikasi dibandingkan dengan *model* yang sudah ada, namun akurasi klasifikasi menurun saat evaluasi *real-time*, menunjukkan bahwa arsitektur TCN tidak cocok untuk aplikasi *real-time*. Kesimpulan penelitian ini menyarankan bahwa adaptasi prosedur pemberian data selama pelatihan untuk aplikasi *real-time* dapat lebih efektif daripada skema analisis offline.

Pengenalan Bahasa Isyarat Indonesia (BISINDO) dinamis dengan menggunakan *MediaPipe* dan pengimplementasian algoritma *Temporal Convolutional Network* (TCN) adalah topik penelitian yang menarik dalam pengembangan teknologi untuk meningkatkan aksesibilitas bagi komunitas tuli-bisu. *MediaPipe* menyediakan solusi *real-time* yang efisien untuk melacak gerakan tubuh, tangan, dan wajah, sehingga

mempermudah deteksi gestur. Sementara itu, TCN unggul dalam menangani data sekuensial seperti gerakan tangan dalam bahasa isyarat, memungkinkan sistem mengenali pola gerakan dengan lebih akurat. Kombinasi ini dapat membantu memperluas akses komunikasi bagi individu dengan disabilitas tuli-bisu.

Berdasarkan uraian latar belakang di atas, dilakukan penelitian dengan judul “PENGENALAN BAHASA ISYARAT INDONESIA (BISINDO) DINAMIS MENGGUNAKAN *MEDIPIPE* DENGAN ALGORITMA *TEMPORAL CONVOLUTIONAL NETWORK* (TCN)”

## 1.2 Rumusan Masalah

Komunitas tuli-bisu di Indonesia menghadapi tantangan komunikasi karena keterbatasan berbicara dan mendengar, sementara Bahasa Isyarat Indonesia (BISINDO), sebagai alat komunikasi utama mereka, belum dipahami luas oleh masyarakat umum. Metode pembelajaran bahasa isyarat yang ada, seperti buku dan materi visual statis, kurang efektif dalam menangkap gerakan dinamis yang menjadi kunci dalam komunikasi isyarat. Untuk mengatasi masalah ini, dibutuhkan sebuah teknologi yang dapat digunakan untuk mengenali dan menerjemahkan gerakan isyarat BISINDO ke dalam teks atau suara sehingga dapat menjembatani komunikasi antara komunitas tuli-bisu dan masyarakat umum, meningkatkan aksesibilitas dan partisipasi mereka dalam kehidupan sosial

## 1.3 Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk membuat aplikasi penerjemah bahasa isyarat BISINDO secara dinamis menggunakan *MediaPipe* dan algoritma *Temporal Convolutional Network* (TCN). Aplikasi ini diharapkan mampu mengenali gerakan isyarat dengan akurasi tinggi dan menerjemahkannya ke dalam bahasa Indonesia dalam bentuk teks dan suara. Penelitian ini juga bertujuan untuk meningkatkan aksesibilitas komunikasi bagi komunitas tuli-bisu serta memperkenalkan teknologi yang dapat digunakan untuk mempermudah interaksi antara komunitas tuli-bisu dan masyarakat umum.

## 1.4 Batasan Masalah

Adapun batasan masalah untuk mencegah meluasnya ruang lingkup permasalahan dalam penelitian ini adalah:

1. Bahasa Isyarat yang Digunakan: Penelitian ini hanya akan fokus pada Bahasa Isyarat Indonesia (BISINDO) sebagai bahasa isyarat yang digunakan dalam sistem.
2. Data *training* dan *testing* yang digunakan untuk penelitian ini adalah video isyarat BISINDO berekstensi .mp4 yang dikonversi menjadi *frame* dengan ekstensi .jpg.
3. Penelitian ini mengidentifikasi beberapa jenis kata seperti kata ganti orang, kata tanya, kata sapaan, kata sifat, dan kata kerja berjumlah 40 kata.
4. *Output* yang dihasilkan adalah teks dan suara terjemahan dari bahasa isyarat.
5. Pengembangan aplikasi dilakukan untuk perangkat dengan sistem operasi *Windows*.

## 1.5 Manfaat Penelitian

Adapun manfaat dari penelitian ini adalah:

1. Aplikasi ini meningkatkan aksesibilitas komunikasi bagi komunitas tuli-bisu dalam berkomunikasi menggunakan bahasa isyarat berstandar Bahasa Isyarat Indonesia (BISINDO).
2. Menjadi sumber referensi dan bahan pembelajaran bagi penelitian lain yang menerapkan metode *Temporal Convolutional Network* (TCN).

## 1.6 Metodologi Penelitian

Berikut adalah beberapa tahapan yang akan dilakukan dalam penelitian ini :

### 1.6.1 Studi Literatur

Studi literatur adalah proses sistematis untuk mengumpulkan, meninjau, dan menganalisis referensi yang relevan mengenai Image Processing yaitu pengenalan bahasa isyarat dengan menggunakan *framework MediaPipe* serta pengimplementasian metode pengolahan citra seperti *Temporal Convolutional Network* (TCN) yang diperoleh dari artikel, jurnal, skripsi, internet, atau sumber informasi lainnya.

### 1.6.2 Analisis Permasalahan

Tahap analisis permasalahan merupakan tahap yang dilakukan untuk memahami konsep penerapan *framework MediaPipe* serta metode *Temporal Convolutional Network* (TCN) yang akan digunakan untuk mengidentifikasi Bahasa Isyarat Indonesia (BISINDO).

#### 1.6.3 Perancangan Sistem

Dalam tahap ini, penulis melakukan perancangan arsitektur, penentuan *dataset* untuk *training* dan *testing*, dan perancangan antarmuka berdasarkan hasil analisis permasalahan dari studi literatur.

#### 1.6.4 Implementasi

Tahap implementasi merupakan penerapan dari perancangan arsitektur umum berdasarkan analisis permasalahan yang akan diteliti yang akan menghasilkan sebuah sistem pengenalan Bahasa Isyarat Indonesia (BISINDO).

#### 1.6.5 Pengujian Sistem

Pada tahap ini dilakukan pengujian sistem yang telah dibangun untuk mengetahui tingkat akurasi yang diperoleh dari metode *Temporal Convolutional Network* (TCN) dalam mengenali bahasa isyarat dari teman tuli-bisu.

#### 1.6.6 Penyusunan Laporan

Pada tahap ini, dilakukan penyusunan laporan berdasarkan keseluruhan hasil dan temuan dari penelitian yang telah dilakukan sebagai Dokumen yang mendetail dan komprehensif mengenai metodologi, analisis data, serta interpretasi hasil penelitian. Laporan ini bertujuan untuk memberikan gambaran yang jelas dan sistematis mengenai proses penelitian, temuan yang ditemukan, dan implikasi dari hasil penelitian tersebut.

### 1.7 Sistematika Penulisan

Sistematika penulisan dalam penelitian ini mencakup beberapa bagian utama, yang terdiri dari :

#### BAB 1 : Pendahuluan

Bagian pendahuluan menguraikan latar belakang penelitian, rumusan masalah, tujuan penelitian, batasan masalah, manfaat penelitian, metodologi penelitian dan sistematika penulisan dari penelitian.

#### BAB 2 : Landasan Teori

Landasan teori memberikan gambaran umum mengenai konsep, teori, atau kerangka pemikiran yang dipakai untuk memahami dan menganalisis masalah yang sedang diteliti. Dalam bab ini, akan dijelaskan berbagai teori terkait masalah penelitian, termasuk metode yang akan digunakan, teknik penyelesaian masalah, pengumpulan

data, serta fitur-fitur yang akan diaplikasikan dalam penelitian. Selain itu, bagian ini juga akan mencantumkan penelitian-penelitian sebelumnya yang dilakukan oleh peneliti lain sebagai referensi untuk penelitian yang akan dilakukan.

### **BAB 3 : Analisis dan Perancangan Sistem**

Bagian ini menjelaskan proses dan alur arsitektur umum, yaitu tahap pengumpulan *dataset*, *pre-processing*, ekstraksi fitur menggunakan *MediaPipe*, dan penerapan algoritma *Temporal Convolutional Network* (TCN) pada pengenalan Bahasa Isyarat Indonesia (BISINDO).

### **BAB 4 : Implementasi dan Pengujian Sistem**

Bab ini berisi implementasi dari sistem yang dibahas pada bab sebelumnya, dan juga menampilkan dan menjelaskan hasil dari pengujian sistem yang telah dirancang.

### **BAB 5 : Kesimpulan dan Saran**

Bab ini berisi hasil pemikiran penulis terhadap penelitian yang telah dilakukan dan saran yang dapat diterapkan pada pengembangan penelitian selanjutnya.

## BAB 2

### LANDASAN TEORI

#### 2.1 Pengolahan Citra Digital

Pengolahan citra digital adalah suatu teknik yang digunakan untuk mengolah atau memanipulasi gambar dalam format 2 dimensi. Pengolahan citra digital dapat juga dikatakan sebagai segala operasi untuk memperbaiki, menganalisa, atau mengubah suatu gambar (Gonzalez & Woods, 2002). Dalam penelitian ini, pengolahan citra digital berperan penting dalam mengenali gerakan isyarat BISINDO dari gambar yang ditangkap oleh kamera. Sistem penerjemahan bahasa isyarat memerlukan langkah-langkah pengolahan seperti segmentasi dan ekstraksi fitur dari citra untuk mendeteksi gerakan tangan dan wajah secara akurat. Hal ini membantu meningkatkan kualitas data visual yang akan diproses lebih lanjut oleh *MediaPipe* sebelum diklasifikasikan menggunakan algoritma *Temporal Convolutional Network* (TCN).

#### 2.2 Citra Digital

Citra digital adalah representasi visual yang terdiri dari fungsi dua dimensi  $f(x,y)$ , dengan ukuran M baris dan N kolom. Koordinat x dan y menunjukkan posisi spasial pada citra, sementara nilai amplitudo di setiap titik koordinat  $(x,y)$  menggambarkan intensitas (Putra et al., 2013). Pada penelitian ini, citra digital digunakan sebagai data utama yang diolah oleh sistem untuk mengenali gerakan isyarat. Kamera menangkap gerakan isyarat BISINDO dan menghasilkan citra digital yang dianalisis melalui tahap-tahap seperti konversi menjadi *grayscale* atau pemrosesan lebih lanjut dengan *MediaPipe*. Citra yang diolah ini kemudian digunakan untuk mendeteksi dan mengenali pola gerakan yang akan diterjemahkan oleh algoritma TCN menjadi teks atau suara.

Citra digital terdiri dari 3 jenis, yaitu sebagai berikut :

##### 2.2.1 Citra Biner

Citra biner adalah jenis citra digital yang hanya memiliki dua nilai intensitas yang mungkin untuk setiap piksel, yaitu 0 dan 1. Nilai 0 biasanya merepresentasikan warna

hitam, dan nilai 1 merepresentasikan warna putih (Bhahri & Rachmat, 2018). Contoh representasi citra biner ialah seperti pada Gambar 2.1.



**Gambar 2. 1** Contoh Representasi Citra Biner

(Sumber: <https://koleksibukukuliah.blogspot.com/2017/07/5-jenis-citra-digital-yang-harus-kamu.html>)

### 2.2.2 *Citra Grayscale*

Citra *grayscale* merupakan jenis citra digital di mana setiap piksel hanya memiliki satu nilai intensitas yang merepresentasikan tingkat keabuan. Nilai intensitas ini umumnya berada dalam rentang tertentu, misalnya dari 0 hingga 255 pada citra 8-bit, di mana 0 menunjukkan warna hitam, 255 menunjukkan warna putih, dan nilai-nilai di antara keduanya menunjukkan berbagai tingkat abu-abu (Safrizal & Harjoko, 2013). Contoh representasi dari citra *grayscale* seperti pada Gambar 2.2.



**Gambar 2. 2** Contoh Representasi Citra *Grayscale*

(Sumber: <https://koleksibukukuliah.blogspot.com/2017/07/5-jenis-citra-digital-yang-harus-kamu.html>)

### 2.2.3 *Citra Warna*

Citra warna adalah jenis gambar digital di mana setiap piksel menyimpan informasi tentang warna yang dihasilkan dari kombinasi beberapa kanal warna. Kanal yang paling sering digunakan adalah merah, hijau, dan biru (RGB), di mana setiap piksel memiliki tiga nilai intensitas yang menunjukkan proporsi relatif dari ketiga warna dasar tersebut

(Pramudiya et al., 2024). Contoh representasi dari citra warna ialah seperti pada Gambar 2.3.



**Gambar 2.3** Contoh Representasi Citra Warna

(Sumber: <https://koleksibukukuliah.blogspot.com/2017/07/5-jenis-citra-digital-yang-harus-kamu.html>)

### 2.3 Deteksi Objek

Deteksi objek adalah teknik dalam visi komputer yang digunakan untuk menganalisis gambar dan video dengan tujuan mengidentifikasi keberadaan dan lokasi objek dalam bentuk gambar atau rekaman video. Tujuan deteksi objek adalah mengidentifikasi segala bentuk objek dari satu atau beberapa kategori tertentu, tanpa memandang ukuran, posisi, pose, sudut pandang, atau kondisi pencahayaan yang mungkin mempengaruhi objek tersebut (Verschae & Ruiz-del-Solar, 2015).

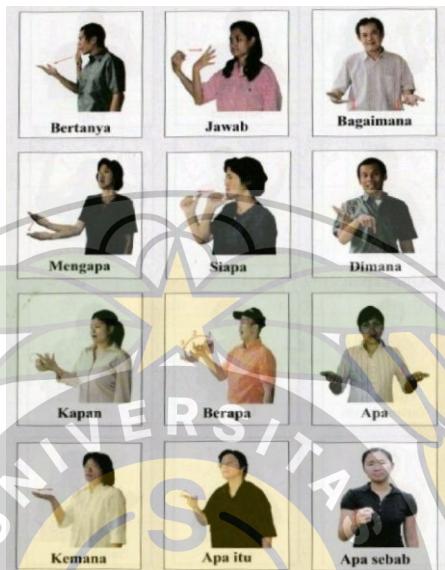
### 2.4 Bahasa Isyarat

Bahasa isyarat adalah bahasa non-lisan yang mengandalkan komunikasi manual, bahasa tubuh, dan gerak bibir. Bahasa isyarat digunakan oleh komunitas tuli untuk berkomunikasi. Bahasa isyarat memungkinkan orang yang tidak dapat mendengar atau berbicara untuk berkomunikasi dengan orang lain. Menurut Clark (1999), bahasa isyarat ialah sebuah cara berkomunikasi tanpa suara (non verbal communication) yang menggunakan gerakan bahasa tubuh sebagai simbol komunikasi.

### 2.5 Bahasa Isyarat Indonesia (BISINDO)

BISINDO adalah bahasa isyarat murni yang digunakan oleh teman-teman tuli sesuai dengan pemahaman mereka terhadap lingkungan sekitar (Palfreyman, 2015). BISINDO memungkinkan para penggunanya untuk berkomunikasi menggunakan gerakan tangan,

mimik wajah dan bahasa tubuh. BISINDO memiliki struktur bahasa yang unik yang terdiri dari elemen-elemen seperti fonologi (sistem fonetik), morfologi (struktur kata), sintaksis (struktur kalimat), dan semantik (makna). Selain itu, BISINDO dapat mengkomunikasikan makna yang abstrak dan kompleks dan dapat digunakan dalam berbagai konteks. Contoh kosakata BISINDO dapat dilihat pada Gambar 2.4.



**Gambar 2.4** Ilustrasi Bahasa Isyarat Indonesia  
(Berkenalan dengan Bahasa Isyarat Indonesia, 2009)

## 2.6 MediaPipe

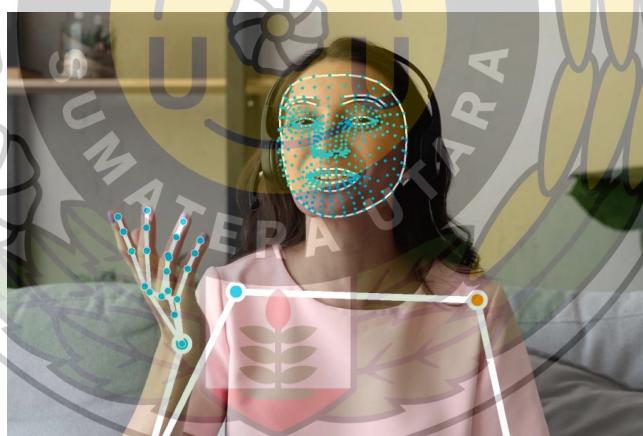
*MediaPipe* adalah *framework open source* yang didesain oleh Google dan dapat digunakan sebagai pipeline untuk melakukan inferensi dari data citra (Lugaresi et al., 2019). Pipeline dalam konteks ini adalah serangkaian proses berurutan yang menghubungkan berbagai komponen dalam pemrosesan data, mulai dari akuisisi data, pemrosesan, hingga menghasilkan *output*. Pada *MediaPipe*, pipeline ini menangani data citra dari video yang diambil, mengubahnya menjadi data *landmarks* tiga dimensi, dan kemudian meneruskannya ke *model machine learning* untuk dianalisis.

*Machine learning*, yang merupakan cabang kecerdasan buatan, memungkinkan sistem untuk mengenali pola dalam data tanpa perlu diprogram secara eksplisit. Dalam *MediaPipe*, *model machine learning* digunakan untuk mendeteksi dan melacak titik-titik penting pada tubuh, seperti tangan, wajah, atau tubuh, yang dikenal sebagai *landmarks*. *Landmarks* ini berupa titik koordinat tiga dimensi yang dinormalisasi dalam rentang [0, 1], dan dengan akurasi tinggi, dapat membantu dalam mengenali pola gerakan yang nantinya bisa diterjemahkan menjadi hasil yang bermanfaat, seperti teks atau suara.

*MediaPipe* tidak hanya menyediakan berbagai jenis solusi siap pakai untuk pemrosesan data sensorik, tetapi juga menawarkan antarmuka pemrograman aplikasi (API) yang memungkinkan pengembang untuk mengakses dan menggunakan komponen-komponen ini secara fleksibel dalam pengembangan aplikasi mereka. Salah satu implementasi *MediaPipe* yang dapat dilakukan adalah pembuatan *dataset* bahasa isyarat dengan data berupa *landmarks* tiga dimensi. *MediaPipe* mencakup berbagai jenis yang siap digunakan untuk tugas-tugas umum yaitu sebagai berikut :

#### 2.6.1 *MediaPipe Holistic*

*MediaPipe Holistic* merupakan solusi komprehensif yang mengintegrasikan deteksi dan pelacakan *pose* tubuh, wajah, dan tangan, dengan memanfaatkan *model landmark* untuk mendeteksi 468 *landmark* wajah, 33 *pose* tubuh, dan 21 *landmark* kerangka tangan yang menghasilkan total 543 *landmark* (Lugaresi et al., 2019). Dengan memanfaatkan *model* pembelajaran mesin yang canggih, *MediaPipe Holistic* mampu memberikan prediksi yang sangat akurat untuk setiap komponen (*pose*, wajah, tangan), bahkan dalam kondisi pencahayaan dan orientasi yang beragam. Semua data dari deteksi *pose*, wajah, dan tangan digabungkan dalam satu pipeline untuk memberikan analisis yang holistik.



**Gambar 2. 5** Mediapipe Holistic (Lugaresi et al., 2019)

#### 2.6.2 *MediaPipe Face Mesh*

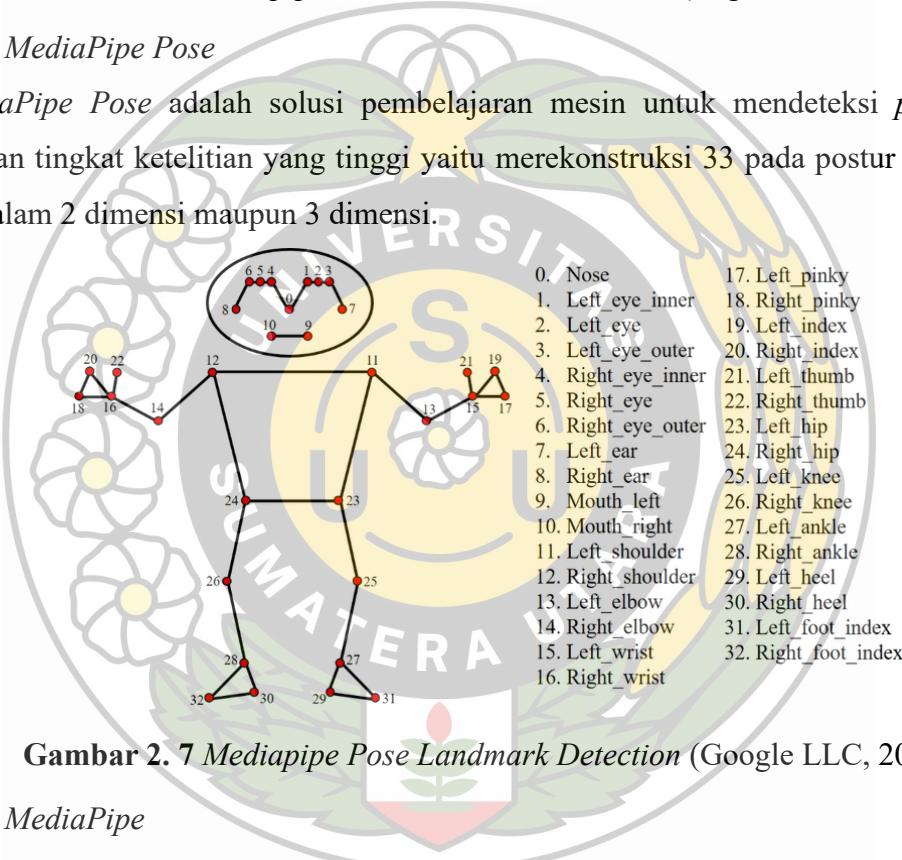
*MediaPipe Face Mesh* mampu mendeteksi 468 *landmark* tiga dimensi pada wajah secara *real time*, memungkinkan aplikasi untuk memetakan fitur wajah dengan presisi tinggi dan kecepatan yang cukup untuk digunakan dalam aplikasi interaktif seperti *augmented reality*, analisis ekspresi wajah, dan berbagai teknologi pengenalan wajah.



**Gambar 2. 6 Mediapipe Face Landmark Detection** (Lugaresi et al., 2019)

#### 2.6.3 MediaPipe Pose

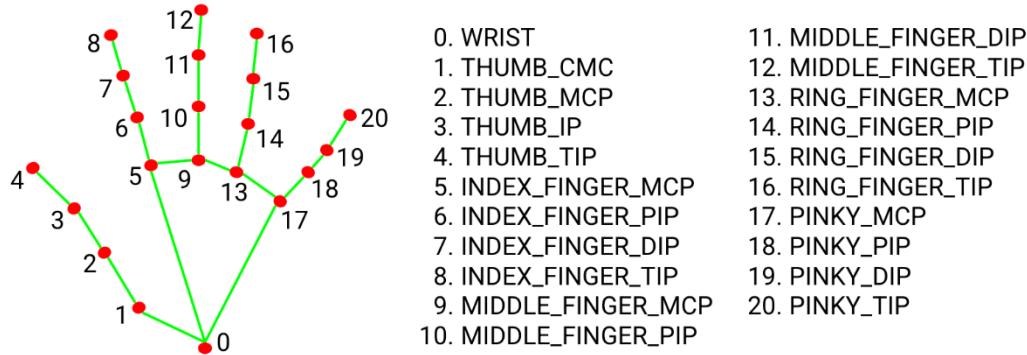
*MediaPipe Pose* adalah solusi pembelajaran mesin untuk mendeteksi *pose* tubuh dengan tingkat ketelitian yang tinggi yaitu merekonstruksi 33 pada postur tubuh baik itu dalam 2 dimensi maupun 3 dimensi.



**Gambar 2. 7 Mediapipe Pose Landmark Detection** (Google LLC, 2020)

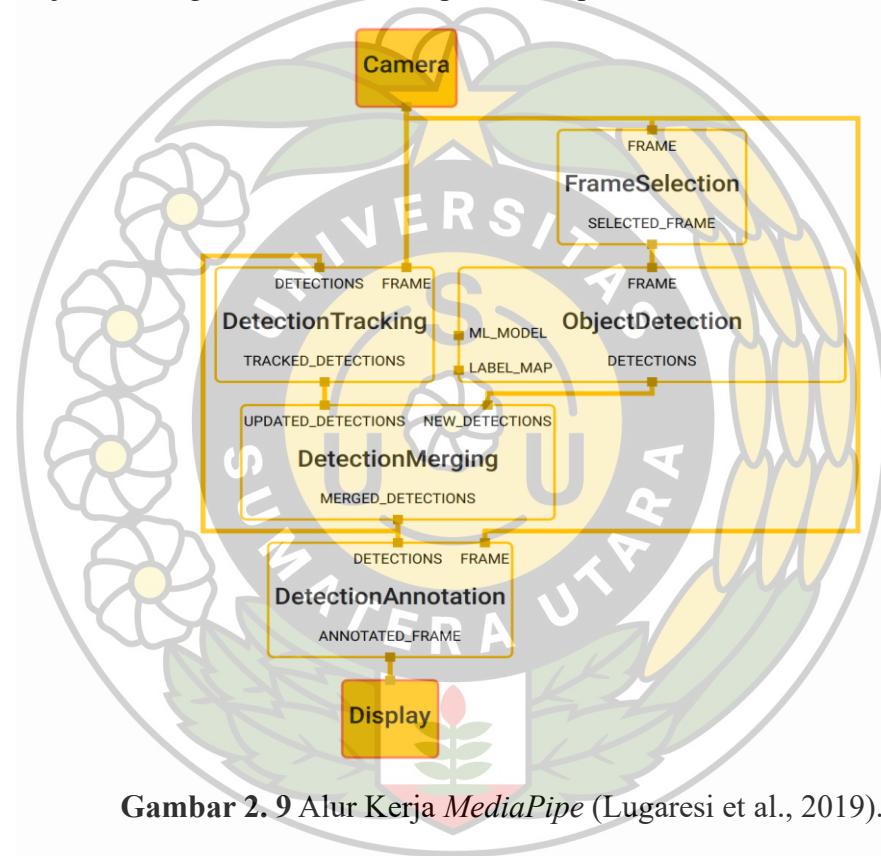
#### 2.6.4 MediaPipe

*MediaPipe* adalah salah satu solusi dari *framework MediaPipe* yang dirancang untuk deteksi dan pelacakan tangan secara *real-time*. Solusi ini menggunakan *model* pembelajaran mesin untuk mendeteksi posisi tangan dan mendeteksi 21 titik *landmark* pada setiap tangan, termasuk jari-jari dan pergelangan tangan dalam satu *frame*.



**Gambar 2. 8 Mediapipe Landmarks Detection** (Google LLC, 2020)

Alur kerja *MediaPipe* secara umum dapat dilihat pada Gambar 2.9.

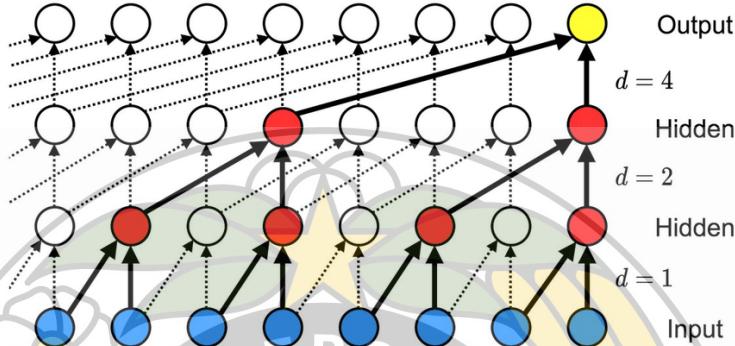


**Gambar 2. 9 Alur Kerja MediaPipe** (Lugaresi et al., 2019).

## 2.7 Temporal Convolutional Network (TCN)

*Temporal Convolutional Network* (TCN) merupakan salah satu algoritma *deep learning* yang dirancang khusus untuk memproses data urutan atau *time series*. TCN merupakan salah satu pengembangan dari *Convolutional Neural Network* (CNN) yang sesuai untuk data *sequential* yang memiliki cakupan area yang luas seperti video. Pada penelitian ini, *dataset* yang digunakan berupa data sekuensial yang diekstrak dari video gerakan isyarat, di mana bahasa isyarat, termasuk BISINDO, terdiri dari gerakan dinamis tangan dan tubuh yang terjadi dalam urutan tertentu. Algoritma TCN

sangat efektif dalam mengenali pola-pola dalam data urutan karena strukturnya yang menggunakan konvolusi kausal dan dilatasi. Hal ini memungkinkan TCN untuk menangkap hubungan temporal yang kompleks dari gerakan-gerakan tersebut, sehingga dianggap cocok untuk penelitian ini. TCN adalah struktur hierarkis yang terdiri dari beberapa lapisan tersembunyi konvolusional yang memiliki dimensi yang sama dengan lapisan *input*. Arsitektur dari TCN dapat dilihat pada gambar 2.11.



Gambar 2. 10 Arsitektur Umum TCN (Lee et al., 2021)

TCN menggunakan tiga teknik utama yaitu, konvolusi kausal, konvolusi dilatasi, dan konvolusi residual.

### 1. Konvolusi Kausal

Konvolusi kausal adalah jenis konvolusi di mana *output* pada setiap titik waktu hanya bergantung pada *input* pada titik waktu sebelumnya atau pada titik waktu yang sama dari lapisan sebelumnya. Dalam konvolusi ini, tidak ada informasi dari masa depan yang digunakan untuk menghasilkan *output* saat ini.

### 2. Konvolusi Dilatasi

Konvolusi dilatasi melibatkan penggunaan nilai dilatasi (atau jarak antar titik) yang lebih besar daripada satu dalam proses konvolusi. Konvolusi ini memungkinkan jarak yang lebih besar antara titik-titik di mana *filter* konvolusi diterapkan pada *input*. Dengan menggunakan nilai dilatasi yang lebih besar, lapisan konvolusi dapat menangkap informasi yang lebih luas dari *input*. Hal ini berguna terutama dalam pengenalan pola di tingkat yang lebih tinggi atau dalam memahami konteks yang lebih luas dari data masukan. *Kernel* konvolusi tetap konsisten di seluruh lapisan, namun faktor dilatasi meningkat secara eksponensial seiring dengan tingkat kedalaman jaringan. Seperti pada gambar 2.11 nilai dilatasi  $d_1$  adalah 1 pada lapisan pertama yang menunjukkan  $d_1$  merupakan konvolusi reguler. Kemudian nilai dilatasi tersebut meningkat secara bertahap pada setiap lapisan berikutnya hingga

mencapai 4 pada lapisan tersembunyi terakhir. Struktur piramidal ini dan proses penggabungan yang efisien meningkatkan jangkauan perseptif dari TCN sehingga memungkinkannya untuk menangani masukan urutan yang panjang. Konvolusi dilatasi dinyatakan pada rumus sebagai berikut :

$$F(s) = \sum_{i=0}^{k-1} f(i) \cdot xs - d \cdot i \quad (2)$$

Keterangan:

$F(s)$  : *output* dari konvolusi dilasi pada posisi  $s$

$k$  : panjang *kernel* konvolusi

$f(i)$  : nilai *kernel* konvolusi pada posisi  $i$

$x$  : *input* atau sinyal yang akan di-konvolusi

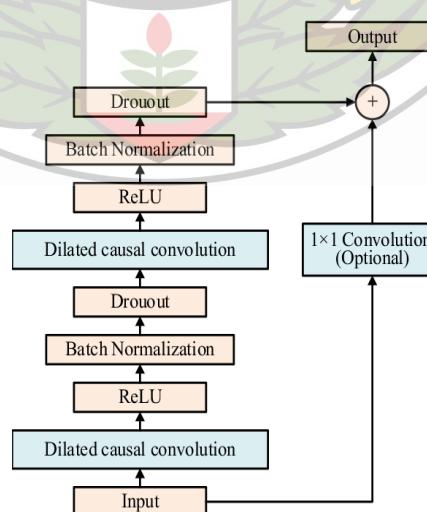
$s$  : posisi atau indeks pada *output*

$d$  : faktor dilasi yang menentukan jarak antar elemen dalam konvolusi

$xs - d \cdot i$  : nilai *input* pada posisi  $s - d \cdot i$ , di mana  $s$  adalah posisi saat ini dalam *output*, dan  $d \cdot i$  menentukan penempatan elemen *kernel* konvolusi dalam *input* dengan faktor dilasi  $d$

### 3. Konvolusi Residual

Konvolusi residual adalah teknik yang digunakan untuk mempercepat dan meningkatkan pelatihan jaringan saraf dalam membangun *model* yang lebih dalam. Dalam konvolusi residual, identitas atau sisa (residual) dari *input* awal diperkenalkan ke lapisan selanjutnya, bersamaan dengan hasil konvolusi dari *input* tersebut.



**Gambar 2. 11 TCN Residual Block (Shang et al., 2021)**

*Residual block* bekerja dengan memperkenalkan koneksi pintasan (shortcut connections) yang langsung melewaskan *input* ke *output* blok, sehingga memungkinkan gradien melewati beberapa lapisan tanpa perubahan yang signifikan.

$$o = \sigma (x + F(x)) \quad (2)$$

dimana :

$o$  : *output* dari blok residual

$\sigma$  : fungsi aktivasi (misalnya, ReLU, sigmoid, atau fungsi aktivasi lainnya)

$x$  : *input* asli yang masuk ke dalam blok residual

$F(x)$  : Fungsi transformasi yang diaplikasikan pada *input*  $x$ , yang mencakup serangkaian operasi seperti konvolusi, *batch normalization*, dan aktivasi.

Kode semu untuk proses pelatihan dengan menggunakan algoritma *Temporal Convolutional Network* dapat dilihat pada Algoritma 2.1.

#### Algoritma 2.1: Pelatihan Temporal Convolutional Network

Model Initialization and Parameters	
1	INPUT: training data ( $tr\_features$ , $tr\_labels$ ), validation data ( $v\_features$ , $v\_labels$ )
2	CREATE TCN model with Conv1D, BatchNorm, Dropout, and Dense layers
3	SET optimizer (Adam) and loss function (CrossEntropy)
4	DEFINE batch size, epochs, and learning rate
5	FOR each epoch:
6	TRAIN the model on training data ( $tr\_features$ , $tr\_labels$ )
7	VALIDATE the model on validation data ( $v\_features$ , $v\_labels$ )
8	SAVE the model if it achieves the best performance
9	PRINT training results
10	END FOR
11	LOAD the model with the best weights

#### 2.8 Confusion Matrix

*Confusion matrix* adalah tabel yang digunakan untuk mengevaluasi kinerja *model* klasifikasi dengan membandingkan nilai prediksi dan aktual dari sebuah *dataset*. *Confusion matrix* terdiri dari empat nilai: *true positives* (TP), *true negatives* (TN), *false positives* (FP), dan *false negatives* (FN) (Jiao & Du, 2016). *True positives* mewakili jumlah nilai positif yang diprediksi dengan benar, sedangkan *true negatives* mewakili jumlah nilai negatif yang diprediksi dengan benar. *False positives* mewakili jumlah nilai

negatif yang salah diprediksi sebagai positif, dan *false negatives* mewakili jumlah nilai positif yang salah diprediksi sebagai negatif.

		Prediction		
		Positives	Negatives	
Real	Positives	TP	FN	$RP = TP + FN$
	Negatives	FP	TN	
		$PP = TP + FP$	$PN = TN + FN$	$RN = FP + TN$

Gambar 2. 12 Confusion Matrix (Jiao & Du, 2016)

Confusion matrix dapat digunakan untuk menghitung berbagai metrik kinerja model klasifikasi, seperti:

### 1. Accuracy

Accuracy adalah rasio antara jumlah prediksi yang benar dengan jumlah total data yang diuji. Accuracy mengukur seberapa sering model memberikan prediksi yang benar. Untuk menghitung nilai accuracy menggunakan persamaan berikut :

$$\text{Accuracy} = \frac{TP + TN \text{ (Jumlah data uji benar)}}{TP + TN + FP + FN \text{ (Total data uji)}} \times 100$$

### 2. Precision

Precision adalah rasio antara jumlah prediksi positif yang benar dengan jumlah total prediksi positif. Precision mengukur seberapa sering prediksi positif benar. Untuk menghitung nilai precision menggunakan persamaan berikut:

$$\text{Precision} = \frac{TP}{FP + TP}$$

### 3. Recall

*Recall* adalah rasio antara jumlah prediksi positif yang benar dengan jumlah total data yang sebenarnya positif. *Recall* mengukur seberapa sering *model* dapat mengidentifikasi data yang sebenarnya positif. Untuk menghitung nilai *recall* menggunakan persamaan berikut:

$$\text{Recall} = \frac{TP}{FN + TP}$$

#### 4. *F1 score*

*F1 score* adalah rata-rata harmonik antara *precision* dan *recall*. *F1 score* mengukur keseimbangan antara *precision* dan *recall*. Untuk menghitung nilai *F1 score* menggunakan persamaan berikut:

$$F1Score = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

### 2.9 Penelitian Terdahulu

Penelitian yang dilakukan oleh Halder & Tayade (2021) berjudul “*Real-time Vernacular Sign Language Recognition using MediaPipe and machine learning*” menyoroti masalah komunikasi yang dihadapi oleh komunitas tunarungu dan bagaimana perkembangan terbaru dalam kecerdasan buatan dapat membantu mengatasi hambatan ini. Penelitian ini bertujuan untuk menunjukkan metodologi yang mempermudah pengenalan Bahasa Isyarat menggunakan kerangka kerja *MediaPipe* dan algoritma *Support Vector Machine* (SVM). Dengan menggunakan berbagai *dataset* Bahasa Isyarat, seperti Amerika, India, Italia, dan Turki, *model* prediktif yang dihasilkan menunjukkan akurasi rata-rata 99%, yang efisien, tepat, dan kuat. *Dataset* yang digunakan dalam penelitian ini mencakup alfabet Bahasa Isyarat Amerika dengan 156.000 gambar yang dibagi ke dalam 26 kelas, alfabet Bahasa Isyarat India dengan 4.972 gambar dalam 24 kelas, alfabet Bahasa Isyarat Italia dengan 12.856 gambar dalam 22 kelas, angka Bahasa Isyarat Amerika dengan 1.400 gambar dalam 10 kelas, serta angka Bahasa Isyarat Turki dengan 4.124 gambar dalam 10 kelas. *Input* dari penelitian ini adalah isyarat statis yang membentuk angka dan alfabet, sedangkan *output* yang dihasilkan berupa teks.

Penelitian oleh (Auziqni, 2022) dalam penelitiannya yang berjudul “*BISINDO Indonesian Sign Language Recognition Using MediaPipe Holistic and LSTM Deep learning Model*” menjelaskan bahwa penelitian ini merancang sistem untuk dapat

mengenali BISINDO sebagai bahasa isyarat yang umum digunakan di Indonesia untuk mempermudah komunikasi antar teman tuli-bisu pengguna bahasa isyarat Indonesia dengan orang normal. Penelitian ini menggunakan kerangka kerja *MediaPipe Holistic* untuk mengekstrak *landmarks* pada tangan, wajah dan gerakan tubuh dan menggunakan Long Short Term Memory (LSTM) sebagai *model* pembelajaran untuk memproses klasifikasi bahasa isyarat. *Dataset* yang digunakan berupa 780 gambar isyarat alfabet BISINDO yang dibagi menjadi 26 kelas alfabet A-Z dengan masing-masing 30 gambar setiap alfabet. *Input* dari penelitian ini berupa gambar isyarat alfabet Bahasa Isyarat Indonesia dengan *output* teks. Adapun hasil yang diperoleh dari penelitian ini ialah mendapatkan akurasi sebesar 97% dan skor *F1* sebesar 96%.

Kemudian penelitian yang dilakukan oleh Suyudi et al. (2022) yang berjudul “Pengenalan Bahasa Isyarat Indonesia menggunakan *MediaPipe* dengan *Model Random Forest* dan *Multinomial Logistic Regression*”. Penelitian ini menggunakan metode pembelajaran *Random Forest* dan *Multinomial Logistic Regression* dan menggunakan kerangka kerja *MediaPipe* dalam membuat sistem yang dapat mendeteksi pengenalan Sistem Isyarat Bahasa Indonesia (SIBI). Hasil yang diperoleh dari penelitian mendapatkan nilai akurasi yang cukup tinggi pada *model Random Forest* dengan nilai akurasi sebesar 97% sedangkan pada *model Multinomial Logistic Regression* memperoleh nilai akurasi sebesar 96%.

Penelitian yang dilakukan oleh Bora et al. (2023) yang berjudul “*Real-time Assamese Sign Language Recognition using MediaPipe and Deep learning*” menjelaskan bahwa penelitian ini mengembangkan alat teknologi komunikasi untuk bahasa isyarat secara *real-time*. Adapun bahasa isyarat yang akan dideteksi oleh sistem ialah bahasa isyarat Assam yang merupakan salah satu dari 22 bahasa modern yang terdapat di India. *Dataset* dikumpulkan menggunakan sensor *Microsoft Kinect* berupa 2094 titik data dari Bahasa Isyarat Assam. *Dataset* ini terdiri dari sembilan gestur tangan statis yang mewakili huruf vokal dan konsonan dari alfabet Assam, termasuk gestur untuk huruf অ, ই, ঈ, উ, এ, ও, ক, ঙ, and ল. Setiap kelas memiliki sekitar 200-300 titik data. *Input* dari program adalah gambar-gambar gestur tangan statis, *output* dari program adalah teks yang merupakan hasil prediksi huruf-huruf yang diwakili oleh gestur tangan tersebut. Penelitian ini menggunakan *framework MediaPipe* yang dimana *framework* ini berfungsi untuk mendeteksi *landmark* pada kumpulan data gambar 2D, 3D dan dinilai cukup akurat serta efektif dengan karakteristik ringan dan kokoh pada

pengimplementasiannya. Metode yang dipakai dalam penelitian ini ialah *Deep learning* yaitu *Feedforward Neural Network* dan menghasilkan akurasi yang cukup tinggi dengan nilai akurasi sebesar 99%.

Selanjutnya penelitian yang dilakukan oleh Singh & Koundal (2024) dengan judul penelitian “*A Temporal Convolutional Network for Modeling Raw 3D Sequences and Air-Writing Recognition*” menjelaskan bahwa sistem penulisan udara, yang berkembang sebagai metode interaksi dalam berbagai lingkungan elektronik seperti realitas virtual, menghadapi kompleksitas lebih tinggi dibanding penulisan 2D karena kurangnya informasi goresan dan titik referensi di ruang 3D. Namun, dengan pemodelan menyeluruh dari trayektori 3D, kompleksitas ini dapat diatasi. *Temporal Convolutional Networks* (TCN) telah diusulkan sebagai solusi untuk sistem pengenalan penulisan udara. TCN, yang memodelkan urutan data secara akurat, diterapkan pada tiga *dataset* publik yang berisi angka dan karakter yang ditulis di udara oleh berbagai penulis. Jaringan dilatih menggunakan trayektori 3D mentah tanpa fitur buatan tangan, dengan hasil dihitung melalui skema validasi silang 10 kali lipat. Hasil penelitian menunjukkan efektivitas TCN dalam pengenalan karakter 3D, dengan akurasi 99,50% untuk angka dan 99,56% untuk karakter. Strategi ini terbukti lebih efektif dibanding metode yang ada. Evaluasi di masa depan dapat dilakukan dalam skenario independen pengguna untuk aplikasi dunia nyata.

Penelitian yang dilakukan oleh (Tsinganos et al., 2022) dengan penelitiannya yang berjudul “*Real-Time Analysis of Gesture Recognition with Temporal Convolutional Networks*”. Studi ini mengusulkan penggunaan *Temporal Convolutional Networks* (TCNs) untuk mengatasi masalah pengenalan gestur tangan berbasis elektromiografi (sEMG). Dalam evaluasi offline menggunakan *dataset* Ninapro DB1, *model* TCN berhasil mengklasifikasikan sinyal sEMG dari 53 gestur dengan baik. Namun, dalam evaluasi *real-time* dengan segmen 200 ms, akurasi klasifikasi menurun, mengindikasikan bahwa arsitektur TCN yang diusulkan mungkin tidak cocok untuk aplikasi *real-time*. Studi ini menyoroti pentingnya pengembangan metode pemberian data yang disesuaikan untuk meningkatkan performa *model* TCN dalam situasi penggunaan yang lebih dinamis. Hasil penelitian ini menghasilkan *model* dengan akurasi terbaik sebesar 78,67%.

Kemudian penelitian yang dilakukan oleh (Ishaq et al., 2023) dengan penelitiannya yang berjudul “*TC-Net: A Modest & Lightweight Emotion Recognition System Using*

*Temporal Convolution Network*". Studi ini menyoroti pentingnya sinyal suara dalam komunikasi serta peran mereka dalam pertukaran informasi efisien antara manusia dan mesin. Pengenalan Emosi dalam Ucapan (*Speech Emotion Recognition*, SER) menjadi krusial untuk evaluasi manusia dalam berbagai aplikasi praktis seperti perawatan kesehatan, pusat panggilan, robotika, keamanan, dan realitas virtual. Penelitian ini memperkenalkan sistem pengenalan emosi berbasis *Temporal Convolutional Network* (TCN), yang menggunakan jaringan konvolusi spasial-temporal untuk mengidentifikasi keadaan emosional pembicara berdasarkan sinyal suara. Model TCN dirancang untuk mengenali dependensi jangka panjang dalam sinyal suara, kemudian mengintegrasikan petunjuk temporal ini dengan fitur spasial melalui jaringan padat untuk klasifikasi akhir. Sistem ini secara otomatis mengekstrak petunjuk urutan yang valid dari sinyal suara, mengungguli kinerja algoritma *machine learning* tradisional dan *state-of-the-art* (SOTA). Hasil eksperimen menunjukkan akurasi pengenalan tinggi, dengan akurasi akhir tanpa bobot mencapai 80.84% untuk *dataset Interactive Emotional Dyadic Motion Captures* (IEMOCAP) dan 92.31% untuk *dataset Berlin Emotional Database* (EMO-DB), menegaskan keandalan dan efisiensi *model* yang dikembangkan.

Penelitian yang dilakukan oleh (Scherer et al., 2020) dengan judul penelitian "*TinyRadarNN: Combining Spatial and Temporal Convolutional Neural Networks for Embedded Gesture Recognition With Short Range Radars*". Penelitian ini memperkenalkan sebuah algoritma pengenalan gerakan tangan yang hemat energi menggunakan sensor RADAR jarak pendek. Algoritma ini mengkombinasikan jaringan saraf konvolusional (CNN) 2-D dengan jaringan saraf konvolusional temporal (TCN) untuk prediksi urutan waktu, menghasilkan *model* yang sangat efisien dengan ukuran 46 ribu parameter dan hanya membutuhkan memori sebesar 92 KB. Pengujian dilakukan pada dua set data yang mencakup 11 jenis gerakan tangan berbeda yang dilakukan oleh 26 orang, dengan total 20,210 contoh sinyal. Hasil pengujian menunjukkan tingkat akurasi sebesar 86.6% untuk 26 pengguna dan 92.4% untuk pengguna tunggal, sebanding dengan teknologi canggih yang menggunakan jaringan TCN yang jauh lebih besar. Implementasi algoritma ini pada prosesor dengan daya sangat rendah menunjukkan kemampuan prediksi *real-time* dengan konsumsi daya yang sangat minim, yaitu hanya 21 mW untuk jaringan prediksi urutan TCN. Penelitian ini juga menyediakan akses terbuka ke kode dan data yang digunakan, dengan tujuan utama

untuk penerapan pengenalan gerakan tangan pada perangkat tertanam menggunakan RADAR berdaya rendah.

Selanjutnya, penelitian yang dilakukan oleh Zhang et al. (2019) yang berjudul “*Short-Term Temporal Convolutional Networks for Dynamic Gesture Recognition*” yang menjelaskan bahwa pengakuan gerakan bertujuan untuk mengenali gerakan tubuh manusia yang bermakna, yang merupakan masalah penting dalam visi komputer. Makalah ini menyajikan metode pengenalan gerakan multimodal menggunakan *3D-DenseNets* untuk analisis spasial dan TCNs yang ditingkatkan untuk analisis temporal. Metode ini dievaluasi pada VIVA dan NVIDIA *Gesture Datasets*, mencapai akurasi 91,54% pada VIVA dan 86,37% pada NVIDIA, menunjukkan kinerja yang sangat kompetitif dan terbaik.

**Tabel 2. 1** Penelitian Terdahulu

No	Penulis	Tahun	Metode	Keterangan
1.	Halder & Tayade	2021	SVM	Penelitian ini membahas pengenalan bahasa isyarat berbagai negara (Amerika, India, Italia, Turki) menggunakan <i>MediaPipe</i> dan <i>machine learning</i> . Akurasi rata-rata yang diperoleh adalah 99%. <i>Dataset</i> terdiri dari 156.000 gambar alfabet ASL, 4.972 gambar alfabet India, 12.856 gambar alfabet Italia, 1.400 gambar angka ASL, dan 4.124 gambar angka Turki.
2.	Auziqni	2022	LSTM	Penelitian ini membahas pengenalan Bahasa Isyarat Indonesia (BISINDO) menggunakan <i>MediaPipe Holistic</i> dan LSTM. Akurasi yang diperoleh adalah 97% dengan skor <i>F1</i> 96%. <i>Dataset</i> terdiri dari 780 gambar alfabet BISINDO yang dibagi

				menjadi 26 kelas (A-Z) dengan masing-masing 30 gambar per alfabet.
3.	Suyudi et al.	2022	<i>Random Forest, Multinomial Logistic Regression</i>	Penelitian ini membahas pengenalan Sistem Isyarat Bahasa Indonesia (SIBI) menggunakan <i>MediaPipe</i> dan <i>model Random Forest</i> serta <i>Multinomial Logistic Regression</i> . Akurasi yang diperoleh adalah 97% untuk <i>Random Forest</i> dan 96% untuk <i>Multinomial Logistic Regression</i> .
4.	Bora et al.	2023	<i>Feedforward Neural Network</i>	Penelitian ini membahas pengenalan bahasa isyarat Assam menggunakan <i>MediaPipe</i> dan <i>Feedforward Neural Network</i> . Akurasi yang diperoleh mencapai 99%. <i>Dataset</i> terdiri dari 2.094 titik data Bahasa Isyarat Assam, dengan sekitar 200-300 titik data per kelas yang mewakili 9 gestur tangan statis.
5.	Singh & Kounda 1	2024	<i>Temporal Convolutional Network</i> (TCN)	Penelitian ini membahas pengenalan penulisan udara ( <i>air-writing</i> ) menggunakan <i>Temporal Convolutional Network</i> (TCN). Akurasi yang diperoleh adalah 99,50% untuk angka dan 99,56% untuk karakter. <i>Dataset</i> terdiri dari tiga <i>dataset</i> publik yang berisi angka dan karakter yang ditulis di udara.

6.	Tsingan os et al.	2022	<i>Temporal Convolutional Networks</i> (TCNs)	Penelitian ini membahas pengenalan gestur tangan berbasis sinyal elektromiografi (sEMG) menggunakan <i>Temporal Convolutional Networks</i> (TCN). Akurasi yang diperoleh dalam evaluasi <i>real-time</i> adalah 78,67%. <i>Dataset</i> menggunakan Ninapro DB1 dengan sinyal sEMG dari 53 gestur tangan.
7.	Ishaq et al.	2023	<i>Temporal Convolutional Network</i> (TCN), <i>Spatial-Temporal Convolution</i>	Penelitian ini membahas pengenalan emosi dalam ucapan menggunakan <i>Temporal Convolutional Network</i> (TCN). Akurasi yang diperoleh adalah 80,84% untuk <i>dataset</i> IEMOCAP dan 92,31% untuk <i>dataset</i> EMO-DB. <i>Dataset</i> terdiri dari sinyal suara emosional untuk berbagai keadaan emosi dari pembicara manusia.
8.	Scherer et al.	2020	2D-CNN, <i>Temporal Convolutional Network</i> (TCN)	Penelitian ini membahas pengenalan gestur tangan menggunakan sensor RADAR jarak pendek dan kombinasi 2D-CNN serta <i>Temporal Convolutional Network</i> (TCN). Akurasi yang diperoleh adalah 92,4% untuk pengguna tunggal dan 86,6% untuk 26 pengguna. <i>Dataset</i> terdiri dari 20.210 contoh sinyal dari 11 jenis gerakan tangan yang dilakukan oleh 26 orang.
9.	Zhang et al.	2019	<i>Temporal Convolutional Network</i> (TCN)	Penelitian ini membahas pengenalan gestur dinamis menggunakan kombinasi 3D-DenseNet untuk analisis spasial dan TCN untuk analisis temporal. Akurasi yang diperoleh adalah 91,54% untuk <i>dataset</i> VIVA dan 86,37% untuk <i>dataset</i> NVIDIA. <i>Dataset</i> terdiri dari data gestur tangan dari kedua <i>dataset</i> tersebut.

Perbedaan utama antara penelitian yang dilakukan oleh penulis dengan penelitian terdahulu terletak pada penggunaan *dataset* berupa video kata dalam Bahasa Isyarat Indonesia (BISINDO), sedangkan penelitian sebelumnya yang menggunakan *MediaPipe* lebih banyak berfokus pada pengenalan abjad dan angka dalam bentuk gambar statis. Sebagai contoh, penelitian Halder & Tayade (2021) menggunakan *dataset* alfabet dan angka dari berbagai bahasa isyarat seperti ASL dan ISL, yang berbasis gambar, demikian juga penelitian oleh Auziqni (2022) dan Suyudi et al. (2022) yang menggunakan *dataset* gambar alfabet BISINDO dan SIBI. Sebaliknya, penelitian ini menggunakan *dataset* video kata isyarat yang bersifat dinamis, yang menambah tantangan dalam pemrosesan temporal karena melibatkan analisis urutan gerakan tangan. Penulis menggunakan algoritma *Temporal Convolutional Network* (TCN) yang mampu menangani data sekuensial dari video untuk mengenali pola gerakan tangan dengan lebih akurat dalam konteks kata dan kalimat, bukan hanya huruf atau angka secara statis.



## **BAB 3**

### **ANALISIS DAN PERANCANGAN SISTEM**

#### **3.1 Dataset**

Data yang digunakan pada penelitian ini berupa video isyarat dari Bahasa Isyarat Indonesia (BISINDO). Pengumpulan *dataset* dilakukan di SLB-B Karya Murni, Jl. Haji Muhammad Joni No.66 A, TELADAN TIMUR, Kec. Medan Kota, Kota Medan, Sumatera Utara didampingi oleh Suster Epi Samosir. Video ini dikumpulkan dengan merekam isyarat yang diperagakan oleh teman tuli-bisu dan diperbanyak oleh penulis sendiri. *Dataset* berekstensi .mp4 dengan resolusi 1920x1080 piksel dan 30FPS dengan total *dataset* sebanyak 2000 video. Contoh data video yang dipakai pada penelitian dapat dilihat pada Gambar 3.1.



**Gambar 3. 1** Contoh data Bahasa Isyarat  
(Sumber: Dokumen Pribadi)

Jumlah isyarat pada penelitian ini sebanyak 40 kata yang dikelompokkan menjadi 6 kategori, seperti:

1. Kata ganti orang: Dia, Kalian, Kami, Kamu, Kita, Mereka, Saya.
2. Kata tanya: Apa, Bagaimana, Berapa, Kapan, Kemana, Kenapa, Siapa.
3. Kata kerja: Bantu, Belajar, Berjalan, Duduk, Makan, Mandi, Melihat, Menulis, Minum, Tidur.

4. Kata sifat: Baik, Bingung, Marah, Pendek, Ramah, Sabar, Sedih, Senang, Tinggi.
5. Kata sapaan: Halo, Selamat Malam, Selamat Pagi, Selamat Siang, Selamat Sore.
6. Kata Ungkapan: Terima kasih, Maaf.

Pada Tabel 3.1 dapat dilihat detail dari jumlah data yang digunakan.

**Tabel 3. 1** Data Kata Isyarat

Kata Isyarat	Jumlah Data (video)		
	Dataset	Setelah Augmentasi	
Kata Ganti Orang	Dia	50	600
	Kalian	50	600
	Kami	50	600
	Kamu	50	600
	Kita	50	600
	Mereka	50	600
	Saya	50	600
Kata Tanya	Apa	50	600
	Bagaimana	50	600
	Berapa	50	600
	Kapan	50	600
	Kemana	50	600
	Kenapa	50	600
	Siapa	50	600
Kata Kerja	Bantu	50	600
	Belajar	50	600
	Berjalan	50	600
	Duduk	50	600
	Makan	50	600
	Mandi	50	600
	Melihat	50	600
	Menulis	50	600
	Minum	50	600

	Tidur	50	600
Kata Sifat	Baik	50	600
	Bingung	50	600
	Marah	50	600
	Pendek	50	600
	Ramah	50	600
	Sabar	50	600
	Sedih	50	600
	Senang	50	600
	Tinggi	50	600
	Halo	50	600
Kata Sapaan	Selamat Malam	50	600
	Selamat Pagi	50	600
	Selamat Siang	50	600
	Selamat Sore	50	600
Kata Ungkapan	Maaf	50	600
	Terima kasih	50	600
	Total	2000	24000

Penelitian ini memilih pembagian data dengan perbandingan 70% untuk pelatihan dan 30% untuk validasi karena rasio ini telah terbukti secara empiris memberikan keseimbangan yang baik antara jumlah data pelatihan dan validasi. Berdasarkan penelitian sebelumnya seperti dalam jurnal "Deep Convolutional Neural Network-Based System for Fish Classification," proporsi 70-30 memungkinkan model mendapatkan cukup banyak data untuk mempelajari pola-pola penting selama pelatihan, sambil menyediakan data validasi yang cukup untuk mengevaluasi kinerja model secara obyektif. Dengan proporsi ini, model mampu mengurangi risiko overfitting sekaligus menjaga tingkat akurasi dan generalisasi yang optimal terhadap data baru. Kombinasi ini menjadikan perbandingan 70-30 sebagai salah satu standar umum yang sering digunakan dalam penelitian berbasis machine learning dan deep learning.

**Tabel 3. 2** Pembagian Data

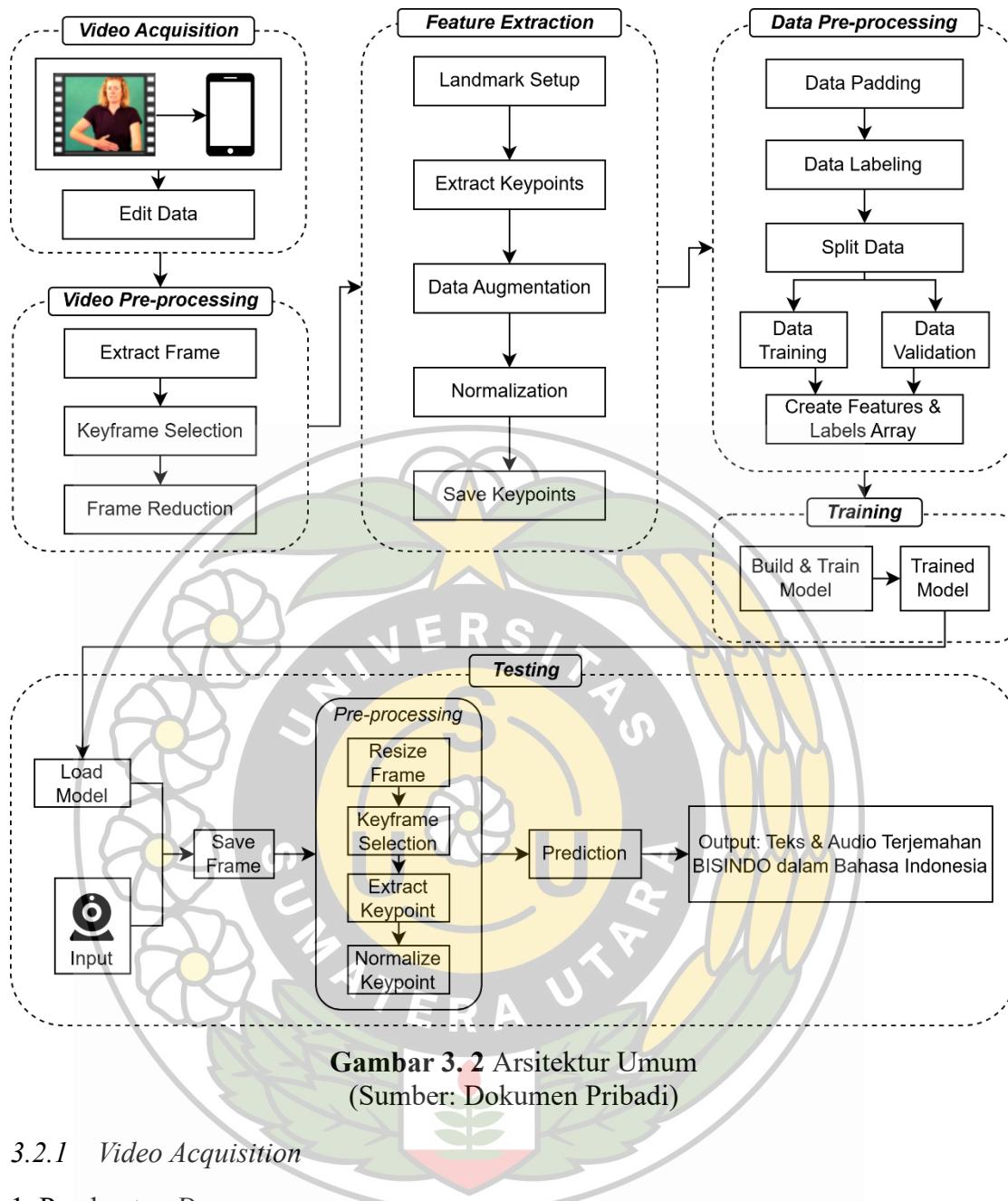
Pembagian Data	Persentase	Jumlah Data

Data Latih	70%	16800
Data Validasi	30%	7200
Total	100%	24000

Data latih dan data validasi berupa potongan gambar, data latih digunakan untuk melatih *model* TCN kemudian data validasi digunakan untuk mengevaluasi kinerja *model* selama pelatihan.

### 3.2 Arsitektur Umum

Pada bagian ini dijelaskan bagaimana alur dan proses dari penelitian yang dilakukan. Tahap pertama adalah pengumpulan data video, selanjutnya dilakukan *pre-processing* terhadap video tersebut seperti *extract frame*, *keyframe selection*, *frame reduction*. Tahap selanjutnya adalah ekstraksi fitur dari *frame* tersebut menggunakan *MediaPipe*. Selanjutnya tahap *pre-processing* data, pada tahap ini terdapat proses *data padding*, *data labeling*, *split data* (data dibagi menjadi data latih dan data validasi), dan *create features and labels array*. Setelah itu dilakukan tahap pelatihan *model*. Tahap terakhir, *model* yang sudah dilatih dipakai pada program uji untuk menguji performa *model* secara langsung. Arsitektur umum penelitian ini dapat dilihat pada Gambar 3.2.



### 3.2.1 Video Acquisition

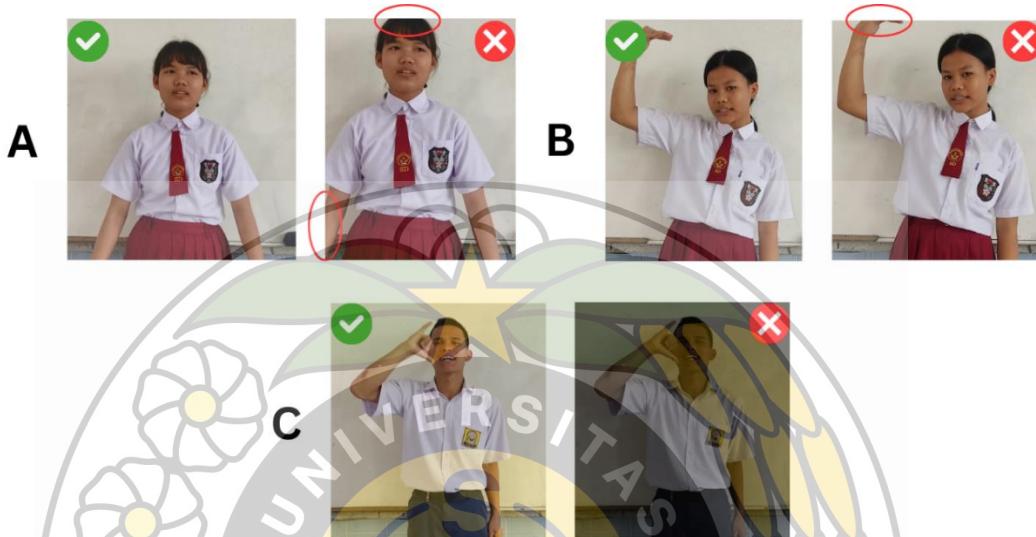
#### 1. Pembuatan Dataset

Penelitian ini dimulai dengan pengumpulan data video isyarat BISINDO.

Untuk membuat *dataset* harus mengikuti petunjuk berikut:

- Keseluruhan badan (dari pinggang hingga ujung kepala) peraga harus muncul pada setiap *frame* video seperti pada Gambar 3.3 bagian A.
- Detail gerakan isyarat harus jelas dan tidak keluar dari *frame* seperti pada Gambar 3.3 bagian B.
- Perekaman harus pada kondisi pencahayaan yang baik seperti pada Gambar 3.3 bagian C untuk memastikan *keypoint* yang ditangkap jelas.

- Rekamlah video isyarat pada latar belakang yang stabil dan tidak memuat tangan atau wajah lain selain tangan dan wajah peraga.
- Atur perangkat perekaman agar tidak bergoyang dan sefokus mungkin.
- Perangkat diletakkan pada tripod yang diatur pada ketinggian 150 cm dengan jarak dari peraga isyarat sejauh 75 cm.



**Gambar 3.3** Contoh data yang benar dan salah

Dataset pada penelitian ini berupa 40 video isyarat BISINDO yang berisi 50 kali pengulangan isyarat pada masing-masing video.

## 2. Edit Data

Setelah pengumpulan video isyarat tahap selanjutnya yang dilakukan adalah perubahan ukuran *frame* video menjadi lebih kecil agar ukuran *dataset* tidak terlalu besar sehingga waktu yang dibutuhkan untuk proses *training* lebih efisien. *Frame* video awalnya berukuran 1920 x 1080 *pixel* diubah menjadi 400 x 300 *pixel*. Kemudian dilakukan pemotongan video dari 40 video isyarat menjadi 2000 video, video tersebut disimpan ke dalam 40 *folder* dengan nama yang sesuai dengan arti isyarat tersebut masing-masing.

### 3.2.2 Video Pre-processing

Selanjutnya tahap ini dilakukan untuk mengolah video menjadi data yang dibutuhkan untuk data *training*. Adapun tahapannya yaitu *extract frame*, *keyframe selection*, *frame reduction* dengan penjelasan sebagai berikut:

### 1. Extract Frame

Proses ini dilakukan menggunakan *library OpenCV* untuk mengambil *frame* atau gambar dari sebuah video. *Frame* yang diambil disimpan ke *folder* masing-masing dengan ekstensi .jpg. Gambar di bawah ini adalah *pseudocode* dari proses ekstrak *frame*.

#### Algoritma 3. 1 Ekstrak *Frame*

```

for folder_isyarat in folder_video do
    for video in folder_isyarat do
        Buat folder penyimpanan
        video_path = (path)
        while video masih ada do
            simpan frame video ke folder penyimpanan
        end
    end
end

```

Setelah proses *extract frame*, *frame* yang didapatkan dari 2000 video sebanyak 143.800 *frame*. Gambar 3.4 adalah hasil ekstrak *frame* dari video.



Gambar 3. 4 Hasil proses Extract *Frame*

### 2. Keyframe Selection

Pemilihan *keyframe* merupakan langkah penting dalam pra-pemrosesan data pengenalan gerakan isyarat. Tujuan dari proses ini adalah untuk mengidentifikasi dan mempertahankan *frame* paling informatif dari serangkaian gambar yang menangkap gerakan tangan dan menghapus bingkai tambahan atau lebih kecil. Hal ini meningkatkan efisiensi pasca-pemrosesan dan meningkatkan akurasi prediksi. Dalam penelitian ini, proses pemilihan *keyframe* mempertimbangkan tiga variabel utama: pergerakan tangan, posisi tangan, dan tingkat keburaman tangan. Dengan menganalisis

variabel-variabel ini, algoritma memastikan bahwa *frame* yang dipilih merupakan representasi yang jelas dan lengkap dari gerakan yang sedang dilakukan.

Gerakan tangan berisyarat dimulai dengan gerakan vertikal dari bawah ke atas dan diakhiri dengan gerakan tangan turun ke posisi semula. Analisis gerakan ini dapat membantu mengidentifikasi *frame-frame* transisi. Variabel pertama dihitung dengan membandingkan koordinat tengah kedua tangan pada *frame* ke-*f* dengan *frame* ke-*f*+1. Nilai tertinggi dari hasil perbandingan ini dijadikan nilai variabel pertama. Koordinat tengah dihitung sebagai rata-rata dari koordinat y minimum dan maksimum. Perhitungan variabel pertama dijelaskan dalam persamaan di bawah ini:

$$\text{variabel}_1 = \max \left( \frac{\sqrt{(x_{f+1,1} - x_{f,1})^2 + (y_{f+1,1} - y_{f,1})^2},}{\sqrt{(x_{f+1,2} - x_{f,2})^2 + (y_{f+1,2} - y_{f,2})^2}} \right)$$

Keterangan:

- $x_{f+1,1}$  dan  $y_{f+1,1}$  adalah koordinat tangan pertama pada *frame* ke-*f*+1.
- $x_{f,1}$  dan  $y_{f,1}$  adalah koordinat tangan pertama pada *frame* ke-*f*.
- $x_{f+1,2}$  dan  $y_{f+1,2}$  adalah koordinat tangan kedua pada *frame* ke-*f*+1.
- $x_{f,2}$  dan  $y_{f,2}$  adalah koordinat tangan kedua pada *frame* ke-*f*.

Posisi tangan aktif dalam berisyarat biasanya berada di area antara kepala dan perut, sementara tangan tidak aktif berada di sekitar pinggang atau paha. Variabel posisi tangan digunakan untuk menghindari pemilihan *frame* sebelum isyarat dimulai atau setelah isyarat selesai. Nilai variabel ini adalah nilai terkecil dari koordinat tengah kedua tangan. Perhitungan variabel kedua dirumuskan dalam persamaan di bawah ini:

$$\text{variabel}_2 = \min(y_{tengah,1}, y_{tengah,2})$$

Keterangan:

- $y_{tengah,1}$  adalah koordinat y tengah tangan pertama.
- $y_{tengah,2}$  adalah koordinat y tengah tangan kedua.

Keburaman pada *frame* video menunjukkan adanya gerakan, yang mengindikasikan bahwa tangan sedang bergerak. *Variation of the Laplacian* adalah salah satu metode yang digunakan untuk mendeteksi keburaman pada gambar digital. Metode ini

mengubah citra *grayscale* menjadi konvolusi dengan *kernel Laplacian* berbentuk matriks 3x3. Hasilnya adalah citra yang hanya menampilkan tepi-tepi dari citra *input*. Untuk mendapatkan hasil yang lebih akurat, metode ini diterapkan hanya pada bagian tangan, bukan seluruh *frame*. Lokasi tangan diidentifikasi menggunakan *MediaPipe*, yang kemudian digunakan untuk memotong bagian tangan dari citra berdasarkan koordinat minimum dan maksimum dari sumbu x dan y. Tingkat keburaman tangan kemudian diukur menggunakan metode ini. Persamaan di bawah ini menjelaskan langkah-langkah dalam metode *variation of the Laplacian*.

Persamaan mengubah citra berwarna menjadi *grayscale*:

$$\text{gray} = 0.2989 \times R + 0.5870 \times G + 0.1140 \times B$$

Keterangan:

- R, G, dan B adalah nilai komponen warna merah, hijau, dan biru.

Persamaan *kernel Laplacian* yang digunakan:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Keterangan:

- *Kernel Laplacian* ini digunakan untuk menekankan tepi pada citra *grayscale*.

Persamaan menghitung nilai *Laplacian* dari citra:

$$\text{Laplacian} = \sum_{i,j} \text{gray}[i,j] \times \text{Laplacian kernel}[i,j]$$

Keterangan:

- *gray[i,j]* adalah nilai *pixel* pada posisi *i,j* dari citra *grayscale*.
- *Laplacian kernel[i,j]* adalah nilai *kernel Laplacian* pada posisi *i,j*.

Persamaan menjelaskan perhitungan variansi dari citra hasil metode *variation of the Laplacian*:

$$\text{variance} = \frac{1}{N} \sum_{i=1}^N (\text{Laplacian}[i] - \text{mean})^2$$

Keterangan:

- *N* adalah jumlah *pixel* pada citra.
- *mean* adalah nilai rata-rata dari *pixel* citra *Laplacian*.

Nilai variansi dari kedua tangan dijumlahkan untuk mendapatkan nilai dari variabel ketiga, sebagaimana dijelaskan dalam persamaan di bawah ini:

$$\text{variabel}_3 = \text{variance}_{\text{tangan},1} + \text{variance}_{\text{tangan},2}$$

Keterangan:

- $\text{variance}_{\text{tangan},1}$  adalah variansi tangan pertama.
- $\text{variance}_{\text{tangan},2}$  adalah variansi tangan kedua.

Skor setiap *frame* dihitung menggunakan persamaan, dimana setiap variabel dikalikan dengan konstanta positif yang berbeda untuk meningkatkan kontribusi variabel tersebut dalam menentukan *frame* penting:

$$\text{skor} = a \times \text{variabel}_1 + b \times \text{variabel}_2 + c \times \text{variabel}_3$$

di mana a, b, dan c adalah konstanta positif.

*Threshold* digunakan untuk menentukan *frame* penting dari kelompok *frame*. Jika skor *frame* lebih besar dari *threshold*, *frame* tersebut dianggap penting. Jika skor lebih rendah, *frame* tersebut dianggap tidak penting. Nilai *threshold* bervariasi untuk setiap kelompok *frame* dan dihitung dengan mengalikan konstanta positif dengan rata-rata skor dari kelompok tersebut. Rumus untuk menghitung *threshold* dijelaskan dalam persamaan di bawah ini:

$$\text{threshold} = k \times \text{rata-rata skor}$$

di mana k adalah konstanta positif.

*Pseudocode* dari proses *keyframe selection* dapat dilihat pada algoritma 3.2.

### Algoritma 3.2 Keyframe Selection

```

for setiap folder in DATASET_FRAME do
    with model holistic from Mediapipe do
        for setiap folder frame in folder saat ini do
            Tetapkan extracted_folder sebagai path
            Urutkan dan baca frame dari extracted_folder
            Inisialisasi daftar koordinat
            for setiap frame in frames do
                Baca gambar dan dapatkan dimensi gambar
                Deteksi Landmark tangan menggunakan Mediapipe holistic
                if tangan terdeteksi then
                    Inisialisasi laplacian_sum = 0
                    for setiap tangan in tangan do
                        Hitung kotak pembatas dan potong gambar tangan
                        Konversi potongan gambar menjadi grayscale
                        Hitung variansi Laplacian dan perbarui laplacian_sum

```

```

        Tambahkan dimensi kotak tangan ke data_tangan
    end
    Tambahkan data_tangan dan Laplacian_sum ke koordinat
else
    Tambahkan [[0, 0], [0, 0], 0] ke koordinat
end
end
Inisialisasi daftar skor
for setiap i, koordinat in enumerate(koordinat[1:]) do
    Hitung variabel_1, variabel_2, variabel_3 (Rumus 3.1-3.7)
    Hitung skor dengan penjumlahan tertimbang variabel (Rumus 3.8)
    Tambahkan skor ke daftar skor
end
Hitung threshold (Rumus 3.9)
Hapus frame pertama
for setiap i, skor in enumerate(skor) do
    if skor < threshold then
        Hapus frame pada indeks i+1
    end
end
end

```

*Frame* dengan nilai di bawah *threshold* akan dihapus, setelah proses *keyframe selection* dihasilkan jumlah data sebanyak 68.403 *frame*. Pada Gambar 3.5 adalah contoh hasil *keyframe selection*.



Gambar 3.5 Hasil proses Keyframe Selection

### 3. Frame Reduction

*Frame reduction* adalah langkah penting dalam *preprocessing* untuk deteksi gerakan isyarat, yang bertujuan mengurangi jumlah *frame* dalam *dataset*. Proses ini dilakukan setelah *keyframe selection* untuk mengatasi masalah duplikasi dan kemiripan antar *frame* yang masih ada, meskipun *frame-frame* tersebut telah dipilih berdasarkan keberadaan gerakan isyarat. Reduksi *frame* ini penting karena jumlah *frame* yang terlalu banyak dapat memperlambat proses pelatihan *model* dan mempengaruhi kinerja keseluruhan sistem.

Setelah semua *keypoint* tangan diekstraksi, proses pengurangan *frame* dimulai. Pertama, selisih absolut antara *keypoint* dari *frame* berurutan dihitung. Nilai selisih ini digunakan untuk menentukan tingkat perbedaan antara *frame-frame* tersebut. Kemudian, nilai *threshold* ditentukan dengan mengalikan rata-rata selisih dengan konstanta tertentu. *Frame* yang memiliki selisih lebih besar dari *threshold* dianggap sebagai *frame* penting dan disimpan, sedangkan *frame* lainnya dihapus.

Dengan cara ini, hanya *frame-frame* yang benar-benar unik dan berbeda yang dipertahankan, sementara *frame* yang mirip atau duplikat dihapus. Proses ini dilakukan secara iteratif untuk setiap *folder* isyarat hingga semua *frame* telah diproses. Proses ini tidak hanya mengurangi jumlah *frame* secara signifikan tetapi juga memastikan bahwa data yang digunakan untuk pelatihan *model* adalah representatif dan tidak redundan.

Melalui metode *frame reduction* ini, jumlah *frame* dalam *dataset* dapat dikurangi secara efektif tanpa mengorbankan informasi penting dari gerakan isyarat. Hal ini membantu dalam mempercepat proses pelatihan *model* dan meningkatkan efisiensi sistem deteksi gerakan isyarat.

*Pseudocode* dari proses *frame reduction* dapat dilihat pada Algoritma 3.3:

### Algoritma 3.3 Frame Reduction

```

for setiap folder di direktori dataset do
    for setiap subfolder di folder saat ini do
        Inisialisasi variabel _num, non, dan keypoint_a
        Sortir file di subfolder
        for setiap file di subfolder do
            Baca gambar
            Deteksi titik kunci tangan menggunakan Mediapipe holistic
            if titik kunci tangan terdeteksi then
                Tambahkan titik kunci ke keypoint_a
            else
                Tambahkan file ke daftar non dan hapus file
            end
            Hitung jarak titik kunci dan tentukan ambang batas (t)
            Hitung array perubahan dan temukan indeks signifikan
            Pilih hanya frame pada indeks signifikan
            Hapus frame lain yang tidak dibutuhkan
        end
    end
end

```

*Frame* dengan nilai di bawah *threshold* akan dihapus, setelah proses *keyframe selection* dihasilkan jumlah data sebanyak 20.890 *frame*. Pada Gambar 3.6 adalah contoh hasil *frame reduction*.



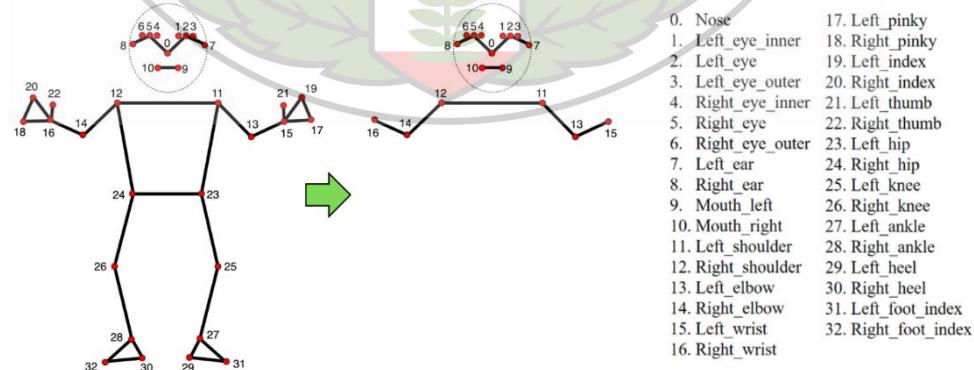
**Gambar 3. 6 Hasil proses Reduction Frame**

### 3.2.3 Feature Extraction

*Feature Extraction* bertujuan untuk mengubah data mentah menjadi representasi yang lebih sederhana dan lebih informatif. Tahap ini menggunakan *MediaPipe* yang terdapat beberapa proses didalamnya, yaitu *Landmark Setup*, *Extract Keypoints*, *Data Augmentation*, *Normalization*, dan *Save Keypoint* dengan penjelasan sebagai berikut:

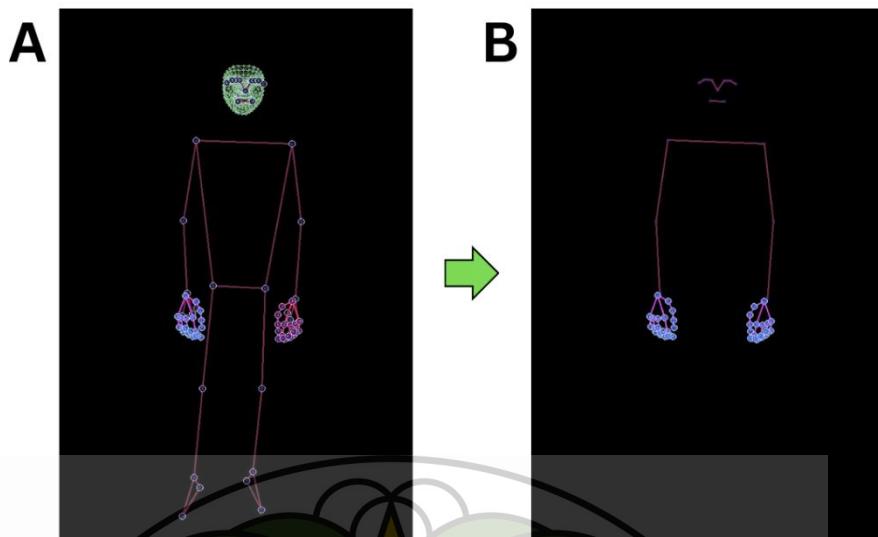
#### 1. Landmark Setup

Perlu dilakukan konfigurasi untuk mengurangi titik *landmark* yang tidak diperlukan agar pendekslsian gerakan isyarat menjadi lebih efisien. *MediaPipe Holistic* merupakan gabungan dari pendekslsian tangan, *pose*, dan wajah yang pada pengaturan bawaanya terdapat 543 *landmark* yang terdiri dari 33 *landmark Pose*, 468 *landmark Face*, dan 21 *landmark* (per tangan). Gerakan dalam bahasa isyarat hanya fokus pada gerakan tangan dan tubuh bagian atas, oleh karena itu perlu adanya perubahan pengaturan pada jumlah *landmark* yang dideteksi. Untuk *landmark* tidak ada perubahan, sedangkan untuk *landmark pose* yang dipakai sebanyak 17 titik seperti pada Gambar 3.7.



**Gambar 3. 7 Mediapipe Holistic (Chen, Kuan-Yu, et al., 2022)**

Dapat dilihat perbandingan antara *MediaPipe Holistic* dengan pengaturan bawaan seperti pada gambar 3.8A dan setelah perubahan pengaturan seperti pada gambar 3.8B.



**Gambar 3. 8 Landmark Mediapipe Holistic (A), Landmark yang Dikonfigurasi (B)**  
(Sumber: Dokumen Pribadi)

Total *landmark* yang digunakan untuk ekstrak *keypoint* sebanyak 17 *landmark pose* + 21 *landmark* kiri + 21 *landmark* kanan = 59 *landmark*.

## 2. Extract Keypoint

Pada tahap ini dilakukan ekstraksi *keypoint* dari *landmark* dan *Pose* yang sudah dipersiapkan sebelumnya. Pada penelitian ini memakai sistem koordinat normalisasi yaitu titik-titik yang terdeteksi oleh *MediaPipe* biasanya dinyatakan dalam koordinat yang dinormalisasi, dengan rentang nilai antara 0 dan 1. Pada gambar atau video sumbu X menunjukkan posisi *horizontal* dari titik kunci, dengan nilai 0 berada di tepi kiri gambar dan nilai 1 di tepi kanan gambar, sumbu Y menunjukkan posisi vertikal dari titik kunci, dengan nilai 0 berada di tepi atas gambar dan nilai 1 di tepi bawah gambar. Meskipun koordinat Z tersedia untuk menunjukkan kedalaman relatif, dalam konteks ekstraksi titik kunci untuk bahasa isyarat, hanya menggunakan koordinat X dan Y untuk penyederhanaan. Proses ekstraksi *keypoint* dari *MediaPipe Holistic* melibatkan beberapa langkah utama, yang mencakup inisialisasi *MediaPipe Holistic*, pembacaan *frame*, deteksi *landmark*, dan konversi koordinat. *Pseudocode* dari proses ini dapat dilihat pada Algoritma 3.4.

### Algoritma 3. 4 Extract Keypoint

```
# Iterasi melalui folder isyarat
for direktori_isyarat in direktori_dataset:
    for file_frame in direktori_dataset:
        Buat path untuk menyimpan file numpy
```

```

Tentukan path file
Baca gambar dari path
Konversi frame ke RGB
Proses frame menggunakan MediaPipe Holistic
Cek apakah terdapat Landmark tangan
if results.pose_Landmarks:
    Gambar Landmark tangan
    Ekstrak keypoint
    Simpan keypoint dalam array

```

### 3. Data Augmentation

*Data augmentation* adalah teknik yang digunakan untuk memperbanyak jumlah data dengan cara memodifikasi data yang telah ada. Teknik ini berguna untuk meningkatkan generalisasi *model* dan menghindari *overfitting* pada saat melatih *model* pada beberapa salinan data yang telah dimodifikasi sedikit. Proses ini dilakukan setelah ekstraksi *keypoint* dengan memodifikasi koordinat asli. Pada penelitian ini menggunakan teknik augmentasi transformasi geometris, yaitu:

#### 1. Flip

*Flip horizontal* adalah transformasi yang membalik gambar di sepanjang sumbu vertikal untuk menambah variasi. Ini dapat direpresentasikan dengan rumus transformasi:

$$x' = W - x$$

Keterangan:

- $W$  adalah lebar gambar,
- $x$  adalah koordinat titik asli,
- $x'$  adalah koordinat titik setelah *flip*.

*Pseudocode* dari proses *flip* dapat dilihat pada Algoritma 3.5.

#### Algoritma 3.5 Augmentation Flip

```

BEGIN flip_image
    FUNCTION flip_image(image):
        # Step 1: Terapkan flip horizontal pada gambar
        flipped_image = cv2.flip(image, 1)
        return flipped_image
END

```

#### 2. Rotation

Rotasi adalah transformasi yang memutar gambar di sekitar titik tertentu, yang biasanya adalah pusat gambar. Setiap gambar dalam *dataset* akan diputar dengan sudut acak yang

diamambil dari interval 6 hingga 15 derajat dengan pusat rotasi di tengah gambar, rumus transformasinya adalah:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x - x_c \\ y - y_c \end{bmatrix} + \begin{bmatrix} x_c \\ y_c \end{bmatrix}$$

Keterangan:

- $\begin{bmatrix} x \\ y \end{bmatrix}$  adalah koordinat titik asli,
- $\begin{bmatrix} x' \\ y' \end{bmatrix}$  adalah koordinat titik setelah rotasi,
- $\begin{bmatrix} x_c \\ y_c \end{bmatrix}$  adalah koordinat pusat rotasi,
- $\theta$  adalah sudut rotasi yang dipilih secara acak dari interval  $[6^\circ, 15^\circ]$ .

*Pseudocode* dari proses *Rotation* dapat dilihat pada Algoritma 3.6.

#### Algoritma 3.6 Augmentation Rotation

```
BEGIN rotate_image
FUNCTION rotate_image(image):
    # Step 1: Tentukan sudut rotasi acak antara 6 dan 15 derajat
    angle = random.uniform(6, 15)
    # Step 2: Dapatkan ukuran gambar
    h, w = image.shape[:2]
    # Step 3: Tentukan pusat rotasi (tengah gambar)
    center = (w / 2, h / 2)
    # Step 4: Buat matriks rotasi menggunakan sudut yang telah ditentukan
    rotation_matrix = cv2.getRotationMatrix2D(center, angle, 1)
    # Step 5: Terapkan rotasi pada gambar
    rotated_image = cv2.warpAffine(image, rotation_matrix, (w, h))
    return rotated_image
END
```

### 3. Perspective Transform

*Perspective transform* adalah teknik di mana koordinat suatu gambar diproyeksikan ke dalam bidang baru berdasarkan pusat proyeksi yang dipilih secara acak dalam rentang antara 0 dan 1. Proses transformasi ini dapat dilakukan menggunakan fungsi *perspectiveTransform* dari pustaka *OpenCV*. Rumus Persamaan *Perspective Transform*: Jika  $(x,y)$  adalah koordinat awal dan  $(x',y')$  adalah koordinat setelah perspektif transformasi, maka transformasi ini dapat dinyatakan sebagai:

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Koordinat akhir diperoleh dengan normalisasi:

$$x' = \frac{x'}{w}$$

$$y' = \frac{y'}{w}$$

*Pseudocode* dari proses *perspective transform* dapat dilihat pada Algoritma 3.7.

#### **Algoritma 3.7 Augmentation Perspective Transform**

```
BEGIN perspective_transform
FUNCTION perspective_transform(image):
    # Step 1: Dapatkan ukuran gambar
    h, w = image.shape[:2]
    # Step 2: Tentukan titik-titik sumber (4 titik sudut gambar asli)
    src_points = np.float32([[0, 0], [w, 0], [0, h], [w, h]])
    # Step 3: Tentukan titik-titik tujuan dengan penambahan offset acak
    dst_points = src_points + np.random.uniform (-30, 30,
    src_points.shape).astype(np.float32)
    # Step 4: Buat matriks transformasi perspektif
    perspective_matrix = cv2.getPerspectiveTransform(src_points, dst_points)
    # Step 5: Terapkan transformasi perspektif pada gambar
    perspective_transformed_image = cv2.warpPerspective(image, _matrix, (w, h))
    return perspective_transformed_image
END
```

#### 4. Squeeze

*Squeeze* adalah transformasi yang mengurangi skala gambar pada sumbu *horizontal*. Jika  $s$  adalah faktor *squeeze* yang dipilih secara acak dari interval  $[0.05, 0.15]$ , maka rumus transformasinya adalah:

$$x' = x \cdot (1 - s)$$

$$y' = y$$

*Pseudocode* dari proses *squeeze* dapat dilihat pada Algoritma 3.8.

#### **Algoritma 3.8 Augmentation Squeeze**

```
BEGIN squeeze_image
FUNCTION squeeze_image(image):
    # Step 1: Tentukan faktor squeeze acak antara 5% dan 15%
    factor = 1 - random.uniform(0.05, 0.15)
    # Step 2: Dapatkan ukuran gambar
    h, w = image.shape[:2]
    # Step 3: Terapkan squeeze pada gambar
    squeezed_image = cv2.resize(image, (int(w * factor), h))
    return squeezed_image
END
```

#### 5. Stretch

*Stretch* adalah transformasi yang meningkatkan skala gambar pada sumbu *horizontal*. Jika  $s$  adalah faktor *stretch* yang dipilih secara acak dari interval  $[0.05, 0.15]$ , maka rumus transformasinya adalah:

$$x' = x \cdot (1 + s)$$

$$y' = y$$

*Pseudocode* dari proses *Stretch* dapat dilihat pada Algoritma 3.9.

#### Algoritma 3.9 Augmentation Stretch

```
BEGIN stretch_image
FUNCTION stretch_image(image):
    # Step 1: Tentukan faktor stretch acak antara 5% dan 15%
    factor = 1 + random.uniform(0.05, 0.15)
    # Step 2: Dapatkan ukuran gambar
    h, w = image.shape[:2]
    # Step 3: Terapkan stretch pada gambar
    stretch_image = cv2.resize(image, (int(w * factor), h))
    return stretch_image
END
```

#### 6. Arm Joint Rotation

*Arm joint rotation* adalah teknik di mana titik-titik kunci pada sendi lengan diputar dengan sudut acak antara 2 hingga 4 derajat. Peluang setiap sendi untuk berotasi adalah 3 banding 10.

Rumus Persamaan *Arm Joint Rotation*:

Untuk setiap sendi dengan koordinat  $(x, y)$ , jika dipilih untuk berotasi dengan sudut  $\theta$  (dipilih secara acak antara 2 hingga 4 derajat), maka koordinat baru  $(x', y')$  adalah:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Di mana  $\theta$  dipilih dengan peluang 3 banding 10 untuk setiap sendi.

*Pseudocode* dari proses *Arm Joint Rotation* dapat dilihat pada Algoritma 3.10.

#### Algoritma 3.10 Augmentation Rotate Arm Joints

```
BEGIN rotate_arm_joints
BEGIN rotate_arm_joints
FUNCTION rotate_arm_joints(image, Landmarks):
    # Step 1: Dapatkan ukuran gambar
    h, w = image.shape[:2]
    center_x, center_y = w / 2, h/2
    # Step 2: Iterasi melalui setiap sendi
```

```

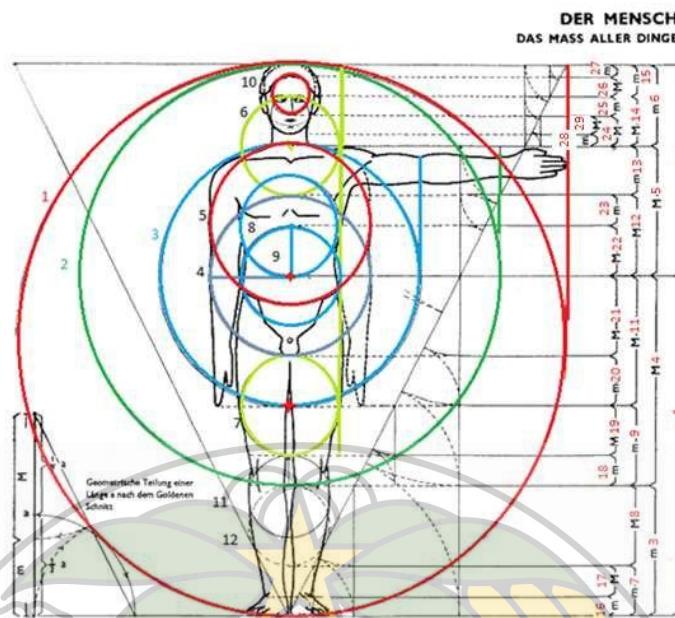
for joint in Landmarks:
    #Step 3: Tentukan apakah sendi akan dirotasi berdasarkan peluang 3 banding 10
    if random.random() < 0.3 maka
        # Step 4: Tentukan sudut rotasi acak antara 2 dan 4 derajat
        angle = random.uniform(2, 4)
        # Step 5: Dapatkan koordinat sendi
        x, y = joint.x, joint.y
        # Step 6: Buat matriks rotasi menggunakan sudut yang telah ditentukan
        rotation_matrix = cv2.getRotationMatrix2D((center_x, center_y), angle, 1)
        # Step 7: Terapkan rotasi pada koordinat sendi
        joint.x, joint.y = np.dot(rotation_matrix, np.array([x, y, 1]))[:2]
    return Landmarks
END

```

Setelah implementasi 6 proses augmentasi data diperoleh penambahan data sebanyak 12 kali lipat seperti pada Tabel 3.1.

#### 4. Bounding Box Normalization

*Bounding Box Normalization* merupakan langkah penting dalam *preprocessing* untuk deteksi *pose* manusia dan tangan. Tujuan utama dari proses ini adalah untuk menormalisasi ukuran dan posisi *bounding box* agar konsisten, memungkinkan *model machine learning* untuk mengenali dan menganalisis *pose* manusia dan tangan secara akurat, tanpa terpengaruh oleh variasi dalam skala dan posisi *input* gambar. Ukuran *bounding box* ditetapkan berdasarkan konsep proporsi tubuh ideal dari karya Leonardo da Vinci yaitu “*The Vitruvian Man*”, dalam konsep tersebut menyatakan bahwa tinggi manusia adalah delapan kepala panjangnya. Sehingga teknik normalisasi ini dapat diaplikasikan tanpa terpengaruh oleh variasi skala dan posisi *input* gambar. Ilustrasi dari proporsi tubuh manusia oleh Ernst Neufert berdasarkan “*The Vitruvian Man*” dapat dilihat pada Gambar 3.9.



Gambar 3. 9 Bauentwurfslehre oleh Ernst Neufert (1936)

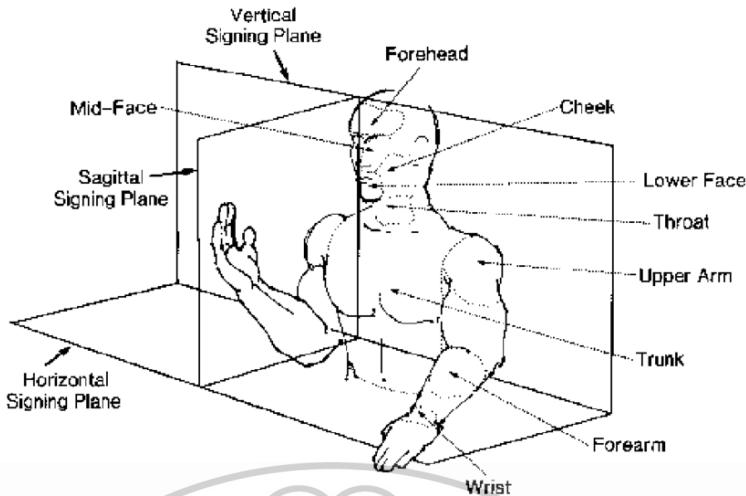
Menurut Abu-Taieh (2016) pada penelitiannya yang berjudul “*An Algorithm for Human Modeling in Information Technology Multimedia Using Human Biometrics Found in Golden Ratio, Vitruvian Man and Neufert*”, panjang kepala adalah 1/8 dari tinggi manusia, sedangkan lebar maksimal bahu adalah 1/4 dari tinggi manusia. Dapat disimpulkan bahwa kepala adalah 1/2 dari panjang bahu, maka ditemukan rumus persamaan sebagai berikut:

$$h = \frac{1}{2} (b_1 - b_0)$$

Keterangan:

- $h$  adalah tinggi kepala
- $b_1$  adalah koordinat bahu kanan
- $b_0$  adalah koordinat bahu kiri

*Bounding box* diaplikasikan masing-masing pada *landmark pose* dan *landmark*. Ukuran *bounding box* pada *landmark pose* sama dengan ukuran ruang isyarat, berdasarkan penelitian yang dilakukan oleh Miljan (1995) pada penelitiannya yang berjudul “*The noun phrase in Estonian Sign Language from the typological perspective*”, menyatakan bahwa ruang isyarat dibatasi oleh bagian atas kepala, bagian belakang, ruang yang meluas hingga lebar siku di sisi-sisinya, dan hingga pinggul; titik-titik berbeda pada tubuh berfungsi sebagai lokasi untuk konfigurasi tangan seperti Gambar 3.10.



**Gambar 3. 10** Ilustrasi ruang isyarat (Rodda & Grove, 1987)

Berdasarkan pernyataan tersebut maka ukuran *bounding box* pada *landmark pose* ditetapkan dengan lebar 6 tinggi kepala dengan hidung sebagai titik tengah atau 3 tinggi kepala disebelah kiri dan sebelah kanan hidung, dan tinggi dari *bounding box* yaitu 4,5 tinggi kepala dengan 0,5 di atas hidung dan 4 di bawah hidung. Sehingga didapatkan *pseudocode* untuk *bounding box normalization* pada *landmark pose* seperti pada Algoritma 3.11.

#### **Algoritma 3. 11 Pseudocode Pose Bounding Box Normalization**

```
function pose_box(Landmarks: list, width: int, height: int, head_unit: float) ->
list:
    nose_x = Landmarks[0][0] * width # Mengambil koordinat x hidung
    nose_y = Landmarks[0][1] * height # Mengambil koordinat y hidung

    # Menghitung batas kotak bounding box
    min_x = round(nose_x - head_unit * 3)
    max_x = round(nose_x + head_unit * 3)
    min_y = round(nose_y - head_unit / 2)
    max_y = round(nose_y + head_unit * 4)

    norm_pose = [] # Inisialisasi list untuk Landmark pose yang dinormalisasi
    for id, l in enumerate(Landmarks): # Iterasi semua Landmark
        if id in CONF_PL: #Hanya Landmark yang diinginkan yang akan diproses
            # Normalisasi koordinat x Landmark pose
            x = round((width * l[0] - min_x) / (max_x - min_x), 12)
            # Normalisasi koordinat y Landmark pose
            y = round((height * l[1] - min_y) / (max_y - min_y), 12)
            # Menambahkan koordinat Landmark yang sudah dinormalisasi ke list
            norm_pose.append([x, y])
```

```
    return norm_pose # Mengembalikan Landmark pose yang sudah dinormalisasi
```

Ukuran *bounding box* pada *landmark* ditetapkan dengan lebar dan tinggi 1,5 kali kepala. Sehingga didapatkan pseudocode dari *bounding box* pada *landmark* seperti Algoritma 3.12.

### Algoritma 3.12 Pseudocode Bounding Box Normalization

```
function _box(Landmarks: list, width: int, height: int, head_unit: float) ->
list:
    if sum(max(Landmarks)) != 0: # Cek jika Landmark tidak kosong
        x_max = y_max = 0
        x_min = width
        y_min = height

        # Iterasi semua Landmark untuk menemukan batas-batas bounding box
        for Landmark in Landmarks:
            x = int(Landmark[0] * width)
            y = int(Landmark[1] * height)
            if x > x_max:
                x_max = x
            if x < x_min:
                x_min = x
            if y > y_max:
                y_max = y
            if y < y_min:
                y_min = y

            dist = head_unit / 1.5
            x_mid = (x_min + x_max) / 2
            y_mid = (y_min + y_max) / 2

            min_x = round(x_mid - dist)
            max_x = round(x_mid + dist)

            min_y = round(y_mid - dist)
            max_y = round(y_mid + dist)

            norm_ = [] # Inisialisasi list untuk Landmark tangan yang
            dinormalisasi
            for l in Landmarks:
                # Normalisasi koordinat x Landmark tangan
                x = round((width * l[0] - min_x) / (max_x - min_x), 12)
                # Normalisasi koordinat y Landmark tangan
                y = round((height * l[1] - min_y) / (max_y - min_y), 12)
                # Menambahkan koordinat Landmark yang sudah dinormalisasi ke list
```

```

norm_.append([x, y])

return norm_ # Mengembalikan Landmark tangan yang sudah dinormalisasi
return Landmarks # Jika Landmark kosong, kembalikan Landmark asli

```

Pada gambar 3.11 dapat dilihat penerapan kedua fungsi *bounding box normalization* pada sebuah *frame*.



**Gambar 3. 11** Contoh *Bounding Box* pada salah satu *frame*  
(Sumber: Dokumen Pribadi)

##### 5. Save Keypoint

Penyimpanan *keypoint* dari ekstraksi menggunakan *MediaPipe* ke dalam *file NumPy* melibatkan langkah-langkah sederhana. Pertama, hasil ekstraksi *keypoint* dikonversi menjadi *array NumPy*. Kemudian, *array NumPy* ini disimpan dalam *file* menggunakan fungsi *np.save()*. Dengan demikian, *keypoint* yang diekstraksi dapat dengan mudah disimpan dan diakses kembali untuk penggunaan atau analisis lebih lanjut. Contoh *array* dari salah satu data dapat dilihat pada Gambar 3.12.

```
[0.49991738 0.10971641 0.52183447 0.06217413 0.53578362 0.06439476
 0.54828562 0.067603 0.47723078 0.06289648 0.46117016 0.06576965
 0.44778855 0.07039165 0.57163963 0.10128577 0.43110919 0.09928544
 0.52559435 0.16913262 0.4704837 0.16954835 0.66814759 0.40513618
 0.33488769 0.41465234 0.77263972 0.75271717 0.22854714 0.75124684
 0.72178797 0.64083838 0.30850449 0.64095806 0.89353971 0.75965838
 0.87596506 0.54625623 0.72970309 0.37545796 0.56064902 0.31468525
 0.44488043 0.2568787 0.61695708 0.44428852 0.38059068 0.34866683
 0.27168285 0.29937077 0.201217 0.25887341 0.56212294 0.56267535
 0.33258065 0.44853148 0.21896139 0.37612038 0.15011002 0.31529161
 0.53311523 0.66771573 0.31065611 0.55270901 0.20041527 0.47595646
 0.12764529 0.4109568 0.52317318 0.7580987 0.35872627 0.67366676
 0.26660383 0.6159942 0.19204464 0.56100117 0.16136754 0.81335489
 0.16896277 0.5747984 0.30364898 0.40536218 0.45467577 0.30200718
 0.55415574 0.20544793 0.45767166 0.46246695 0.6498413 0.32238249
 0.72526626 0.25241286 0.76632532 0.20659315 0.52473756 0.56859979
 0.7213425 0.41182359 0.80094228 0.3177932 0.8422741 0.25095445
 0.56269735 0.6634847 0.74708975 0.50478137 0.82361891 0.40950714
 0.86293129 0.34021999 0.576787 0.74082335 0.7260865 0.60847963
 0.78881033 0.52327913 0.82022838 0.45802992]
```

Gambar 3. 12 Contoh array setelah ekstrak *keypoint*

### 3.2.4 Data Pre-processing

Sebelum masuk ke tahap pelatihan *model* dilakukan *pre-processing* data yang didalamnya meliputi beberapa proses seperti *Data Padding*, *Data Labeling*, *Split Data*, dan *Create Features and Labels Array* dengan penjelasan sebagai berikut:

#### 1. Data Padding

*Data Padding* adalah proses menambahkan nilai atau elemen tertentu ke data untuk menyamakan panjangnya dengan data lainnya. Penulis melakukan *data padding* karena jumlah *frame* yang diekstrak dari setiap bahasa isyarat berbeda, sehingga dibutuhkan *data padding* agar data memiliki ukuran atau panjang data yang sama. Sebelum proses *data padding*, dilakukan pengecekan total maksimal *frame* dari *dataset*, *pseudocode* dari proses ini dapat dilihat pada Algoritma 3.13.

#### Algoritma 3. 13 Pseudocode Data Padding

```
max_val = 0

for setiap folder_isyarat in DATASET_FRAME:
    for setiap folder_frames in folder_isyarat:
        count = hitung jumlah file in folder_frames
        Jika count > max_val, max_val = count

Output max_val
```

Jumlah *frame* maksimal yang didapat adalah 26. Kemudian langkah *data padding* dimulai dengan menginisialisasi sebuah list yang berisi nilai 0.1 sebanyak jumlah *keypoint* yaitu 118 kali, kemudian list tersebut dijadikan sebagai *template* untuk dilakukan *padding* pada data. Selanjutnya, dilakukan iterasi untuk mengecek jumlah *frame* melalui struktur *dataset*, dimulai dari *folder* isyarat, kemudian ke *folder frame* di setiap isyarat, dan akhirnya ke setiap *frame* di dalam *folder* tersebut. Jika jumlah *frame* kurang dari nilai maksimum yang diinginkan (*max\_val*), maka kode tersebut melakukan *padding* dengan menambahkan nilai *pad\_list* sebanyak yang dibutuhkan untuk mencapai jumlah maksimum *frame* yang diinginkan. Nilai-nilai yang sudah *padding* kemudian disimpan dalam format *numpy array* dengan menggunakan *np.save()*. Pada Gambar 3.13 dapat dilihat contoh hasil *padding* data.

```
Array from: dataset_keypoints\Apa\BISINDO_Apa_001\10.npy
[0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1]
```

**Gambar 3. 13** Contoh *array* setelah *padding* data

*Pseudocode* dari proses *padding* data dapat dilihat pada Algoritma 3.14.

#### **Algoritma 3. 14** *Pseudocode Padding Data*

```
Inisialisasi pad_list dengan [0.1] * KEYPOINT_SHAPE
totalDir = hitung_data(DATASET_NP)
for setiap folder_isyarat in DATASET_NP:
    for setiap folder_frames in folder_isyarat:
        count = count_data(folder_frames)

        if count < max_val:
            for i dalam rentang(hitung, max_val):
                Simpan pad_list ke folder_frames[i]
```

#### 2. Data *Labeling*

Data *labeling* adalah proses memberikan *label* atau *tag* untuk mengidentifikasi dan mengategorikan data. *Labeling* pada penelitian ini dilakukan dengan mengambil nama *folder* dari masing-masing isyarat dan menetapkannya menjadi *label* masing-masing isyarat itu sendiri. Proses ini menggunakan fungsi *enumerate*. Kode dan hasil dari proses ini dapat dilihat pada Gambar 3.14.

```

labels = {folder:num for num, folder in enumerate(os.listdir(DATASET_NP))}
print(labels)

{'Apa': 0, 'Apa Kabar': 1, 'Bagaimana': 2, 'Baik': 3, 'Belajar': 4, 'Berapa': 5, 'Berdiri': 6,
'Bingung': 7, 'Dia': 8, 'Duduk': 9, 'Halo': 10, 'Kalian': 11, 'Kami': 12, 'Kamu': 13, 'Kapan': 14,
'Kemana': 15, 'Kenapa': 16, 'Kita': 17, 'Makan': 18, 'Mandi': 19, 'Marah': 20, 'Melihat': 21,
'Membaca': 22, 'Menulis': 23, 'Mereka': 24, 'Minum': 25, 'Pendek': 26, 'Ramah': 27, 'Sabar': 28,
'Saya': 29, 'Sedih': 30, 'Selamat Malam': 31, 'Selamat Pagi': 32, 'Selamat Siang': 33,
'Selamat Sore': 34, 'Senang': 35, 'Siapa': 36, 'Terima Kasih': 37, 'Tidur': 38, 'Tinggi': 39}

```

**Gambar 3. 14** Kode dan hasil dari proses *labelling* data

### 3. Split Data

Pada proses ini data diacak dan dibagi menjadi dua bagian dengan perbandingan: 70% data pelatihan, yang digunakan untuk melatih *model*, dan 30% data validasi, yang digunakan untuk menguji kinerja *model*. *Pseudocode* dari proses ini dapat dilihat pada Algoritma 3.15.

#### Algoritma 3. 15 Pseudocode Split Data

```

Definisikan DATASET_NP
Definisikan train_list dan val_list sebagai list kosong
Definisikan tr_rt dan vl_rt sebagai 70 dan 30

for setiap folder_isyarat in os.listdir(DATASET_NP):
    current_folder_path = gabungkan(DATASET_NP, folder_isyarat)
    vid_arr = os.listdir(current_folder_path)

    data_num = daftar indeks dari vid_arr
    partition = bagi data_num menjadi 50 bagian

    for setiap item in partition:
        Ambil 2 elemen dari awal dan 4 elemen dari akhir
        Berikan kepada f_h (first_half) dan s_h (second_half)
        Hapus f_h dan s_h dari partisi
        Berikan elemen yang tersisa kepada partition[id]

    a = daftar angka dari 0 hingga 49

    val_number = []
    chosen = a digabungkan dengan 40 angka yang dipilih secara acak dari a
    for setiap indeks in chosen:
        Pilih secara acak satu elemen dari f_h dan satu dari s_h dari partisi
        yang sesuai
        Tambahkan mereka ke val_number
        Hapus elemen-elemen ini dari partisi mereka masing-masing

    train_number = daftar indeks in data_num yang tidak ada di val_number
    Tambahkan path ke val_list dan train_list berdasarkan val_number dan

```

```

train_number secara berturut-turut

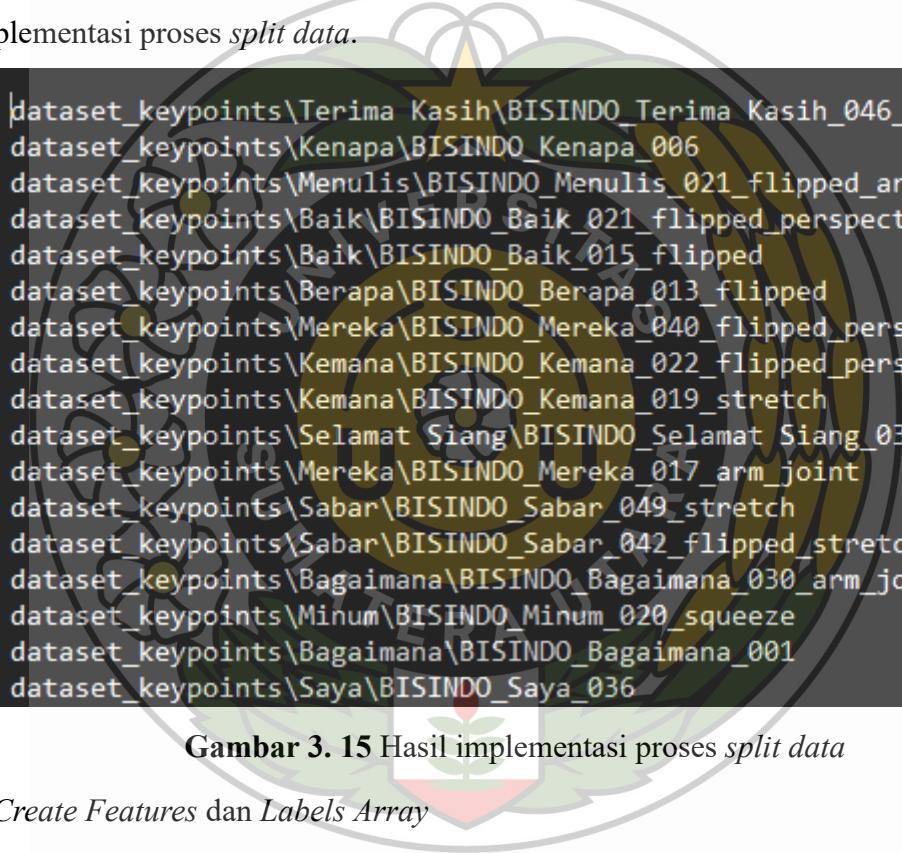
Buka file_train dan file_val for ditulis

Acak train_list dan val_list

for setiap path in train_list:
    Tulis path ke file_train
for setiap path in val_list:
    Tulis path ke file_val

```

Proses ini mengacak kemudian membagi path dari *file dataset* kemudian menyimpannya ke dalam *file* berekstensi .txt. Gambar 3.15 adalah contoh hasil implementasi proses *split data*.



```

dataset_keypoints\Terima Kasih\BISINDO_Terima_Kasih_046_rotation
dataset_keypoints\Kenapa\BISINDO_Kenapa_006
dataset_keypoints\Menulis\BISINDO_Menulis_021_flipped_arm_joint
dataset_keypoints\Baik\BISINDO_Baik_021_flipped_perspective
dataset_keypoints\Baik\BISINDO_Baik_015_flipped
dataset_keypoints\Berapa\BISINDO_Berapa_013_flipped
dataset_keypoints\Mereka\BISINDO_Mereka_040_flipped_perspective
dataset_keypoints\Kemana\BISINDO_Kemana_022_flipped_perspective
dataset_keypoints\Kemana\BISINDO_Kemana_019_stretch
dataset_keypoints\Selamat Siang\BISINDO_Selamat_Siang_032
dataset_keypoints\Mereka\BISINDO_Mereka_017_arm_joint
dataset_keypoints\Sabar\BISINDO_Sabar_049_stretch
dataset_keypoints\Sabar\BISINDO_Sabar_042_flipped_stretch
dataset_keypoints\Bagaimana\BISINDO_Bagaimana_030_arm_joint
dataset_keypoints\Minum\BISINDO_Minum_020_squeeze
dataset_keypoints\Bagaimana\BISINDO_Bagaimana_001
dataset_keypoints\Saya\BISINDO_Saya_036

```

**Gambar 3. 15** Hasil implementasi proses *split data*

#### 4. Create Features dan Labels Array

Pada tahap ini dilakukan pembuatan *features* dan *label* pada masing-masing data latih dan data validasi, maka dihasilkan sepasang *array features* dan *labels* yaitu *training features*, *training labels*, *validation features*, dan *validation labels*. *Features* akan digunakan sebagai *input* pada saat pelatihan *model* sedangkan *labels* merupakan *output* target yang sesuai untuk data latih. *Pseudocode* dari pembuatan *features* dan *label array* dapat dilihat pada Algoritma 3.16.

#### Algoritma 3. 16 Pseudocode Create Features dan Labels Array

```

Inisialisasi list kosong: features, labels
Inisialisasi count ke 0

```

```

Buka file 'txt' dan baca setiap barisnya
for setiap baris in 'txt':
    Hapus spasi dari baris dan simpan in variabel path
    Inisialisasi list kosong: keypoints
    Tambah count dengan 1

    for setiap file (sequence) in direktori yang ditentukan oleh path:
        Muat file .npy dan tambahkan ke keypoints

        Tambahkan keypoints ke features
        Ekstrak label dari path dan tambahkan ke labels

        Cetak count (progres) pada baris yang sama

Simpan labels
Simpan features

```

*Shape* pada array *features* berupa tiga dimensi dengan dimensi pertama adalah jumlah baris data, dimensi kedua adalah jumlah *sequence* atau *frame*, dan dimensi ketiga adalah jumlah *keypoint* yang dapat dideteksi. *Shape* yang didapatkan pada *features training* data yaitu (16800,26,118), sedangkan pada *validation* data (7200,26,118). Dapat dilihat pada Gambar 3.16 array dari *feature training* data.

```

[[[0.49904696 0.11269788 0.52680557 ... 0.64354035 0.62431777 0.71224202]
 [0.49918077 0.11284533 0.52565197 ... 0.61160722 0.68377727 0.67960409]
 [0.50116196 0.10945496 0.52786756 ... 0.55508521 0.77873465 0.60554476]
 ...
 [0.49931612 0.11258288 0.52587213 ... 0.41600491 0.85472815 0.44115551]
 [0.49994193 0.11135401 0.52627046 ... 0.44886186 0.8281975 0.48631784]
 [0.50033288 0.11061658 0.52669896 ... 0.4472411 0.82873011 0.48289396]]

[[0.49966517 0.11267998 0.52080966 ... 0.52804127 0.55739754 0.55617527]
 [0.49960853 0.10894167 0.52086124 ... 0.52724162 0.58472571 0.47570815]
 [0.50033181 0.11126967 0.52006991 ... 0.60020218 0.4975465 0.5245657 ]
 ...
 [0.50013444 0.11090592 0.52020696 ... 0.39177691 0.55920447 0.30507905]
 [0.49998924 0.11125977 0.52005682 ... 0.52723101 0.56235929 0.45845629]
 [0.49997902 0.11126744 0.52002671 ... 0.59792504 0.45551716 0.51954269]]

[[0.49912711 0.11187068 0.51957648 ... 0.45648268 0.84591725 0.40891287]
 [0.50072052 0.11133099 0.52092741 ... 0.436635 0.80245614 0.37495174]
 [0.1 0.1 0.1 ... 0.1 0.1 0.1 ]
 ...
 [0.50004739 0.1129911 0.52474731 ... 0.52863536 0.81303781 0.48666096]
 [0.1 0.1 0.1 ... 0.1 0.1 0.1 ]
 [0.1 0.1 0.1 ... 0.1 0.1 0.1 ]]]

```

Gambar 3. 16 Contoh array *features*

*Shape* pada array *labels* berupa satu dimensi yang dimana dimensi satu tersebut adalah jumlah *label* pada *dataset*, (16800) pada *training* data dan (7200) pada *validation* data. Pada Gambar 3.17 dapat dilihat contoh array dari *labels*.

[37 16 23 3 3 5 24 15 15 33 24 28 28 2 25 2 29 35 7 17 4 38 16 7
38 8 3 6 14 39 4 38 31 21 22 29 1 0 2 20 26 37 4 21 25 39 34 36
9 29 20 14 34 29 25 20 31 23 3 14 24 17 3 32 13 32 32 5 35 2 15 9
6 18 36 23 22 28 8 18 38 31 37 4 37 11 3 24 30 17 12 25 37 1 31 20

Gambar 3.17 Contoh array *labels*

Pada *model* pembelajaran mesin, terutama dalam konteks klasifikasi, mengharapkan *output* yang berupa vektor *one-hot* atau biner yaitu 1 dan 0. *One-hot encoding* adalah teknik yang umum digunakan dalam pemrosesan data untuk mengubah variabel kategori menjadi representasi biner. Dalam jaringan saraf tiruan (*neural networks*), lapisan *output* terakhir biasanya memiliki neuron untuk setiap kelas yang mungkin. Vektor *one-hot* memungkinkan *model* untuk secara langsung membandingkan *output* prediksi dengan *label* yang sebenarnya. Dengan menggunakan vektor *one-hot*, setiap elemen dalam vektor mewakili keanggotaan sampel dalam kelas tertentu. Ini membuat interpretasi hasil prediksi lebih jelas. Sehingga dilakukan *one-hot encoding* pada *label* yang merupakan *output* target dari pelatihan *model* pembelajaran mesin. Proses ini memakai fungsi *to\_categorical* dari Tensorflow Keras. Sehingga contoh array dari *label* yang didapatkan setelah proses *one-hot encoding* dapat dilihat pada Gambar 3.18.

[[[0. 0. 0. ... 0. 0. 0.]
[0. 0. 1. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
...
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]]

Gambar 3.18 Contoh array *labels* setelah proses *one-hot encoding*

### 3.2.5 Training

Dalam penelitian ini, penulis menggunakan arsitektur *Temporal Convolutional Network* (TCN) untuk memodelkan data *sequential*. Arsitektur TCN ini terdiri dari beberapa lapisan yang bertujuan untuk mengekstraksi fitur-fitur penting dari data. Pertama-tama, data *input* di-mask dengan menggunakan lapisan *Masking* untuk mengabaikan nilai *padding*. Kemudian, dilakukan serangkaian konvolusi satu dimensi (Conv1D) dengan *kernel* yang berbeda dan *padding* '*causal*' untuk mempertahankan urutan temporal. Konvolusi ini dilakukan dengan aktivasi fungsi tangen hiperbolik (tanh) untuk

memperkenalkan non-linearitas. Penggunaan *dilation rate* yang berbeda pada lapisan-lapisan konvolusi memungkinkan *model* untuk memperluas jangkauan temporalnya secara efektif. Setelah itu, dilakukan *Global Average Pooling* untuk meratakan hasil konvolusi menjadi vektor fitur tunggal. Kemudian, vektor fitur tersebut disalurkan melalui lapisan-lapisan *Dense* yang terdiri dari unit-unit tersembunyi dengan aktivasi ReLU, diikuti oleh lapisan *Dropout* untuk mengurangi *overfitting*. Pada akhirnya, lapisan *output* menggunakan fungsi aktivasi softmax untuk menghasilkan probabilitas *output* dari kelas-kelas yang mungkin.

Selama proses pelatihan, pengoptimalan dilakukan menggunakan algoritma Adam. Algoritma optimasi Adam dipilih karena keunggulannya dalam menangani masalah-masalah pada algoritma optimasi *gradient stochastic*, seperti *learning rate* yang adaptif, penyimpanan momentum dari iterasi sebelumnya, dan adanya perhitungan eksponensial dari gradien kedua. Hal ini membuat Adam cocok untuk pelatihan *model* dengan data *sequential* seperti yang digunakan dalam penelitian ini, karena kemampuannya untuk menyesuaikan laju pembelajaran secara dinamis terhadap perubahan dalam gradien dan untuk menavigasi ruang parameter yang mungkin memiliki banyak minimum lokal. Selain itu, pemilihan fungsi *loss categorical crossentropy* didasarkan pada sifat data yang memiliki banyak *label* kelas. Fungsi loss ini merupakan pilihan yang tepat untuk tugas klasifikasi multi-kelas, di mana setiap sampel data dapat terkait dengan satu atau lebih kelas. *Categorical crossentropy* membantu *model* dalam mengevaluasi seberapa baik distribusi probabilitas *output* yang dihasilkan oleh *model* sesuai dengan distribusi probabilitas yang diharapkan dari *label* sebenarnya. Selain itu, penggunaan *TensorBoard*, *EarlyStopping*, dan *ModelCheckpoint* sebagai *callback* memungkinkan pemantauan pelatihan melalui log, menghentikan pelatihan jika tidak terjadi peningkatan dalam kinerja validasi selama beberapa *epoch* berturut-turut, dan menyimpan *model* dengan bobot terbaik berdasarkan kinerja validasi. Proses pelatihan dilakukan dengan membagi data menjadi *batch-batch* kecil, dengan validasi dilakukan menggunakan data terpisah. Dengan demikian, *model* TCN ini akan disesuaikan dengan data pelatihan untuk mengoptimalkan klasifikasi kelas-kelas yang diinginkan. Hasil akhir dari *model* yang dilatih akan disimpan dengan format '.h5' untuk penggunaan dan evaluasi lebih lanjut. *Hyperparameter* yang digunakan dijabarkan pada tabel.

**Tabel 3. 3** Hyperparameter Tuning pada TCN

Konfigurasi	Jenis	Unit	Parameter	Parameter Conf.	
Conv1D	Activation	64	tanh	<i>dilation_rate=1</i>	
				<i>dilation_rate=2</i>	
		128		<i>dilation_rate=4</i>	
				<i>dilation_rate=8</i>	
<i>GlobalAveragePooling1D</i>					
<i>Dense Layer</i>	Activation	256	Relu		
		128			
<i>Dropout</i>		0.4			
Konfigurasi Model	Optimizer		Adam		
	<i>learning Rate</i>		0.001		
Konfigurasi Training	Batch Size	8			
		16			
		32			

### 3.2.6 Testing

Testing merupakan tahap penting dalam pengembangan *model machine learning* untuk memastikan bahwa *model* berfungsi dengan baik dan memenuhi spesifikasi yang telah ditentukan. Tahap ini dilakukan secara langsung kepada pengguna. Tahapan testing dalam skripsi ini mencakup beberapa proses, yaitu *Input*, *Load Model*, *Pre-processing*, *Prediction*, dan *Output*. Berikut adalah penjelasan rinci dari masing-masing proses:

#### 1. *Input*

Tahap ini bertujuan untuk mengumpulkan data masukan yang akan digunakan oleh *model* untuk melakukan prediksi. Data masukan berupa *frame* yang diambil secara langsung menggunakan kamera *smartphone* yang dihubungkan ke komputer. Proses *input* dalam skripsi ini dilakukan dengan menggunakan fungsi cv2.VideoCapture() untuk mengakses kamera dan mengambil *frame* secara *real-time*. Setiap *frame* akan diproses lebih lanjut pada tahap *pre-processing*.

## 2. Load Model

Pada tahap ini, *model machine learning* yang telah dilatih sebelumnya akan dimuat ke dalam program untuk digunakan dalam prediksi. Proses pemuatan *model* dilakukan dengan menggunakan fungsi *load\_model()* dari *library TensorFlow Keras*. *Model* yang dimuat adalah *model* yang telah dilatih untuk mengenali isyarat BISINDO (Bahasa Isyarat Indonesia).

## 3. Pre-processing

Tahap *pre-processing* dilakukan untuk mempersiapkan data masukan yang telah diambil dari kamera agar sesuai dengan format yang diperlukan oleh *model* untuk melakukan prediksi. Tahapan *pre-processing* dalam skripsi ini meliputi:

- *Resize frame*: Setiap *frame* yang diambil dari video diubah ukurannya menjadi 400x300 piksel. Proses ini dilakukan agar ukuran *frame* konsisten dengan data yang digunakan saat pelatihan *model*.
- *Frame Selection*: Pemilihan *frame* yang relevan dari video, yang mencakup proses pemilihan *keyframe* dan pengurangan *frame* untuk mengurangi jumlah *frame* yang diproses tanpa menghilangkan informasi penting.
- *Extract Keypoint*: Ekstraksi titik-titik kunci (*keypoints*) dari setiap *frame* menggunakan *MediaPipe Holistic* yang sudah dikonfigurasi, yang mencakup titik-titik pada *pose*, tangan kiri, dan tangan kanan.
- *Normalize Keypoint*: Normalisasi titik-titik kunci yang diekstraksi agar sesuai dengan format *input* yang diperlukan oleh *model*.

## 4. Prediction

Pada tahap ini, *model* melakukan prediksi dengan menerjemahkan isyarat BISINDO menjadi bahasa Indonesia. Titik-titik kunci yang telah dinormalisasi dari tahap *pre-processing* digunakan sebagai *input* untuk *model* yang telah dimuat. *Model* akan memproses *input* tersebut dan menghasilkan *output* berupa *label* yang merupakan hasil prediksi dari isyarat yang ditunjukkan dalam video. Proses ini melibatkan pengenalan pola dari titik-titik kunci yang telah diekstraksi dan dinormalisasi.

## 5. Output

*Output* yang dihasilkan oleh *model* berupa teks dan suara terjemahan dalam bahasa Indonesia. Teks yang dihasilkan akan ditampilkan pada layar, dan suara akan dihasilkan menggunakan teknologi *text-to-speech* untuk memberikan umpan balik suara.

Proses *output* dalam skripsi ini mencakup:

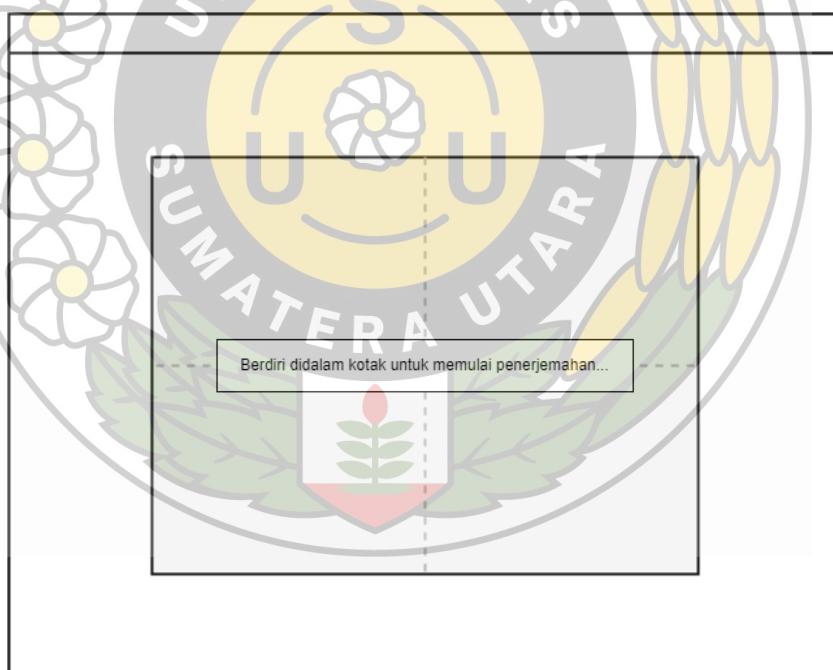
- Teks: Teks hasil prediksi ditampilkan pada layar menggunakan fungsi cv2.putText().
- Suara: Menghasilkan suara dari teks hasil prediksi menggunakan *library* gTTS (Google *Text-to-Speech*) dan memutar suara tersebut menggunakan *playsound*.

### 3.3 Perancangan Antarmuka Sistem

Tahap ini dilakukan dalam pengembangan sistem yang bertujuan untuk merancang bagaimana pengguna akan berinteraksi dengan sistem tersebut. Ada beberapa tampilan yang dapat ditampilkan seperti tampilan utama, tampilan *loading* penerjemahan, dan tampilan hasil terjemahan.

#### 3.3.1 Tampilan Utama

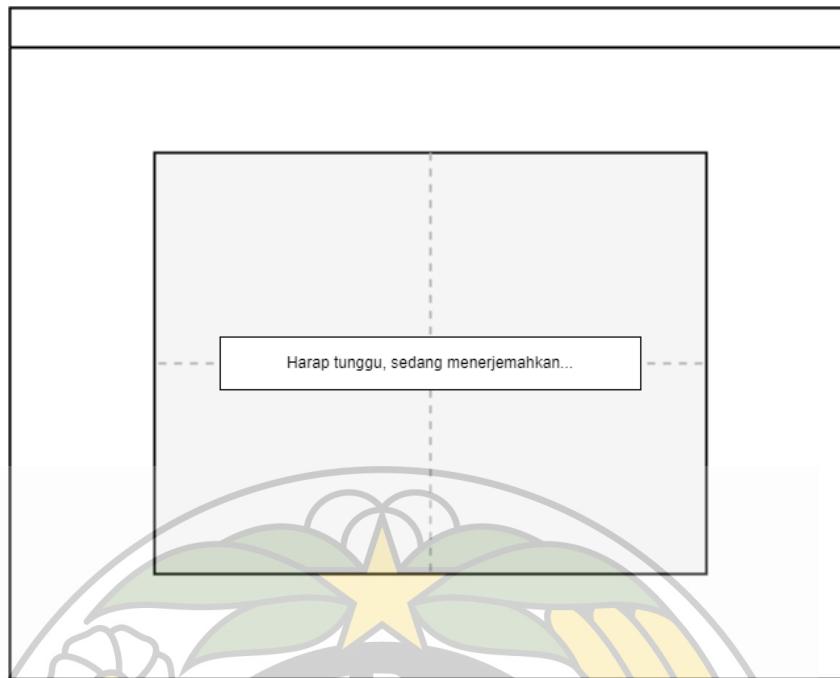
Saat aplikasi dijalankan tampilan yang pertama kali muncul adalah window yang menampilkan *input* dari kamera dengan *overlay* kotak pembatas yang berfungsi sebagai penanda area pengguna untuk melakukan isyarat. Kerangka tampilan utama dapat dilihat pada Gambar 3.19.



**Gambar 3. 19** Tampilan Utama

#### 3.3.2 Tampilan Loading Penerjemahan

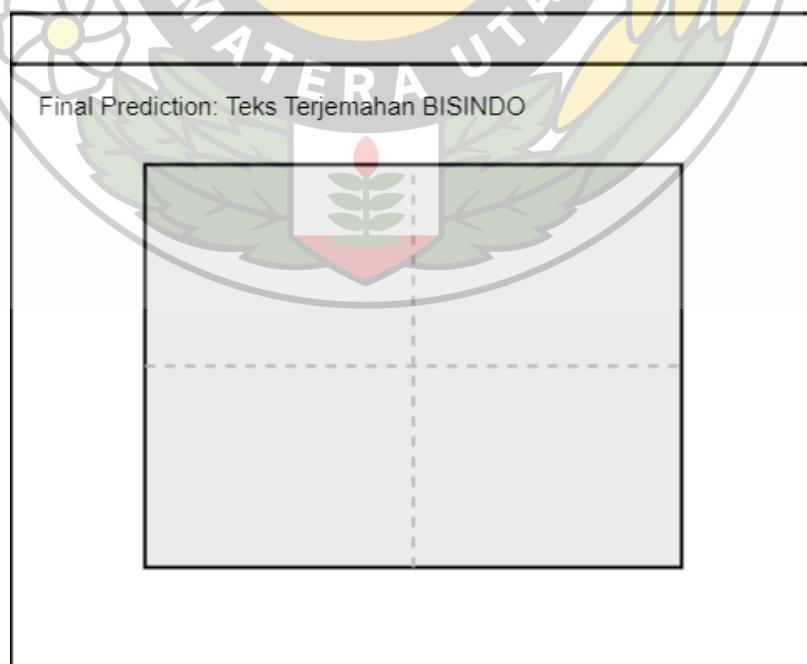
Tampilan *loading* penerjemahan muncul ketika program sudah selesai merekam gerakan isyarat yang ditandai dengan keluarnya tangan dari *frame* kamera dan memulai prediksi menggunakan *model*. Kerangka tampilan *loading* penerjemahan dapat dilihat pada Gambar 3.20.



Gambar 3. 20 Tampilan Loading Penerjemahan

### 3.3.3 Tampilan Hasil Terjemahan

Pada tampilan ini akan ditampilkan hasil prediksi dari *model* yaitu kata terjemahan isyarat BISINDO. Kerangka tampilan dari tampilan hasil terjemahan dapat dilihat pada Gambar 3.21.



Gambar 3. 21 Tampilan Hasil Penerjemahan

## **BAB 4**

### **IMPLEMENTASI DAN PENGUJIAN SISTEM**

#### **4.1 Implementasi Sistem**

Berdasarkan penjelasan perancangan sistem yang telah diuraikan pada Bab 3, langkah selanjutnya adalah mengimplementasikan sistem untuk sistem pengenalan Bahasa Isyarat Indonesia (BISINDO) menggunakan *MediaPipe* dan algoritma *Temporal Convolutional Networks* (TCN) yang dimana perancangan sistem ini menggunakan perangkat-perangkat sebagai berikut:

##### *4.1.1 Spesifikasi Hardware dan Software*

Perancangan sistem ini menggunakan spesifikasi *hardware* seperti pada Tabel 3.4.

**Tabel 3. 4 Spesifikasi Hardware**

Nama Perangkat	Spesifikasi
Lenovo IdeaPad C340-14IML	Prosesor: Intel Core i3-8145U CPU 2.10GHz
	GPU: Nvidia GeForce MX230 2 GB
	RAM: 8 GB
	Memori: 512 GB SSD
POCO X3 Pro	Kamera: 48 MP
	RAM: 6 GB
	Memori: 128 GB

##### *4.1.2 Spesifikasi Perangkat Lunak*

Spesifikasi *software* dapat dilihat pada Tabel 3.5.

**Tabel 3. 5 Spesifikasi Software**

Nama Perangkat	Spesifikasi
Lenovo IdeaPad C340-14IML	OS: Windows 11 Home Single Language 64-bit

	Python 3.8.10
	Visual Studio Code 1.90.0
	Adobe Premiere Pro 2024
	Jupyter Notebook 7.0.7
	DroidCam 6.5.2
POCO X3 Pro	Android 12
	DroidCam 6.27

#### 4.1.3 Implementasi Perancangan Tampilan Antarmuka

Pada tahap ini desain antarmuka diimplementasikan berdasarkan rancangan tampilan yang telah dijelaskan pada Bab 3 sebelumnya. Implementasi desain antarmuka ini melibatkan langkah-langkah berikut:

##### 1. Tampilan Utama

Tampilan yang pertama kali dilihat oleh pengguna setelah menjalankan aplikasi adalah *window* yang menampilkan *input* dari kamera dengan *overlay* kotak pembatas yang berfungsi sebagai penanda area pengguna untuk melakukan isyarat. Saat pengguna belum berada pada posisi yang sesuai yaitu didalam kotak pembatas maka *border* dari kotak pembatas akan berwarna merah dan didalamnya terdapat tulisan yaitu ‘Berdiri didalam kotak untuk memulai penerjemahan...’. Ketika posisi pengguna belum sesuai program tidak akan merekam *frame*. Setelah posisi pengguna sesuai, *border* kotak pembatas akan berubah menjadi warna hijau dan program bisa memulai untuk merekam *frame*, *frame* akan mulai direkam saat tangan dari pengguna masuk ke dalam kotak pembatas dan *frame* berhenti direkam saat tangan keluar dari kotak pembatas. Tampilan utama dapat dilihat pada Gambar 3.19. Gambar A adalah tampilan ketika posisi pengguna belum sesuai dan Gambar B adalah tampilan ketika posisi pengguna sesuai.

##### 2. Tampilan *Loading* Penerjemahan

Setelah selesai berisyarat program akan melakukan proses *preprocessing* hingga prediksi berdasarkan *frame* yang telah direkam, pada saat proses ini akan muncul tampilan *loading* penerjemahan yang didalamnya terdapat tulisan ‘Harap menunggu, sedang menerjemahkan...’. Tampilan *loading* penerjemahan dapat dilihat pada Gambar 3.21.

### 3. Tampilan Hasil Terjemahan

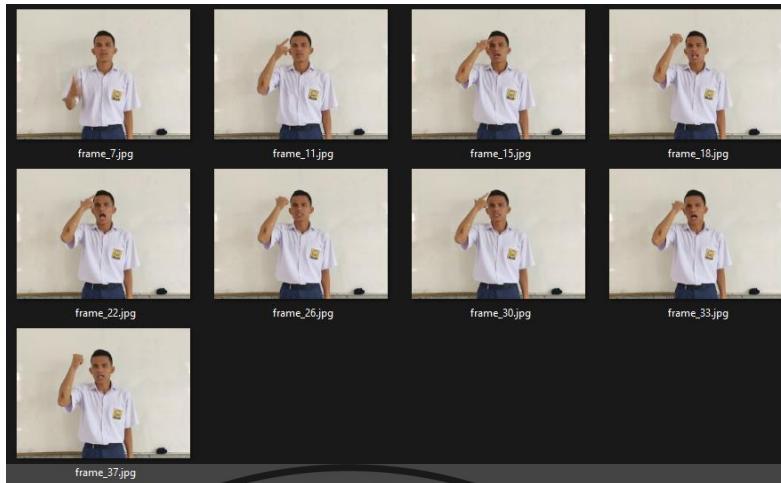
Setelah proses prediksi selesai, muncul tampilan hasil terjemahan yaitu *overlay box* teks dari terjemahan isyarat BISINDO. Tampilan hasil terjemahan dapat dilihat pada Gambar 3.21.

#### 4.1.4 Implementasi Data

Implementasi data mencakup beberapa tahapan penting mulai dari pengumpulan data, *preprocessing*, hingga pembagian data untuk pelatihan dan validasi. Data yang digunakan adalah video isyarat BISINDO seperti pada Gambar 4.1 yang direkam menggunakan perangkat telefon cerdas POCO X3 Pro dengan kamera beresolusi 48MP. Data ini berupa 40 isyarat BISINDO berjumlah 2000 video yang dikelompok menjadi 6 kategori seperti: 1) Kata ganti orang: 'Dia', 'Kalian', 'Kami', 'Kamu', 'Kita', 'Mereka', 'Saya'. 2) Kata tanya: 'Apa', 'Bagaimana', 'Berapa', 'Kapan', 'Kemana', 'Kenapa', 'Siapa'. 3) Kata kerja: 'Bantu', 'Belajar', 'Berjalan', 'Duduk', 'Makan', 'Mandi', 'Melihat', 'Menulis', 'Minum', 'Tidur'. 4) Kata sifat: 'Baik', 'Bingung', 'Marah', 'Pendek', 'Ramah', 'Sabar', 'Sedih', 'Senang', 'Tinggi'. 5) Kata sapaan: 'Halo', 'Selamat Malam', 'Selamat Pagi', 'Selamat Siang', 'Selamat Sore'. 6) Kata Ungkapan: 'Maaf', 'Terima Kasih'. Selanjutnya data ini melewati tahap *preprocessing* meliputi beberapa sub-tahap seperti ekstraksi *frame* yang disimpan dengan ekstensi *file jpg* seperti pada Gambar 4.2, pemilihan *keyframe*, dan pengurangan *frame*. Data juga diaugmentasi menghasilkan jumlah data sebanyak 24000 video. Setiap *frame* kemudian diekstrak fiturnya menggunakan *MediaPipe* untuk mendeteksi *pose manusia* dan tangan dan disimpan ke dalam *file Numpy*.



**Gambar 4. 1** Contoh dataset video isyarat BISINDO



**Gambar 4. 2** Contoh *dataset frame* isyarat BISINDO

#### *4.1.5 Pelatihan Sistem*

Pada tahap ini dilakukan *hyperparameter tuning* menggunakan *Keras Tuner* untuk mencari *hyperparameter* terbaik untuk membangun, mengoptimalkan, dan melatih model TCN (*Temporal Convolutional Network*). Parameter yang diuji adalah jumlah *layer*, *filter unit*, *activation*, *dilation rate*, *dropout rate*, *dense unit*, dan *batch size*. Pengujian dilakukan sebanyak 30 kali dengan masing-masing percobaan memiliki maksimal 10 *epoch*. Hasil parameter terbaik dan detail tiap percobaan dapat dilihat pada Tabel 4.1 dan Tabel 4.2.d

**Tabel 4. 1** Hyperparameter Tuning

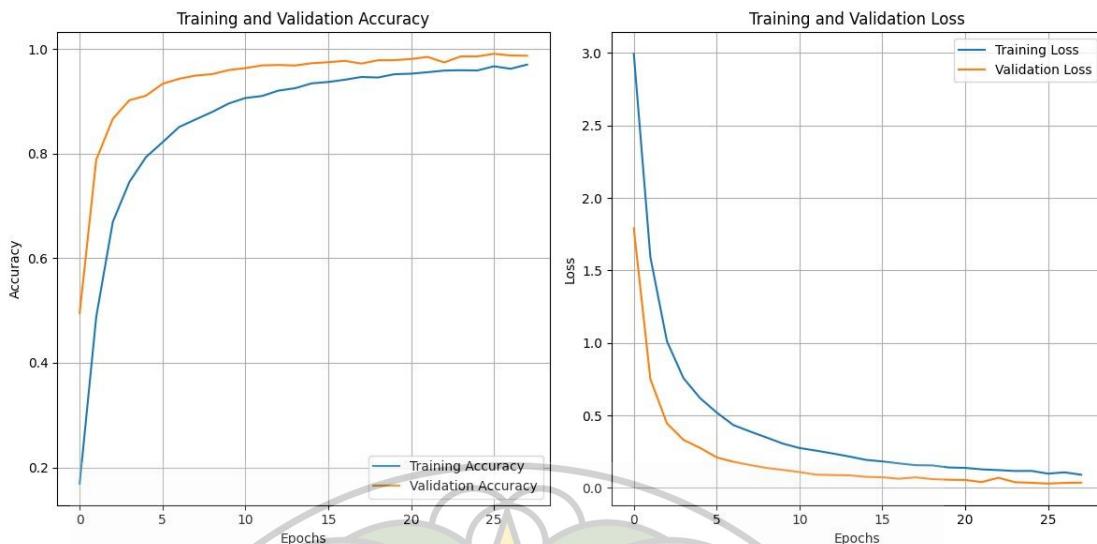
Parameter	Nilai Minimal	Nilai Maksimal	Hasil Terbaik		
Jumlah Layer	1	3	3		
	Layer TCN			Layer 1	Layer 2
Filter Unit	32	256	64	64	128
Activation	relu tanh		relu	relu	relu
Dilation Rate	1	8	1	4	8
Dropout Rate	0.2	0.5	0.4		
Dense Units	128	512	256 & 128		
Batch Size	32	128	32		

**Tabel 4.2** Hasil Percobaan *Hyperparameter Tuning*

Jumlah Layer	filter 1	aktivasi 1	dilation rate 1	filter 2	aktivasi 2	dilation rate 2	filter 3	aktivasi 3	dilation rate 3	Dense Unit	Drop out	Val Accuracy
--------------	----------	------------	-----------------	----------	------------	-----------------	----------	------------	-----------------	------------	----------	--------------

3	64	tanh	1	64	relu	4	128	tanh	8	256	0.4	97.89%
3	32	tanh	4	512	relu	1	256	tanh	2	256	0.4	97.87%
3	128	tanh	1	64	relu	2	64	tanh	8	256	0.2	97.78%
3	32	tanh	2	64	tanh	1	512	tanh	8	256	0.2	97.53%
2	128	relu	8	256	tanh	4				512	0.2	97.44%
3	64	tanh	4	128	tanh	1	128	relu	4	128	0.3	97.39%
3	64	relu	8	256	tanh	1	64	relu	4	256	0.2	96.76%
2	128	relu	1	32	relu	4				512	0.2	96.75%
2	512	tanh	1	32	tanh	4			8	512	0.3	96.10%
2	128	tanh	8	32	relu	1				256	0.3	96.01%
2	64	relu	8	64	tanh	8				128	0.2	95.47%
1	512	relu	8							256	0.2	95.44%
3	64	tanh	1	32	relu	8	64	tanh	1	128	0.3	95.25%
1	256	tanh	8							512	0.2	95.03%
1	256	tanh	1							128	0.2	94.97%
1	128	relu	2							512	0.4	94.76%
1	512	relu	2							256	0.3	94.69%
1	64	tanh	8							512	0.4	94.49%
3	256	relu	2	32	relu	1	32	relu	1	128	0.3	94.22%
1	512	relu	4							128	0.2	94.14%
1	64	tanh	8							256	0.3	94.06%
1	64	tanh	4							128	0.3	93.93%
2	32	tanh	2	32	tanh	8				256	0.4	93.76%
2	512	relu	1	32	relu	4				128	0.4	93.69%
1	128	tanh	8							128	0.3	93.62%
3	128	relu	4	128	tanh	1	32	tanh	2	512	0.3	93.54%
1	64	relu	4							512	0.4	93.43%
1	32	relu	8							512	0.2	92.51%
1	64	relu	2							128	0.2	92.14%
1	32	tanh	8							128	0.4	91.67%

Setelah dilakukan *hyperparameter tuning* didapatkan akurasi tertinggi pada kombinasi neuron (64,64,128) dengan dilasi 1, 4, 8 dengan *dropout* 0.4, sehingga kombinasi tersebut yang dipakai pada *model*. Grafik proses pelatihan dapat dilihat pada Gambar 4.3.



**Gambar 4.3** Grafik akurasi dan *loss* dari *train* dan *validation*

Gambar 4.3 menunjukkan kurva akurasi dan loss selama proses pelatihan dan validasi model Temporal Convolutional Network (TCN) dalam tugas klasifikasi urutan gerakan tangan. Grafik sebelah kiri menggambarkan akurasi pelatihan dan validasi, sedangkan grafik sebelah kanan menunjukkan nilai loss selama pelatihan dan validasi dalam setiap epoch. Pada grafik akurasi, terlihat bahwa model mengalami peningkatan performa seiring bertambahnya epoch. Akurasi pelatihan meningkat secara bertahap, dimulai dari sekitar 20% pada epoch pertama dan mencapai hampir 99% pada epoch terakhir. Akurasi validasi juga menunjukkan pola peningkatan yang mirip, meskipun sedikit lebih tinggi pada awal pelatihan dan kemudian stabil di angka sekitar 97-98% setelah 25 epoch. Perbedaan kecil antara akurasi pelatihan dan validasi menunjukkan bahwa model mampu melakukan generalisasi dengan baik terhadap data yang tidak digunakan dalam pelatihan. Pada grafik loss, terlihat bahwa training loss dan validation loss mengalami penurunan signifikan seiring bertambahnya epoch. Loss pelatihan dimulai dari angka sekitar 3.0 dan mengalami penurunan tajam hingga mendekati 0.1, yang menunjukkan bahwa model semakin baik dalam meminimalkan kesalahan selama pelatihan. Hal serupa terjadi pada loss validasi, yang dimulai dari angka lebih rendah dibandingkan loss pelatihan, kemudian menurun hingga nilai mendekati 0.05. Perbedaan kecil antara training loss dan validation loss mengindikasikan bahwa model tidak mengalami overfitting, yang berarti model tetap bekerja dengan baik saat diuji menggunakan data yang tidak pernah dilihat sebelumnya. Secara keseluruhan, kurva yang dihasilkan menunjukkan bahwa model berhasil mencapai konvergensi dengan

baik. Peningkatan akurasi dan penurunan loss yang stabil menegaskan bahwa penggunaan arsitektur TCN dengan dilated convolutions dan dropout layers efektif dalam menangkap pola gerakan tangan serta menghindari masalah overfitting.

#### 4.2 Pengujian Sistem

Pada tahap ini dilakukan pengujian terhadap *model* TCN (*Temporal Convolutional Network*) yang sudah dilatih. Pengujian dilakukan secara langsung menggunakan *laptop* Lenovo IdeaPad C340-14IML yang dihubungkan dengan *smartphone* POCO X3 PRO sebagai *input* kamera menggunakan aplikasi DroidCam. Pengujian dilakukan sebanyak 20 kali pada masing-masing 40 kata BISINDO seperti pada Tabel 3.1 sesuai dengan *dataset* pada penelitian ini. Hasil pengujian dapat dilihat pada Tabel 4.3.

**Tabel 4. 3** Hasil Pengujian Sistem (Indikator Kata)

No.	Frame	Arti Isyarat	Output	Keterangan
1		Apa	Bagaimana	Salah
2		Bagaimana	Bagaimana	Benar
3		Baik	Baik	Benar
4		Bantu	Bantu	Benar
5		Belajar	Minum	Salah
6		Berapa	Kemana	Salah

7		Berjalan	Berjalan	Benar
8		Bingung	Bingung	Benar
9		Dia	Dia	Benar
10		Duduk	Duduk	Benar
11		Halo	Halo	Benar
12		Kalian	Kalian	Benar
13		Kami	Saya	Salah
14		Kamu	Kamu	Benar
15		Kapan	Kapan	Benar
16		Kemana	Kemana	Benar

17		Kenapa	Kenapa	Benar
18		Kita	Kita	Benar
19		Maaf	Halo	Salah
20		Makan	Makan	Benar
21		Mandi	Minum	Salah
22		Marah	Marah	Benar
23		Melihat	Melihat	Benar
24		Menulis	Menulis	Benar
25		Mereka	Mereka	Benar
26		Minum	Minum	Benar

27			Pendek	Tinggi	Salah
28			Ramah	Ramah	Benar
29			Sabar	Saya	Salah
30			Saya	Saya	Benar
31			Sedih	Sedih	Benar
32			Selamat Malam	Selamat Sore	Salah
33			Selamat Pagi	Selamat Malam	Salah
34			Selamat Siang	Selamat Malam	Salah
35			Selamat Sore	Selamat Pagi	Salah
36			Senang	Senang	Benar

37		Siapa	Siapa	Benar
38		Terima kasih	Terima kasih	Benar
39		Tidur	Tidur	Benar
40		Tinggi	Tinggi	Benar

Penulis juga melakukan pengujian *MediaPipe* terhadap jarak kamera, dengan jarak yang diuji yaitu 50 cm, 100 cm, dan 150 cm. Hasil pengujian sistem untuk berbagai jarak ini dapat dilihat pada Tabel 4.4.

**Tabel 4.4** Hasil Pengujian Sistem (Indikator Jarak)

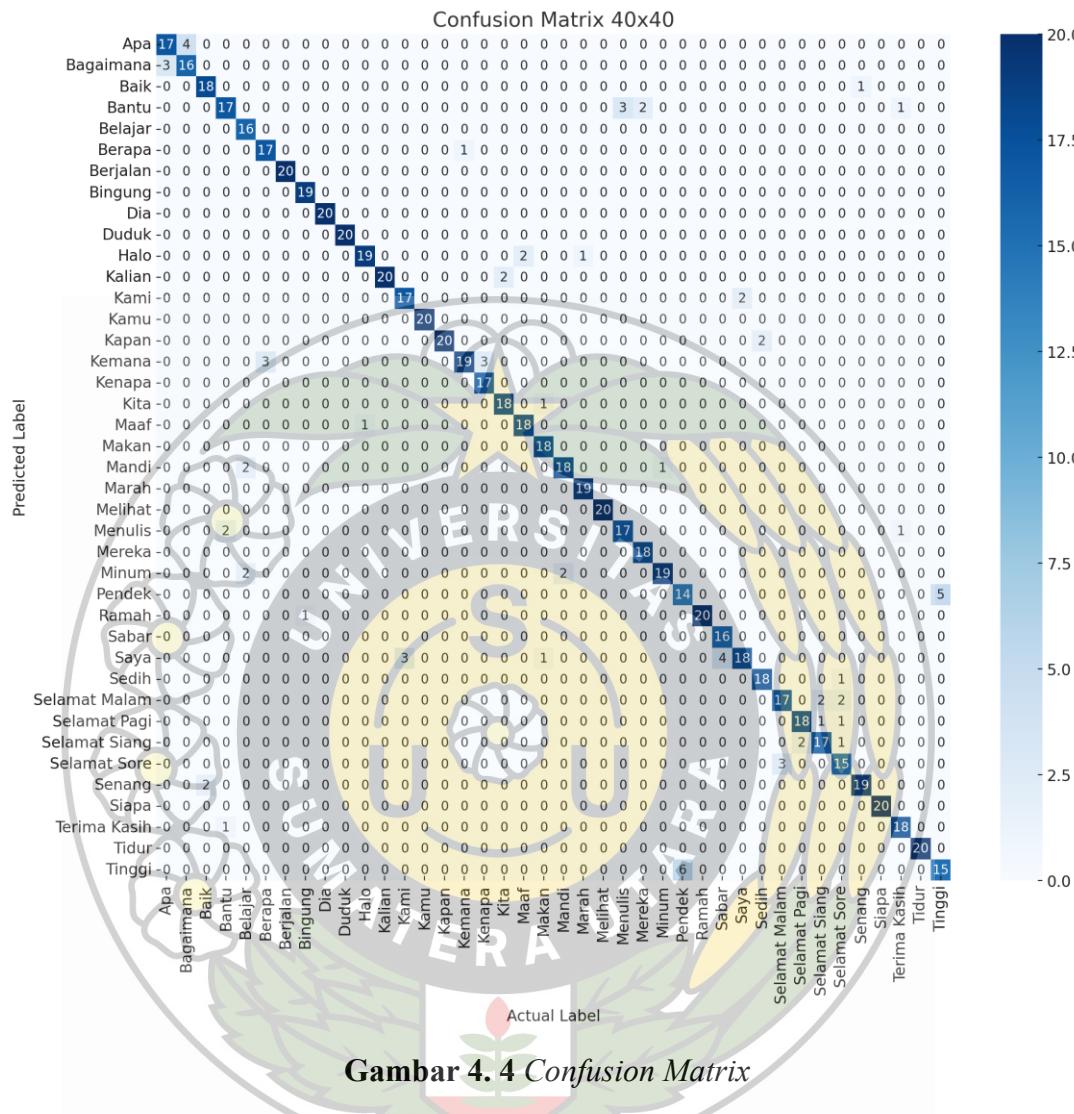
No.	Jarak	Frame	Keterangan
1	50cm		<i>MediaPipe</i> mampu melakukan deteksi dengan akurat.
2	100cm		<i>MediaPipe</i> mampu melakukan deteksi dengan akurat.
3	150cm		<i>MediaPipe</i> tidak dapat mendeteksi secara akurat.

*MediaPipe* mampu mendeteksi pada semua jarak yang diuji, tetapi performanya mulai menurun pada jarak 150 cm. Pada jarak ini, deteksi menjadi tidak konsisten dan sering kali salah dalam memprediksi *landmark*, yang dapat memengaruhi proses seleksi *frame* dan hasil klasifikasi. Oleh karena itu, dapat disimpulkan bahwa jarak optimal untuk deteksi terbaik oleh *model* adalah antara 50 cm hingga 100 cm.

### 4.3 Evaluasi Sistem

Pada tahap ini, dilakukan evaluasi kinerja sistem dengan menggunakan *confusion matrix*. *Confusion matrix* adalah alat yang membantu mengukur performa *model*

klasifikasi dengan menampilkan perbandingan antara prediksi yang dilakukan oleh sistem dan hasil sebenarnya. Hasil pengujian sistem dapat dilihat dari *confusion matrix* seperti pada Gambar 4.4.



#### Gambar 4.4 Confusion Matrix

Tabel 4.5 memuat nilai True Positive (TP), False Positive (FP), dan False Negative (FN) dari setiap label data serta jumlah keseluruhan dari tiap nilai. Tabel ini merupakan hasil pengujian pada model.

**Tabel 4. 5** TP, FN, FP

<b><i>Label</i></b>	<b>TP</b>	<b>FN</b>	<b>FP</b>	<b>Total (TP+FN)</b>
Apa	17	3	4	20
Bagaimana	16	4	3	20
Baik	18	2	1	20
Bantu	17	3	6	20

Belajar	16	4	0	20
Berapa	17	3	1	20
Berjalan	20	0	0	20
Bingung	19	1	0	20
Dia	20	0	0	20
Duduk	20	0	0	20
Halo	19	1	3	20
Kalian	20	0	2	20
Kami	17	3	2	20
Kamu	20	0	0	20
Kapan	20	0	2	20
Kemana	19	1	6	20
Kenapa	17	3	0	20
Kita	18	2	1	20
Maaf	18	2	1	20
Makan	18	2	0	20
Mandi	18	2	3	20
Marah	19	1	0	20
Melihat	20	0	0	20
Menulis	17	3	3	20
Mereka	18	2	0	20
Minum	19	1	4	20
Pendek	14	6	5	20
Ramah	20	0	1	20
Sabar	16	4	0	20
Saya	18	2	8	20
Sedih	18	2	1	20
Selamat Malam	17	3	4	20
Selamat Pagi	18	2	2	20
Selamat Siang	17	3	3	20
Selamat Sore	15	5	3	20
Senang	19	1	2	20

Siapa	20	0	0	20
Terima Kasih	18	2	1	20
Tidur	20	0	0	20
Tinggi	15	5	6	20
Total	722	78	78	800

Dalam pengujian model yang dilakukan menggunakan 800 data uji, sebanyak 722 data berhasil diprediksi dengan benar, sementara 66 data tidak dapat diprediksi dengan tepat. Tabel 4.6 memperlihatkan nilai *precision*, *recall*, dan *f1-score* untuk pengujian setiap label data, serta rata-rata keseluruhan nilainya. *Precision* mengukur seberapa banyak prediksi positif yang benar dibandingkan dengan semua prediksi positif, berguna untuk mengetahui akurasi prediksi yang dilakukan oleh *model* terhadap data yang dianggap relevan. *Recall*, di sisi lain, mengukur sejauh mana *model* dapat mendeteksi data yang relevan dari keseluruhan data sebenarnya yang relevan. *F1-score* adalah rata-rata harmonis dari *precision* dan *recall*, yang memberikan keseimbangan antara keduanya, terutama jika terdapat ketidakseimbangan dalam data. Perhitungan *precision*, *recall*, dan *f1-score* dilakukan untuk mengevaluasi performa model secara lebih mendalam, bukan hanya berdasarkan akurasi. *Precision* relevan ketika kesalahan positif palsu harus dihindari, sedangkan *recall* lebih penting jika mendeteksi sebanyak mungkin data relevan adalah prioritas. Dalam penelitian ini, ketiga metrik ini membantu memahami sejauh mana *model* dapat mengenali dan memprediksi gerakan isyarat secara akurat, serta mengidentifikasi area yang perlu ditingkatkan, seperti penurunan *recall* pada label tertentu. Hubungan antara *precision*, *recall*, dan *f1-score* memberikan gambaran yang holistik tentang kemampuan *model* dalam menangani data uji.

**Tabel 4. 6** Nilai *precision*, *recall*, dan *f1-score* dari masing-masing *label*

<b><i>Label</i></b>	<b><i>Precision</i></b>	<b><i>Recall</i></b>	<b><i>F1-Score</i></b>
Apa	0.81	0.85	0.83
Bagaimana	0.84	0.80	0.82
Baik	0.95	0.90	0.92
Bantu	0.74	0.85	0.79
Belajar	1.00	0.80	0.89
Berapa	0.94	0.85	0.89
Bingung	1.00	1.00	1.00

Dia	1.00	0.95	0.97
Duduk	1.00	1.00	1.00
Halo	1.00	1.00	1.00
Kalian	0.86	0.95	0.90
Kami	0.91	1.00	0.95
Kamu	0.89	0.85	0.87
Kapan	1.00	1.00	1.00
Kemana	0.91	1.00	0.95
Kenapa	0.76	0.95	0.84
Kita	1.00	0.85	0.92
Maaf	0.95	0.90	0.92
Makan	0.95	0.90	0.92
Mandi	1.00	0.90	0.95
Marah	0.86	0.90	0.88
Melihat	1.00	0.95	0.97
Membaca	1.00	1.00	1.00
Menulis	0.85	0.85	0.85
Mereka	1.00	0.90	0.95
Minum	0.83	0.95	0.88
Pendek	0.74	0.70	0.72
Ramah	0.95	1.00	0.98
Sabar	1.00	0.80	0.89
Saya	0.69	0.90	0.78
Sedih	0.95	0.90	0.92
Selamat Malam	0.81	0.85	0.83
Selamat Pagi	0.90	0.90	0.90
Selamat Siang	0.85	0.85	0.85
Selamat Sore	0.83	0.75	0.79
Senang	0.90	0.95	0.93
Siapa	1.00	1.00	1.00
Terima Kasih	0.95	0.90	0.92
Tidur	1.00	1.00	1.00

Tinggi	0.71	0.75	0.73
Rata-rata	0.81	0.85	0.83

Untuk menghitung nilai akurasi dapat dihitung dengan persamaan berikut:

$$\text{Accuracy} = \frac{\text{TP (Jumlah data uji benar)}}{\text{TP + FP + FN (Total data uji)}} \times 100$$

$$\text{Accuracy} = \frac{722}{800} \times 100$$

$$\text{Accuracy} = 90.25\%$$

Berdasarkan persamaan di atas didapatkan persentase akurasi sebesar 90.25%. Ini menunjukkan bahwa *model* yang telah dilatih dapat menerjemahkan isyarat BISINDO dengan baik.



## **BAB 5**

### **KESIMPULAN DAN SARAN**

#### **5.1 Kesimpulan**

Adapun kesimpulan yang diperoleh dari hasil penelitian dan pengujian algoritma *Temporal Convolutional Network* (TCN) untuk pengenalan isyarat Bahasa Isyarat Indonesia (BISINDO) adalah sebagai berikut:

1. Sistem mampu mengenali gerakan bahasa isyarat dengan akurasi sebesar 90.25%, dengan memprediksi 722 dari 800 data uji dengan benar.
2. Terjadi kesalahan klasifikasi pada beberapa isyarat dengan kemiripan gerakan tinggi, terutama pada kelas Apa-Bagaimana, Kenapa-Maaf, dan Selamat Malam-Selamat Pagi.
3. Mayoritas kelas memiliki nilai evaluasi yang tinggi, dengan beberapa kelas mencapai F1-score mendekati 1.00, sedangkan kelas Pendek dan Tinggi memiliki nilai yang lebih rendah.
4. MediaPipe mampu mendeteksi isyarat dengan akurat pada jarak 50 cm hingga 100 cm, tetapi performa deteksi menurun pada jarak 150 cm.
5. Algoritma TCN mampu menangani data sekuensial dengan baik, mengenali pola gerakan tangan dalam bahasa isyarat dengan akurasi tinggi.

#### **5.2 Saran**

Berdasarkan hasil penelitian ini, ada beberapa saran yang dapat diberikan berdasarkan kendala yang dialami oleh penulis:

1. Penggunaan *dataset* yang lebih besar dan beragam: Untuk meningkatkan akurasi dan generalisasi *model*, disarankan untuk menggunakan *dataset* yang lebih besar dan lebih beragam, termasuk variasi dalam gerakan tangan, kecepatan gerakan, dan kondisi pencahayaan.
2. Integrasi dengan teknologi *real-time*: Pengembangan sistem yang dapat bekerja secara *real-time* akan sangat bermanfaat. Integrasi dengan perangkat keras seperti kamera yang lebih canggih dapat membantu meningkatkan performa sistem dalam kondisi nyata.

3. Kolaborasi dengan ahli bahasa isyarat: Bekerjasama dengan ahli bahasa isyarat dan komunitas tuli-bisu untuk memastikan bahwa sistem yang dikembangkan benar-benar sesuai dengan kebutuhan pengguna dan akurat dalam mengenali serta menerjemahkan bahasa isyarat.
4. Penggunaan metode yang berbeda dari penelitian ini agar didapatkan perbandingan dari penelitian yang telah dilakukan.



## DAFTAR PUSTAKA

- Abu-Taieh, E. (2016). An algorithm for human modeling in information technology multimedia using human biometrics found in golden ratio, vitruvian man and neufert. Proceedings - 2015 5th International Conference on e-Learning, ECONF 2015, 65–73. <https://doi.org/10.1109/ECONF.2015.43>
- Auziqni, R. (2022). BISINDO Indonesia Sign Language Recognition Using MediaPipe Holistic and LSTM Deep Learning Model [Universitas Mercu Buana]. <http://digilib.mercubuana.ac.id/>
- Bantupalli, K., & Xie, Y. (2018). American Sign Language Recognition using Deep Learning and Computer Vision. 2018 IEEE International Conference on Big Data (Big Data), 4896–4899.
- Bahri, S., & Rachmat, R. (2018). *Transformasi citra biner menggunakan metode thresholding dan Otsu thresholding*. Jurnal Sistem Informasi dan Teknologi Informasi, 7(2), 196–201.
- Bora, J., Dehingia, S., Boruah, A., Chetia, A. A., & Gogoi, D. (2023). Real-time Assamese Sign Language Recognition using MediaPipe and Deep Learning. Procedia Computer Science, 218, 1384–1393. <https://doi.org/10.1016/j.procs.2023.01.117>
- Chen, K. Y., Shin, J., Hasan, M. A. M., Liaw, J. J., Yuichi, O., & Tomioka, Y. (2022). Fitness movement types and completeness detection using a transfer-learning-based deep neural network. *Sensors*, 22(15), 5700.
- Goldberg, D. E., & Holland, J. H. (1988). Genetic Algorithms and Machine Learning. *Machine Learning* 3, 95–98.
- Gonzalez, R. C., & Woods, R. E. (2002). Digital image processing. Prentice Hall.
- Halder, A., & Tayade, A. (2021). Real-time Vernacular Sign Language Recognition using MediaPipe and Machine Learning. International Journal of Research Publication and Reviews, 2(5), 9–17. [www.ijrpr.com](http://www.ijrpr.com)
- Hossain, M. S., Molla, M. K. I., & Islam, M. S. (2024). *Deep Convolutional Neural Network-Based System for Fish Classification*. International Journal of Artificial Intelligence and Applications (IJAIA), 15(1), 89–100.
- Ishaq, M., Khan, M., & Kwon, S. (2023). TC-Net: A Modest & Lightweight Emotion Recognition System Using Temporal Convolution Network. Computer Systems Science and Engineering, 46(3), 3355–3369. <https://doi.org/10.32604/csse.2023.037373>
- Jiao, Y., & Du, P. (2016). Performance measures in evaluating machine learning based bioinformatics predictors for classifications. Quantitative Biology, 4(4), 320–330. <https://doi.org/10.1007/s40484-016-0081-2>
- Lugaresi, C., Tang, J., Nash, H., Mcclanahan, C., Ubweja, E., Hays, M., Zhang, F., Chang, C.-L., Yong, M. G., Lee, J., Chang, W.-T., Hua, W., Georg, M., Grundmann, M., & Research, G. (2019). MediaPipe: A Framework for Perceiving and Processing Reality. <https://github.com/google/mediapipe>.
- Miljan, M. (1995). The noun phrase in Estonian Sign Language from the typological perspective. <https://doi.org/10.13140/RG.2.2.20408.57604>
- Palfreyman, N. (2015). Budaya Tuli Indonesia dan Hak Bahasa (Indonesian Deaf Culture and Language Rights). <https://www.researchgate.net/publication/322818553>
- Pramudiya, R., Asyraq, C., Kadafi, A., & Sardika, R. P. (2024). Analisis gambar menggunakan metode grayscale dan HSV (Hue, Saturation, Value). Just IT: Jurnal Sistem Informasi, Teknologi Informatika dan Komputer, 14(3), 175–180.

- Pratamasunu, G. Q. O., Fajri, F. N., & Sari, P. K. (2022). Deteksi Tangan Otomatis Pada Video Percakapan Bahasa Isyarat Indonesia Menggunakan Metode Deep Gated Recurrent Unit (GRU). *Jurnal Komputer Terapan*, 8(1), 186–193. <https://jurnal.pcr.ac.id/index.php/jkt/>
- Putra, D. K. H., Kushartanthy, & Aris, S. (2013). Pendekripsi Tepi Citra Digital dengan Logika Fuzzy. *Jurnal Masyarakat Informatika*, 4(8), 11–20.
- Putri, H. M., Fadlisyah, & Fuadi, W. (2022). PENDEKSIAN BAHASA ISYARAT INDONESIA SECARA REAL-TIME MENGGUNAKAN LONG SHORT-TERM MEMORY (LSTM). *Jurnal Teknologi Terapan and Sains 4.0*, 3(1), 663–675.
- Rodda, M. & C. Grove. (1987). Language, Cognition, and Deafness. Hillsdale, New Jersey: Lawrence Erlbaum Associates.
- Saraswati, D. A., Towidjojo, V. D., & Hasanuddin. (2022). Bahasa Isyarat Indonesia. *Jurnal Medical Profession (MedPro)*, 4(1), 8–14.
- Scherer, M., Magno, M., Erb, J., Mayer, P., Eggimann, M., & Benini, L. (2020). TinyRadarNN: Combining Spatial and Temporal Convolutional Neural Networks for Embedded Gesture Recognition with Short Range Radars. *IEEE Internet of Things Journal*, 8(13), 10336–10346. <https://doi.org/10.1109/JIOT.2021.3067382>
- Singh, A. K., & Koundal, D. (2024). A Temporal Convolutional Network for modeling raw 3D sequences and air-writing recognition. *Decision Analytics Journal*, 10.
- Situmorang, Z., & Gunawan, H. (2020). PERANCANGAN APLIKASI KAMUS BAHASA ISYARAT MENGGUNAKAN ALGORITMA LEVENSHTEIN SEBAGAI PENCARIAN KATA BERBASIS ANDROID. *IT Journal*, 8(1), 1–11.
- Suyudi, I., Sudadio, S., Suherman, S., Gadjah, U., Daerah, M., Yogyakarta, I., & Artikel, R. (2022). Pengenalan Bahasa Isyarat Indonesia menggunakan Mediapipe dengan Model Random Forest dan Multinomial Logistic Regression (Introduction to Indonesian Sign Language Using Mediapipe With Random Forest Models and Multinomial Logistic Regression). *Jurnal Ilmu Siber Dan Teknologi Digital (JISTED)*, 1(1), 65–80. <https://doi.org/10.35912/jisted.v1i1.1899>
- Safrizal, M., & Harjoko, A. (2013). *Perbandingan pewarnaan citra grayscale menggunakan metode K-Means clustering dan agglomerative hierarchical clustering*. Berkala MIPA, 23(3), 255–262.
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2015). Rethinking the Inception Architecture for Computer Vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2818–2826. <http://arxiv.org/abs/1512.00567>
- Tsinganos, P., Jansen, B., Cornelis, J., & Skodras, A. (2022). Real-Time Analysis of Gesture Recognition with Temporal Convolutional Networks. *Sensors*, 22(5), 1694. <https://doi.org/10.3390/s22051694>
- Verschae, R., & Ruiz-del-Solar, J. (2015). Object detection: Current and future directions. *Frontiers Robotics AI*, 2. <https://doi.org/10.3389/frobt.2015.00029>
- Zhang, Y., Wang, C., Zheng, Y., Zhao, J., Li, Y., & Xie, X. (2019). Short-Term Temporal Convolutional Networks for Dynamic Gesture Recognition. <http://arxiv.org/abs/2001.05833>