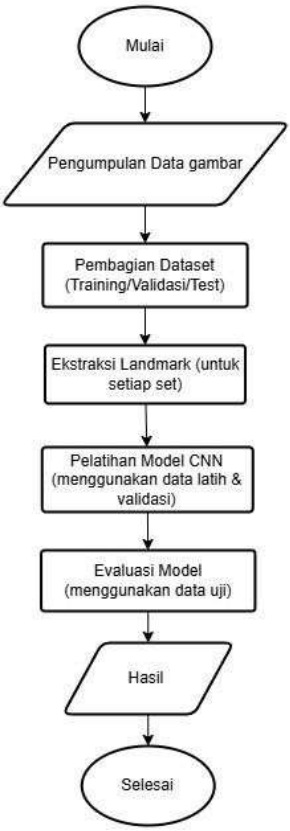
BAB III
METODOLOGI PENELITIAN

3.1 Desain Penelitian

Penelitian ini menggunakan pendekatan eksperimental untuk merancang sistem pengenalan bahasa isyarat American Sign Language (ASL) berbasis teknologi *computer vision* dan framework MediaPipe. Sistem ini difokuskan pada pengenalan gerakan tangan statis dan klasifikasinya menggunakan model *Convolutional Neural Network* (CNN). Metode eksperimen dipilih agar peneliti dapat mengendalikan variabel yang terlibat serta mengevaluasi pengaruhnya terhadap kinerja sistem. Tahapan penelitia terdiri dari lima langkah utama: pengumpulan data, pra-pemrosesan, pelatihan model, implementasi sistem, dan evaluasi performa.



Gambar 3.1 Diagram Alir Pengembangan American Sign Language

Framework MediaPipe digunakan sebagai alat utama untuk pelacakan landmark tangan karena keunggulannya dalam efisiensi dan akurasi pelacakan gerakan secara real-time. MediaPipe memungkinkan pengambilan data gerakan tangan dalam bentuk koordinat landmark yang kemudian digunakan sebagai input untuk algoritma machine learning. Pada tahap klasifikasi, algoritma deep learning seperti CNN (Convolutional Neural

Network) dipilih karena kemampuannya dalam mengenali pola kompleks pada data visual.

Sistem ini dirancang untuk bekerja dengan perangkat keras yang sederhana seperti laptop dengan webcam, menjadikannya solusi yang praktis dan mudah diakses. Studi ini juga mengkaji berbagai elemen yang bisa berdampak pada performa aplikasi, seperti pencahayaan, latar belakang, dan variasi gerakan pengguna.

3.2 Langkah-Langkah Penelitian

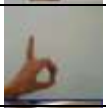
3.2.1 Pengumpulan Dataset

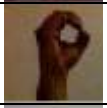
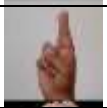
Tahapan pertama dalam penelitian adalah mengumpulkan dataset yang berkaitan dengan gerakan tangan dalam bahasa isyarat ASL (American Sign Language). Dataset ini dirancang untuk mencakup 34 kelas gerakan tangan, yang meliputi angka serta huruf kecuali huruf J dan huruf Z.

Dataset diambil dari data yang direkam secara langsung menggunakan webcam, dataset yang sudah ada yaitu kaggle (sebuah komunitas Data Science yang dimiliki oleh Google). Proses persiapan data sangat penting karena tanpa dataset yang sesuai, proses pembuatan model tidak akan bisa dilakukan dengan baik. Secara umum di hamper semua kasus, proses persiapan data bisa menghabiskan lebih dari separuh waktu dan energi seorang *Data Scientist*. Beberapa orang bahkan menyebutkan bahwa sebagaimana dilaporkan oleh praktisi data, persiapan data dapat menghabiskan lebih dari 70% waktu.

Berikut adalah beberapa gambar yang diambil dari Kaggle:

Tabel 3.1 Gestur Tangan Huruf dari Kaggle

No.	Gambar	Arti	Jumlah
1		A	86
2		B	146
3		C	148
4		D	146
5		E	145
6		F	146
7		G	146
8		H	147
9		I	147
10		K	146
11		L	148

No.	Gambar	Arti	Jumlah
13		N	147
14		O	147
15		P	145
16		Q	147
17		R	146
18		S	147
19		T	147
20		U	147
21		V	147
22		W	145
23		X	148

No.	Gambar	Arti	Jumlah
12		M	147

No.	Gambar	Arti	Jumlah
24		Y	138

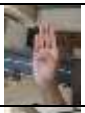
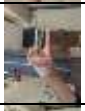
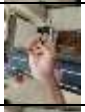
Tabel 3.2 Gestur Angka dari Kaggle

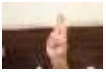
No.	Gambar	Arti	Jumlah
1		1	104
2		2	105
3		3	70
4		4	70
5		5	70
6		6	70
7		7	70
8		8	70
9		9	70

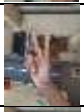
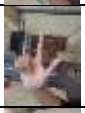
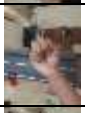
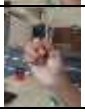
No.	Gambar	Arti	Jumlah
10		10	Tidak Tersedia

Berikut adalah beberapa gambar yang dibuat sendiri:

Tabel 3.3 Gestur Huruf yang dibuat sendiri

No.	Gambar	Arti	Jumlah
1		A	114
2		B	54
3		C	52
4		D	54
5		E	55
6		F	54
7		G	54
8		H	53

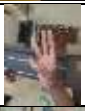
No.	Gambar	Arti	Jumlah
14		O	53
15		P	55
16		Q	53
17		R	54
18		S	53
19		T	53
20		U	53
21		V	53

No.	Gambar	Arti	Jumlah
9		I	53
10		K	54
11		L	52
12		M	53
13		N	53

No.	Gambar	Arti	Jumlah
22		W	55
23		X	52
24		Y	62

Tabel 3.4 Gestur Angka yang dibuat sendiri

No.	Gambar	Arti	Jumlah
1		1	96
2		2	95
3		3	130
4		4	130

No.	Gambar	Arti	Jumlah
5		5	130
6		6	130
7		7	130
8		8	130
9		9	130
10		10	200

Laporan menyebut penggunaan dataset Kaggle dan data yang dibuat sendiri. Pada Tabel 3.2 (Gestur Angka dari Kaggle), untuk angka '10' keterangannya "Tidak Tersedia". Penulis telah menangani ini dengan membuat data sendiri. Ini adalah solusi yang baik.

Nama spesifik dataset Kaggle yang digunakan untuk referensi adalah Sign Language Multi-Class Classfication with CNN. Kekurangan data (seperti angka ’10’ dari Kaggle) ditangani, misalnya dengan menggunakan data yang dibuat sendiri (yang memang ditunjukkan pada table data buatan sendiri memiliki 200 gambar untuk ‘10’).

Pada dataset sampel training didapat gambar secara *real time* dengan total 4362 data, sampel validasi 646 data, sampel test 635 data, kemudian total dataset 6800 data gambar terkait pembagian 80/10/10. Kemudian uji cobanya langsung dengan *OpenCV* secara *real time*.

Dataset terbagi menjadi tiga bagian utama:

- Data pelatihan (*training data*): 80% dari total gambar di setiap kelas yang berjumlah 200. Maka, jumlah data pelatihan adalah sekitar 160 gambar untuk setiap kelas. Total gambar dalam dataset mencapai 5643.
- Data validasi (*validation data*): 10% digunakan untuk menilai kinerja model selama pealtihan. Jumlah data validasi sekitar 20 gambar untuk tiap kelas.
- Data pengujian (*testing data*): 10% digunakan untuk menilai akurasi model setelah proses pelatihan selesai. Jumlah data pengujian sekitar 20 gambar per kelas.

Dan juga, jumlah ekstraksi landmark yang gagal pada bagian preprocessing data untuk memberikan konteks penuh mengenai dataset yang akhirnya digunakan.

- Ekstraksi gagal Pelatihan: 1078
- Ekstraksi gagal Validasi: 34
- Ekstraksi gagal Pengujian: 45

Untuk memastikan bahwa model mampu mengenali gerakan dalam berbagai kondisi, dataset mencakup variasi seperti:

- Pencahayaan yang terang dan redup.
- Latar belakang yang sederhana dan kompleks.
- Sudut pandang kamera yang berbeda.

3.2.2 Preprocessing Data

Langkah ini bertujuan untuk menyiapkann data sehingga dapat dimanfaatkan dalam pelatihan model pembelajaran mesin.

- **Deteksi Titik Penting Tangan**

  MediaPipe digunakan untuk mengidentifikasi dan mengambil 21 titik penting tangan yang menggambarkan posisi sendi jari dan telapak tangan. Masing-masing titik penting memiliki koordinat X, Y, dan Z.

- **Normalisasi Data**

  Data landmark dinormalisasi agar dampak buruk dari pencahayaan dan perbedaan ukuran antara gambar dapat diminimalkan. Langkah normalisasi menjamin bahwa seluruh data memiliki format yang konsisten.

- **Augmentasi Data**

  Teknik augmentasi, seperti rotasi, crop gambar, dan perubahan pencerahan, diterapkan untuk meningkatkan keragaman dataset dan mencegah *overfitting*. Kode augmentasi di Jupyter Notebook tidak berhasil dijalankan atau tidak digunakan untuk dataset final.

  Deskripsikan Proses Augmentasi Manual:

  Augmentasi data dilakukan secara manual menggunakan aplikasi pengolah gambar. Proses yang dilakukan adalah rotasi, zoom, brightness, dark, crop gambar. Perlakuan pengolahan gambar untuk masing-masing gambar berbeda-beda, agar terlihat variasinya. 1 gambar asli menghasilkan n varian. Hal ini dilakukan pada seluruh data yang ada. Berikut ini beberapa contoh augmentasi yang dilakukan

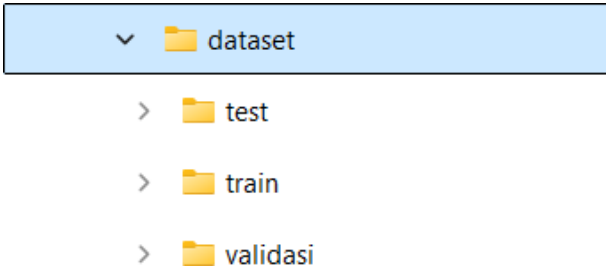
Tabel 3.5 Tabel Struktur Dataset

Simbol	Gambar Asli	Rotasi 30°	Rotasi 15°	Bright terang 50	Bright gelap -60	Crop	Zoom
A							

Tabel 3.6 Lakukan Augmentasi Data untuk Angka 4

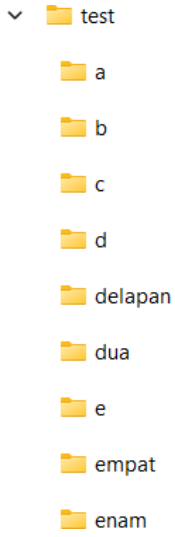
Simbol	Gambar Asli	Rotasi 15°	Bright terang 10	Bright gelap -10	Crop	Zoom perkecil 10%
4						

Hasil pengolahan data dimasukkan ke dalam folder test, train, dan validasi sebagai berikut:



Gambar 3.2 Folder dataset dibagi menjadi 3 folder test, train, validasi

Untuk masing-masing folder, dibuatkan folder sesuai huruf dan angka sebagai berikut



Gambar 3.3 Masing-masing folder terdapat folder angka dan huruf

Kemudian akan dilakukan ekstraksi landmark terhadap semua data sesuai foldernya. Karena jumlah data yang didapat tidak banyak, maka dilakukan data augmentasi gambar asli pada data yang sudah ada,

augmentasi hanya dilakukan untuk variasi model training. Setelah itu, model perlu dilatih ulang untuk meningkatkan akurasi klasifikasi setelah augmentasi data dilakukan.

3.2.3 Pengembangan Model

Tahap ini berfokus pada pelatihan model klasifikasi gerakan tangan menggunakan algoritma deep learning.

- **Arsitektur Model**

  Model CNN digunakan mengekstrak fitur dari data landmark tangan. Arsitektur model dirancang agar ringan dan dapat dioptimalkan untuk pemrosesan real-time.

- **Pelatihan Model**

  Data yang digunakan untuk pelatihan bertujuan untuk melatih model dengan algoritma optimasi seperti Adam.

  - Data validasi berfungsi untuk mengawasi kinerja model dan menghindari *overfitting.*
  - Fungsi kerugian yang diterapkan adalah *categorical cross-entropy*, sedangkan metrik yang digunakan untuk evaluasi adalah akurasi.

3.2.4 Implementasi Aplikasi

Aplikasi dikembangkan menggunakan bahasa pemrograman Python, dengan integrasi MediaPipe untuk pelacakan gerakan tangan dan model CNN untuk klasifikasi gerakan.

- **Antamuka Pengguna**

  Aplikasi menggunakan *Graphical User Interface (GUI)* sederhana yang menampilkan hasil pengenalan gerakan dalam bentuk teks.

- **Input Webcam**

Webcam digunakan sebagai perangkat input untuk menangkap gerakan tangan secara langsung. Data dari webcam diproses secara real-time menggunakan MediaPipe

- **Output teks**

Gerakan tangan yang dikenali ditampilkan dalam bentuk teks di layar, sehingga memudahkan komunikasi dengan orang yang tidak menguasai bahasa isyarat.

3.2.5 Evaluasi dan Uji Coba

Tahap terakhir penelitian adalah evaluasi performa sistem untuk memastikan bahwa aplikasi yang dikembangkan memenuhi tujuan penelitian.

- **Uji Akurasi**

Model diuji pada data pengujian yang mencakup berbagai kondisi pencahayaan, latar belakang, dan variasi pengguna. Akurasi pengenalan dihitung untuk mengukur keandalan sistem.

- **Uji Real-Time**

Latensi sistem diuji untuk memastikan aplikasi dapat memproses gerakan tangan dalam waktu nyata. Latensi yang rendah (<100 ms) dianggap ideal untuk aplikasi ini.

3.3 Perangkat dan Teknologi yang Digunakan

Pada penelitian ini, sejumlah perangkat dan teknologi diterapkan untuk mendukung pengembangan sistem pengenalan bahasa isyarat berbasis *Computer Vision.* Pemilihan perangkat dan teknologi tersebut didasari oleh efisiensi, akurasi, dan kemampuannya dalam membantu proses pengembangan aplikasi secara keseluruhan. Berikut adalah rincian alat dan teknologi yang digunakan:

- Bahasa pemrograman yang digunakan yaitu Python

- Framework dan pustaka pendukung: MediaPipe, Tensorflow dan Keras, OpenCV

3.4 Perangkat Keras

3.4.1 Kamera dengan Resolusi Tinggi

Kamera digunakan untuk menangkap data gerakan tangan secara real-time. Kamera dengan resolusi tinggi untuk memastikan bahwa detail gerakan tangan dapat direkam dengan jelas, bahkan dalam kondisi pencahayaan yang kurang ideal.

Kriteria Kamera:

- Resolusi minimal 720p untuk memastikan keakuratan deteksi landmark tangan.
- Frame rate minimal 30 fps untuk mendukung pelacakan gerakan tangan secara real-time.

3.4.2 Komputer dengan GPU

GPU digunakan untuk mempercepat proses pelatihan model deep learning. Pemilihan GPU didasarkan pada kemampuan untuk memproses data dalam jumalh besar dan melakukan operasi matriks dengan cepat.

Spesifikasi Minimum dari komputer yang akan digunakan

- GPU dengan memori minimal 6GB (misalnya NVIDIA GeForce GTX 1660).
- RAM minimal 16 GB untuk mendukung pengolahan data yang efisien.
- CPU dengan kecepatan tinggi untuk mendukung integrasi sistem secara keseluruhan.

3.4.3 Lingkungan Pengembangan

a. Jupyter Notebook

Digunakan sebagai alat utama untuk pengembangan dan eksplorasi model machine learning. Juypter Notebook memungkinkan peneliti untuk menulis kode, menjalankan eksperimen, dan mendokumentasikan hasil dalam satu platform.

b. Visual Studio Code (VS Code)

Editor teks yang digunakan untuk pengembangan aplikasi secara keseluruhan VS Code dipilih karena:

- Mendukung berbagai ekstensi yang mempermudah pengembangan dengan Python dan MediaPipe.
- Antarmuka yang sederhana dan efisien untuk pengelolaan proyek.

c. Google Colab

Platform berbasis cloud yang digunakan untuk pelatihan model dalam skala besar. Google Colab menyediakan akses gratis ke GPU dan TPU, yang mempercepat proses pelatihan tanpa memerlukan perangkat keras lokal yang canggih.

3.5 Alat Pendukung Lainnya

3.5.1 Git dan Github

- **Git:** Digunakan untuk pengelolaan versi kode sumber, memastikan bahwa perubahan kode terdokumentasi dengan baik dan dapat dikembalikan jika diperlukan.
- **GitHub:** Digunakan sebagai repositori online untuk menyimpan kode proyek dan memfasilitas kolaborasi jika ada tim pengembang.

3.5.2 TensorBoard

TensorBoard digunakan untuk memvisualisasikan metrik pelatihan model, seperti nilai *loss* dan akurasi, yang memudahkan analisis dan pengambilan keputusan selama proses pelatihan.

Dengan memanfaatkan alat dan teknologi ini, studi ini diharapkan dapat menciptakan aplikasi yang dapat mendeteksi gerakan bahasa isyarat secara langsung dengan akurasi tinggi, waktu respons yang cepat, dan mudah diimplementasikan pada perangkat keras yang umum tersedia.

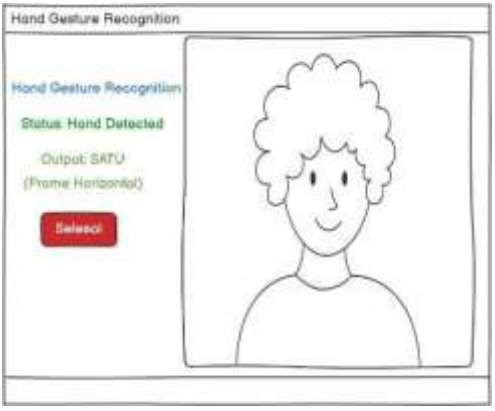
3.6 Perancangan User Interface / *Mock-up* aplikasi

Sistem pengenalan bahasa isyarat ini dibangun dengan memanfaatkan teknologi **MediaPipe Hands** untuk mendeteksi pose tangan dan landmark (titik-titik kunci) secara real-time. Data landmark tersebut kemudian diekstraksi dan diklasifikasikan oleh model **Convolutional Neural Network (CNN)** untuk mengenali gesture tangan pengguna.

Antarmuka pengguna (User Interface/UI) memiliki peranan penting dalam menjembatani interaksi antara pengguna dan sistem. Dalam penelitian ini, antarmuka dirancang untuk menyederhanakan proses interaksi antara pengguna dengan sistem pengenalan bahasa isyarat. Desain ini dikembangkan menggunakan alat mockup Balsamiq, sebagai acuan visual sebelum direalisasikan dalam bentuk kode program menggunakan pustaka Tkinter pada bahasa pemrograman Python. UI ditampilkan dalam bentuk jendela aplikasi dengan fitur utama sebagai berikut:

a. Desain Antarmuka Sistem

Gambar 3.4 berikut menunjukkan desain awal antarmuka aplikasi pengenalan bahasa isyarat menggunakan teknologi *Computer Vision*:



Gambar 3.4 Mock-up (Sketsa Desain)

Antarmuka ini dirancang dengan pendekatan yang sederhana dan minimalis agar dapat dioperasikan dengan mudah oleh pengguna umum, khususnya penyandang tunarungu.

Berdasarkan Gambar 3.4, berikut adalah penjelasan masing-masing komponen utama dalam UI tersebut:

Tabel 3.7 Komponen utama dalam User Interface

No.	Komponen Antarmuka	Deskripsi
1	**Judul Aplikasi**	Menampilkan judul sistem: Hand Gesture Recognition, sebagai identitas aplikasi.
2	**Label Status**	Menampilkan status deteksi tangan secara real-time, misalnya Status: Hand Deetcted (berwarna hijau) atau Tidak Terdeteksi (kuning).
3	**Label Output**	Menampilkan hasil kklasifikasi gesture tangan, misalnya: Output: SATU
4	**Keterangan (Frame Horizontal)**	Memberikan informasi orientasi kamera, seperti Frame Horizontal. Jika merujuk pada proses cv2.flip(frame, 1), pastikan

		ini dikomunikasikan dengan jelas agar tidak membingungkan pengguna akhir.
5	**Tombol 'Selesai'**	Tombol berwarna merah yang digunakan untuk keluar dari aplikasi.
6	**Tampilan Kamera (Webcam)**	Area di sisi kanan antarmuka menampilkan video secara real-time dari webcam, termasuk visualisasi landmark tangan yang dideteksi oleh MediaPipe.

b. Tujuan Desain UI

Tujuan dari perancangan antarmuka ini antara lain:

- **Kesederhanaan,** agar pengguna fokus pada fungsi terhadap hasil gesture yang dikenali;
- **Interaktif dan Informatif,** dengan memberikan feedback status deteksi secara langsung;
- **Responsif,** memperbarui prediksi dan status tangan secara real-time;
- **Aksesibilitas,** warna dan font disesuaikan agar mudah dibaca oleh berbagai kalangan pengguna.

Dengan antarmuka ini kemudian diimplementasikan dalam sistem desktop menggunakan pustaka Tkinter, dan terintegrasi langsung dengan proses pengambilan video dari webcam serta pemodelan CNN untuk klasifikasi gesture.

c. Visualisasi Awal

Mockup pada Gambar 3.4 berperan sebagai **acuan pengembangan GUI**, yang kemudian direalisasikan dalam implementasi Python. Desain ini mendukung integrasi sistem real-time recognition secara efisien.

d. Tools Pengembangan UI

Untuk membangun dan menguji antarmuka serta sistem secara keseluruhan, digunakan **Visual Studio Code (VS Code)** sebagai editor teks utama karena:

- Mendukung berbagai ekstensi untuk Python dan MediaPipe.
- Antarmuka yang efisien untuk manajemen proyek.
- Terminal terintegrasi untuk eksekusi dan pengujian skrip secara langsung.

Pada perancangan user interface (UI) yaitu Membuat rancangan input/output, namun secara jelas, bab ini bukan berisi kode program, melainkan semua yang menjadi dasar kode program. Tidap perlu terlalu rinci (karena akan sama dengan kode program) tapi juga jangan dibuat umum (karena tidak memberikan gambaran yang cukup untuk implementasi program). Karena levelnya baru pada tahap desain, maka tidak diperbolehkan untuk melakukan capture pada layar (terutama pada desain user interface). Desain antarmuka mencakup:

- Tampilan jendela kamera real-time
- Indikator proses deteksi tangan
- Kotak teks hasil klasifikasi
- Tombol mulai/berhenti sistem

3.7 Arsitektur Sistem

Berikut tahapan alur kerja sistem:

1. **Akuisisi Citra (Image Acquisition):**

   Gambar tangan pengguna diambil melalui kamera atau dataset citra statis.
2. **Deteksi Landmark dengan MediaPipe:**

   MediaPipe Hands mendeteksi hingga 21 titik kunci (landmark) pada tangan. Setiap landmark memiliki koordinat **(x, y, z).**
3. **Ekstraksi Fitur:**

   Koordinat landmark digunakan sebagai fitur utama. Normalisasi dilakukan agar fitur tidak terpengaruh oleh posisi tangan atau ukuran gambar.

4. **Data Augmentasi:**

   Gambar diperluas dengan transformasi seperti rotasi, zoom, crop, dan perubahan pencahayaan.

5. **Klasifikasi Gestur:**

   CNN (Convolutional Neural Network) digunakan untuk mengklasifikasikan fitur landmark ke salah satu dari 34 kelas gesture (A-Y dan 1-10).

6. **Evaluasi Model:**

   Model dievaluasi menggunakan data validasi dan efektivitasnya diukur dengan confusion matrix, presisi, recall, dan skor F1.

3.8 Rancangan Deteksi Landmark

Pada tahapan ini, sistem dibangun untuk mendeteksi posisi titik-titik penting pada tangan (landmark) menggunakan teknologi MediaPipe sebagai alat utama. Pemilihan Framework ini dilatarbelakangi oleh keunggulannya dalam mengenali struktur tangan secara akurat dan efisien tanpa memerlukan penanda fisik tambahan (*markerless detection*).

- **Pustaka yang Digunakan**

  Solusi *MediaPipe Hands* digunakan karena memiliki perfroma ringan serta mampu bekerja dalam kondisi pencahayaan dan latar belakang. Framework ini dapat diintegrasikan dengan baik dalam aplikasi berbasis Python dan cocok untuk kebutuhan deteksi real-time

- **Hasil Deteksi**

  Sistem menghasilkan 21 titik utama (*landmark*) pada area tangan yang berhasil terdeteksi. Masing-masing titik ini memiliki koordinat dalam ruang tiga dimensi (3D), yakni nilai x (horizontal), y (vertical), dan z (kedalaman relatif terhadap bidang gambar).

- **Contoh Hasil Koordinat Landmark tangan** yang dihasilkan oleh **MediaPipe Hands**:

  Titik 0: x = 0.45, y = 0.68, z = -0.032

  Titik 1: x = 0.47, y = 0.65, z = -0.028

  Titik 2: x = 0.49, y = 0.62, z = -0.024

  Titik 3: x = 0.51, y = 0.60, z = -0.020

  Titik 4: x = 0.53, y = 0.58, z = -0.017

  Titik 5: x = 0.46, y = 0.62, z = -0.030

  Titik 6: x = 0.48, y = 0.59, z = -0.026

  Titik 7: x = 0.50, y = 0.56, z = -0.021

  Titik 8: x = 0.52, y = 0.54, z = -0.018

  Titik 9: x = 0.44, y = 0.61, z = -0.033

  Titik 10: x = 0.46, y = 0.58, z = -0.029

  Titik 11: x = 0.48, y = 0.56, z = -0.024

  Titik 12: x = 0.50, y = 0.54, z = -0.020

  Titik 13: x = 0.43, y = 0.60, z = -0.034

  Titik 14: x = 0.45, y = 0.57, z = -0.030

  Titik 15: x = 0.47, y = 0.55, z = -0.025

  Titik16: x = 0.49, y = 0.53, z = -0.022

  Titik 17: x = 0.42, y = 0.61, z = -0.035

  Titik 18: x = 0.44, y = 0.59, z = -0.031

  Titik 19: x = 0.46, y = 0.57, z = -0.027

Titik 20: x = 0.48, y = 0.55, z = -0.023

Nilai koordinat tersebut dapat bervariasi tergantung pada posisi tangan pengguna di depan kamera.

- **Pemanfaatan Data Landmark**

  Koordinat dari masing-masing titik tersebut kemudian digunakan sebagai input bagi model klasifikasi yang dirancang untuk mengenali dan mengelompkkan gerakan bahasa isyarat ke dalam kelas yang sesuai.

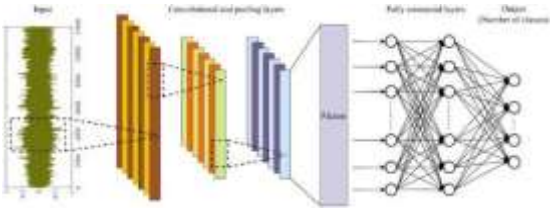
3.9 Implementasi Model CNN

Pada tahap klasifikasi gestur, penelitian ini menerapkan arsitektur **Convolutional Neural Network** (**CNN**) sebagai algoritma utama untuk mengenali pola dari data landmark yang telah diekstraksi melalui MediaPipe. CNN dipilih karena kemampuannya mengekstraksi dan mempelajari fitur spasial dan temporal dari data sekuensial, yang sangat relevan untuk pengenalan pola gesture tangan. Model yang diimplementasikan adalah CNN 1D yang dirancang khusus untuk memproses data landmark tangan yang berbentuk vektor fitur satu dimensi.

Penggunaan CNN 1D memungkinkan pemrosesan data sekuensial dengan efisien, di mana setiap fitur landmark tangan direpresentasikan sebagai urutan nilai numerik yang merefleksikan posisi koordinat tangan dalam ruang tiga dimensi. Pendekatan ini memanfaatkan kekuatan CNN dalam mendeteksi pola lokal dan hierarki fitur yang penting untuk klasifikasi gesture secara akurat.

3.9.1 Arsitektur Model CNN 1D

Model tersusun atas kombinasi lapisan konvolusional dan *max-pooling* yang diikuti oleh lapisan fully connected (dense) untuk klasifikasi akhir. Arsitektur ini dirancang untuk mengekstrak fitur penting dari data input dan mengurangi dimensi data secara bertahap sebelum dilakukan klasifikasi.



Gambar 3.5 Diagram Arsitektur Model CNN 1D

Gambar di atas menunjukkan diagram arsitektur model CNN 1D yang digunakan. Model menerima input berupa data landmark tangan dengan dimensi 63×1, yang merupakan hasil reshape dari 21 titik landmark tangan dengan 3 koordinat (x, y, z) masing-masing.

3.9.2 Penjabaran Arsitektur Model

3.9.2.1 Lapisan Konvolusi Satu Dimensi (Conv1D)

Lapisan **Conv1D** berfungsi untuk mengekstraksi fitur lokal dari data berurutan, khususnya vektor landmark tangan. Dalam arsitektur yang diimplementasikan, terdapat dua lapisan Conv1D yang masing-masing dilengkapi dengan **32** dan **64 filter**, dengan **ukuran kernel 3**. Fungsi aktivasi yang digunakan adalah **ReLU (Rectified Linear Unit)**, yang efektif dalam mempelajari relasi non-linier antar fitur.

- **Conv1D pertama** menerima input berdimensi *(63, 1)* dan mengeluarkan 32 buah *feature maps* berukuran 61 (karena kernel size = 3).
- **Conv1D kedua** memproses output dari lapisan sebelumnya dan menghasilkan 64 peta fitur dengan konfigurasi kernel serupa.

3.9.2.2 Lapisan MaxPooling1D

Setelah masing-masing lapisan konvolusi, diterapkan proses **max pooling** menggunakan **MaxPooling1D** dengan ukuran jendela (*pool size*) 2. Pooling ini bertujuan untuk mereduksi dimensi spasial dari hasil ekstraksi fitur, sehingga beban komputasi lebih ringan serta membantu mengurangi risiko *overfitting*. Proses ini mempertahankan fitur paling dominan dalam setiap area lokal hasil konvolusi.

3.9.2.3 Lapisan Flatten

Lapisan **Flatten** bertugas untuk mengubah data dari bentuk dua dimensi (matriks hasil pooling) menjadi satu vektor linier. Proses ini diperlukan agar data dapat diteruskan ke lapisan berikutnya yang bertipe *fully connected* (dense). Dengan kata lain, Flatten menjadi jembatan antara tahap ekstraksi fitur dan tahap klasifikasi.

3.9.2.4 Lapisan Fully Connected (Dense) dan Regularisasi Dropout

Model terdiri dari tiga lapisan dense yang masing-masing dirancang dengan peran sebagai berikut:

- Lapisan dense pertama terdiri dari **128 neuron** dan menggunakan aktivasi **ReLU**, disertai **Dropout sebesar 0.3** sebagai metode regulasi untuk menonaktifkan sejumlah neuron secara acak selama pelatihan.

- Lapisan dense kedua memiliki **64 neuron** dengan fungsi aktivasi **ReLU**, serta **Dropout 0.2** untuk memperkecil kemungkinan *overfitting*.
- Lapisan output memuat **34 neuron**, yang masing-masing mewakili satu kelas gestur (huruf dan angka ASL), dan menggunakan **fungsi aktivasi softmax** sebagai mekanisme klasifikasi multikelas.

Teknik **Dropout** ini menjadi kunci dalam meningkatkan kemampuan generalisasi model, khususnya saat dihadapkan pada data baru atau data uji.

3.9.3 Parameter Pelatihan

Pelatihan model dilakukan dengan parameter sebagai berikut:

- **Batch size:** 16, dipilih karena dataset relatif kecil sehingga batch kecil membantu stabilitas pembelajaran.
- **Epochs:** Maksimal 100, dengan sistem penghentian awal yang akan menghentikan proses pelatihan jika tidak terdapat peningkatan validasi selama 15 epoch berturut-turut.
- **Optimizer:** Adam dengan learning rate 0.001, yang merupakan algoritma optimasi adaptif yang populer dan efektif untuk pelatihan jaringan saraf dalam.
- **Loss function:** Categorical crossentropy, sesuai untuk klasifikasi multi-kelas.
- **Callbacks:**
    - EarlyStopping untuk mencegah overfitting dengan menghentikan pelatihan lebih awal.
    - ReduceLROnPlateau untuk menurunkan learning rate secara otomatis jika validasi loss tidak membaik selama 5 epoch.

Pengaturan ini bertujuan untuk mencapai keseimbangan antara kecepatan konvergensi dan kemampuan generalisasi model.

3.9.4 Pra-Pemrosesan Data

Data yang digunakan adalah landmark tangan yang telah diekstraksi menggunakan MediaPipe, berupa koordinat 3D dari 21 titik landmark tangan. Setiap sampel data memiliki dimensi awal (21, 3), yang kemudian di-flatten menjadi vektor 63 fitur. Langkah pra-pemrosesan meliputi:

- **Load data:** Data training, validasi, dan testing dimuat dari file numpy (.npy).
- **Reshape data:** Data diubah bentuknya menjadi (jumlah sampel, 63, 1) agar sesuai dengan input Conv1D.
- **One-hot encoding:** Label kelas diubah menjadi format one-hot vector untuk klasifikasi multi-kelas.
- **Normalisasi:** (Jika diperlukan) Data landmark dapat dinormalisasi untuk mempercepat pelatihan dan meningkatkan stabilitas model.

Pra-pemrosesan ini memastikan data siap untuk dimasukkan ke dalam model CNN dengan format yang tepat dan representasi yang optimal.

3.9.5 Struktur Kode Program

Kode program implementasi model CNN ini disusun secara modular dan sistematis, terdiri dari beberapa bagian utama:

- **Import library:** Memuat pustaka yang diperlukan seperti numpy, tensorflow.keras, dan matplotlib untuk visualisasi.
- **Load dan pra-pemrosesan data:** Memuat data dari file, melakukan reshape dan encoding label.
- **Definisi model:** Fungsi create_model() yang membangun arsitektur CNN sesuai spesifikasi.

- **Callbacks:** Definisi mekanisme early stopping dan pengurangan learning rate.
- **Pelatihan model:** Melakukan training dengan data training dan validasi, serta menyimpan riwayat pelatihan.
- **Evaluasi:** Mengukur performa model pada data testing dan menampilkan akurasi.
- **Visualisasi:** Plot grafik akurasi dan loss selama pelatihan untuk analisis performa.
- **Penyimpanan model:** Menyimpan model terlatih dalam format .h5 untuk penggunaan selanjutnya.

Struktur ini memudahkan pemeliharaan kode dan memungkinkan pengembangan lebih lanjut dengan mudah.

3.9.6 Strategi Evaluasi Model

Untuk mengukur kinerja model klasifikasi yang dikembangkan, digunakan beberapa indicator evaluasi utama sebagai berikut:

- **Akurasi (Accuracy):**

  Merupakan ukuran persentase jumlah prediksi yang benar terhadap seluruh data pengujian. Metrik ini menunjukkan sejauh mana model dapat mengenali dan mengelompokkan gestur tangan secara tepat.
- **Loss Function:**

  Fungsi kerugian yang digunakan adalah **categorical crossentropy**, yang mengukur perbedaan antara distribusi prediksi model dan label sebenarnya. Semakin kecil nilai loss, semakin akurat model dalam memperkirakan kelas target.
- **Kurva Pelatihan dan Validasi:**

  Proses pelatihan model dimonitor melalui grafik visual yang memperlihatkan perubahan akurasi dan loss pada

data pelatihan dan validasi selama beberapa epoch. Pola grafik ini dapat membantu mendeteksi tanda-tanda **overfitting** (model terlalu menyesuaikan data latih) maupun **underfitting** (model gagal belajar pola penting).

- **Callback Training:**

  Untuk menjaga kualitas pelatihan model, digunakan beberapa mekanisme otomatis seperti:

  - **EarlyStopping**, yang menghentikan pelatihan ketika performa model pada data validasi tidak menunjukkan peningkatan.
  - **ReduceLROnPlateau**, yang menurunkan nilai *learning rate* secara bertahap jika tidak ada peningkatan akurasi, guna membantu proses konvergensi lebih stabil.

Evaluasi secara menyeluruh ini memberikan pemahaman lbih mendalam terhadap efektivitas arsitektur CNN yang diterapkan dalam mendeteksi gestur berbasis koordinat landmark, serta membantu dalam pengambilan keputusan untuk penyempurnaan model di tahap selanjutnya.