

TUGAS AKHIR - IF184802

Pengenalan Video Peraga Menggunakan Transformer untuk Pembelajaran Bahasa Isyarat

ADAM HADI PRASETYO

NRP 05111940000224

Dosen Pembimbing 1

Prof Dr. Eng. Nanik Suciati, S.Kom, M.Kom

NIP 197104281994122001

Dosen Pembimbing 2

Dr. Anny Yuniarti, S.Kom., M.Comp.Sc.

NIP 198106222005012002

Program Studi S1 Teknik Informatika

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2023



TUGAS AKHIR - IF184802

Pengenalan Video Peraga Menggunakan Transformer untuk Pembelajaran Bahasa Isyarat

ADAM HADI PRASETYO

NRP 05111940000224

Dosen Pembimbing 1

Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom

NIP 197104281994122001

Dosen Pembimbing 2

Dr. Anny Yuniarti, S.Kom., M.Comp.Sc.

NIP 198106222005012002

Program Studi S1 Teknik Informatika

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2023



FINAL PROJECT - IF184802

VIDEO DEMONSTRATIONS RECOGNITION USING TRANSFORMERS FOR SIGN LANGUAGE LEARNING

ADAM HADI PRASETYO

NRP 05111940000224

Advisor 1

Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom

NIP 197104281994122001

Advisor 2

Dr. Anny Yuniarti, S.Kom., M.Comp.Sc.

NIP 198106222005012002

Study Program Bachelor of Informatics

Department of Informatics

Faculty of Intelligent Electrical and Informatics Technology

Institut Teknologi Sepuluh Nopember

Surabaya

2023

LEMBAR PENGESAHAN

Pengenalan Video Peraga Menggunakan Transformer Untuk Pembelajaran Bahasa Isyarat

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
Memperoleh gelar Sarjana Komputer pada
Program Studi S-1 Teknik Informatika
Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh : **ADAM HADI PRASETYO**

NRP. 05111940000224

Disetujui oleh Tim Penguji Tugas Akhir:

- | | | |
|----|---|---------------|
| 1. | Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom | Pembimbing |
| 2. | Dr. Anny Yuniarti, S.Kom., M.Comp.Sc. | Ko-pembimbing |
| 3. | Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D. | Penguji |
| 4. | Dini Adni Navastara, S.Kom., M.Sc. | Penguji |

SURABAYA
Juli, 2023

APPROVAL SHEET

VIDEO DEMONSTRATIONS RECOGNITION USING TRANSFORMERS FOR SIGN LANGUAGE LEARNING

FINAL PROJECT

Submitted to fulfill one of the requirements
for obtaining a degree Computer Science at
Undergraduate Study Program of Informatics Engineering
Department of Informatics
Faculty of Intelligent Electrical and Information Technology
Institut Teknologi Sepuluh Nopember

By: **ADAM HADI PRASETYO**

NRP. 05111940000224

Approved by Final Project Examiner Team:

- | | | |
|----|---|------------|
| 1. | Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom | Advisor |
| 2. | Dr. Anny Yuniarti, S.Kom., M.Comp.Sc. | Co-Advisor |
| 3. | Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D. | Examiner |
| 4. | Dini Adni Navastara, S.Kom., M.Sc. | Examiner |

SURABAYA
July , 2023

PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini:

Nama mahasiswa / NRP : Adam Hadi Prasetyo
Departemen : Teknik Informatika
Dosen Pembimbing / NIP : Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom
Dr. Anny Yuniarti, S.Kom., M.Comp.Sc.

Dengan ini menyatakan bahwa Tugas Akhir dengan judul “Pengenal Video Peraga Menggunakan Transformer Untuk Pembelajaran Bahasa Isyarat” adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.

Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima saksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, 1 Agustus 2023

Mahasiswa

Adam Hadi Prasetyo
NRP. 05111940000224

Mengetahui
Dosen Pembimbing 1

Mengetahui
Dosen Pembimbing 2

Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom
NIP. 197104281994122001

Dr. Anny Yuniarti, S.Kom., M.Comp.Sc.
NIP. 198106222005012002

[Halaman ini sengaja dikosongkan]

STATEMENT OF ORIGINALITY

The undersigned below:

Name of student / NRP : Adam Hadi Prasetyo
Department : Informatics
Advisor / NIP : Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom
Dr. Anny Yuniarti, S.Kom., M.Comp.Sc.

Hereby declare that Final Project with the title of “Video Demonstrations Recognition Using Transformers for Sign Language Learning” is the result of my own work, is original, and is written by following the rules of scientific writing.

If in the future there is a discrepancy with this statement, then I am willing to accept sanctions in accordance with the provisions that apply at Institut Teknologi Sepuluh Nopember.

Surabaya, 1 August 2023

Student

Adam Hadi Prasetyo
NRP. 05111940000224

Acknowledge

Advisor 1

Acknowledge

Advisor 2

Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom
NIP. 197104281994122001

Dr. Anny Yuniarti, S.Kom., M.Comp.Sc.
NIP. 198106222005012002

[Halaman ini sengaja dikosongkan]

PENGENALAN VIDEO PERAGA MENGGUNAKAN TRANSFORMER UNTUK PEMBELAJARAN BAHASA ISYARAT

Nama Mahasiswa / NRP : Adam Hadi Prasetyo / 05111940000224
Departemen : Teknik Informatika FTEIC - ITS
Dosen Pembimbing : Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom
Dr. Anny Yuniarti, S.Kom., M.Comp.Sc.

Abstrak

Dalam kehidupan sehari-hari, komunikasi merupakan aktivitas manusia yang paling sering dilakukan. Komunikasi bertujuan untuk menyampaikan maksud dalam bentuk lisan, tulisan, gambar atau gestur. Komunikasi dengan gestur biasa disebut bahasa isyarat. Pada umumnya bahasa isyarat digunakan karena keterbatasan fisik yang tidak memungkinkan penggunaan komunikasi lisan. Standar Isyarat Bahasa Indonesia (SIBI) memberikan panduan dalam bentuk buku, namun pembelajaran dengan media buku kurang efektif karena tidak dapat memberikan contoh riil. Selain itu buku kurang praktis untuk dibawa. Bahasa isyarat dapat dikenali oleh komputer melalui pengenalan gambar (*image recognition*). Sebelum komputer mampu mengenali gambar yang diberikan, terdapat model pengenalan gambar yang harus dilatih dengan *machine learning*. Transformer adalah salah satu metode *machine learning* yang dapat digunakan dalam melatih model. Transformer merupakan implementasi mekanisme *self-attention* pada sebuah model, yang dapat memfokuskan bagian *input* yang diberikan. Mekanisme *attention* yang diberikan pada transformer memungkinkan model untuk mendapatkan konteks dari setiap bagian *input*. Model transformer yang digunakan pada tugas akhir ini menggunakan variasi *encoder only*. Model memiliki 4 *multihead*, 1 *encoder layer* dan dilatih selama 50 *epoch*. Dari pengaturan model tersebut dihasilkan akurasi *test* sebesar 100% dengan presisi *test* 100%. Setelah model transformer *encoder only* berhasil dilatih, aplikasi pembelajaran bahasa isyarat dirancang dan dikembangkan menggunakan model untuk mengenali gestur bahasa isyarat secara *real time*. Selain itu, aplikasi pembelajaran bahasa isyarat memiliki fitur untuk memutar video peraga bahasa isyarat.

Kata kunci: Bahasa Isyarat, SIBI, *Image Recognition*, *Machine Learning*, Transformer, *Self-Attention*, *Encoder Only*

[Halaman ini sengaja dikosongkan]

VIDEO DEMONSTRATIONS RECOGNITION USING TRANSFORMERS FOR SIGN LANGUAGE LEARNING

Student Name / NRP : Adam Hadi Prasetyo / 05111940000224

Department : Informatics Engineering FTEIC - ITS

**Advisor : Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom
Dr. Anny Yuniarti, S.Kom., M.Comp.Sc.**

Abstract

In everyday life, communication is the most frequently performed activity by humans. Communication aims to convey intentions in the form of speech, writing, images, or gestures. Communication through gestures is commonly known as sign language. Sign language is generally used due to physical limitations that prevent the use of oral communication. The Indonesian Sign Language Standard (SIBI) provides guidelines in the form of a book, but learning through books is less effective as they cannot provide real-life examples. Additionally, books are not practical to carry around. Sign language can be recognized by computers through image recognition. Before a computer can recognize the given images, there is a need to train an image recognition model using machine learning. Transformer is one of the machine learning methods that can be used to train the model. Transformer is an implementation of the self-attention mechanism in a model, which allows the model to focus on specific parts of the input. The attention mechanism provided by the transformer enables the model to capture the context of each input component. The transformer model used in this final project utilizes an encoder-only variation. The model consists of 4 multiheads, 1 encoder layers, and is trained for 50 epochs. From the configuration of the model, an test accuracy of 100% and a test precision of 100% are achieved. After successfully training the transformer encoder-only model, an application for sign language learning is designed and developed using the model to recognize sign language gestures in real time. Additionally, the sign language learning application features the playback of instructional sign language videos.

Keywords: Sign Language, SIBI, Image Recognition, Machine Learning, Transformer, Self-Attention, Encoder Only

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT atas hidayah, rahmat, dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul:

Pengenalan Video Peraga Menggunakan Transformer untuk Pembelajaran Bahasa Isyarat

Tugas Akhir ini dibuat untuk memenuhi salah satu syarat kelulusan dalam meraih derajat sarjana Teknik Informatika program Strata Satu (S-1) Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember.

Penulis menyadari bahwa penulisan buku Tugas Akhir ini tidak luput dari kendala. Penulisan ini tidak mungkin dapat terselesaikan tanpa adanya dukungan dan bantuan baik moril maupun materiil, serta bimbingan dari berbagai pihak. Maka dari itu, penulis menyampaikan rasa terima kasih yang sebesar-besarnya kepada semua pihak yang telah membantu proses penyelesaian penulisan ini:

1. Allah SWT, karena rahmat-Nya penulis dapat menyelesaikan Tugas Akhir dan juga perkuliahan di Departemen Teknik Informatika ITS.
2. Segenap keluarga terutama kedua orang tua penulis yang senantiasa memberikan dukungan dan doa selama penulis mengerjakan Tugas Akhir ini.
3. Ibu Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom dan Ibu Dr. Anny Yuniarti, S.Kom, M.Kom selaku dosen pembimbing penulis yang senantiasa memberikan arahan dalam pengerjaan Tugas Akhir ini.
4. Ibu Prof. Dr. Eng. Chastine Fatichah, S.Kom, M.Kom, selaku kepala Departemen Teknik Informatika ITS.
5. Teman-teman dekat yang memberikan dukungan saat pengerjaan tugas akhir yaitu Aya, Rafif, Arka, Ida dan Hasnan.
6. Teman-teman selama perkuliahan di Teknik Informatika ITS yaitu, Gerry, Bayu, Syafiq, Thomas, Evelyn, Fredrick, Raja, Fitrah, Rihan, Alfarabi, Tsania, Lathifa, serta teman-teman Angkatan TC '19.
7. Bapak, Ibu, dan teman-teman kosan yang menemani selama perkuliahan di Teknik Informatika ITS.
8. Orang-orang yang tidak dapat disebutkan satu-persatu oleh penulis, yang telah membantu penulis selama perkuliahan.

Penulis telah berusaha sebaik-baiknya dalam menyelesaikan Tugas Akhir ini. Namun apabila terdapat kekurangan ataupun kesalahan yang penulis lakukan, penulis mohon maaf. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan untuk kedepannya.

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

HALAMAN JUDUL.....	i
LEMBAR PENGESAHAN	iii
PERNYATAAN ORISINALITAS	v
ABSTRAK	ix
KATA PENGANTAR	xiii
DAFTAR ISI.....	xv
DAFTAR GAMBAR	xix
DAFTAR TABEL.....	xxi
DAFTAR KODE SEMU	xxiii
DAFTAR KODE SUMBER	xxv
BAB I PENDAHULUAN.....	1
1.1 Latar belakang	1
1.2 Rumusan Permasalahan.....	2
1.3 Batasan Masalah.....	2
1.4 Tujuan.....	2
1.5 Manfaat.....	2
BAB II TINJAUAN PUSTAKA.....	3
2.1 Penelitian Terkait.....	3
2.2 Dasar Teori	5
2.2.1 Ekstraksi Fitur.....	5
2.2.2 Augmentasi Data	7
2.2.3 Metrik Evaluasi.....	7
2.2.4 Deep Learning	8
2.2.5 CNN.....	9
2.2.6 RNN.....	10
2.2.7 Transformer	10
2.2.4.1 Encoder-Decoder	11
2.2.4.2 Encoder Only	12
2.2.4.3 Decoder Only.....	12
2.2.5 TensorFlow	13
2.2.6 OpenCV	13
2.2.7 Tkinter	14
BAB III METODOLOGI.....	15

3.1	Metode yang dirancang	15
3.1.1	Perancangan Model Pengenalan Bahasa Isyarat Menggunakan Transformer ...	15
3.1.2	Perancangan Aplikasi Pembelajaran Bahasa Isyarat.....	16
3.1.3	Dataset.....	17
3.2	Peralatan pendukung	18
3.3	Rancangan dan Implementasi Aplikasi	18
3.3.1	Pseudocode.....	18
3.3.1.1	Pelabelan Video.....	18
3.3.1.2	Ekstraksi Fitur	19
3.3.1.3	Normalisasi Data	19
3.3.1.4	Augmentasi Data	19
3.3.1.5	Train dan Evaluasi Model	20
3.3.1.6	Modul Belajar.....	20
3.3.1.7	Modul Evaluasi	21
3.3.2	Model Pengenalan Bahasa Isyarat.....	21
3.3.3	Aplikasi Pembelajaran Bahasa Isyarat	24
3.3.3.1	Modul Belajar.....	24
3.3.3.2	Modul Evaluasi	25
BAB IV HASIL DAN PEMBAHASAN		27
4.1	Analisis Model Pengenalan Bahasa Isyarat	27
4.1.1	Analisis Hasil Skenario 1	27
4.1.2	Analisis Hasil Skenario 2	28
4.1.3	Analisis Hasil Skenario 3	29
4.1.4	Analisis Hasil Skenario 4	29
4.1.5	Analisis Hasil Skenario 5	30
4.1.6	Analisis Hasil Skenario 6	31
4.1.7	Analisis Hasil Skenario 7	31
4.1.8	Analisis Hasil Skenario 8	32
4.1.9	Analisis Uji Coba	33
4.2	Analisis Aplikasi Pembelajaran Bahasa Isyarat	33
4.2.1	Modul Belajar Aplikasi Pembelajaran Bahasa Isyarat.....	34
4.2.2	Modul Evaluasi Aplikasi Pembelajaran Bahasa Isyarat.....	35
4.3	Pembahasan.....	36
4.3.1	Pembahasan Model Pengenalan Bahasa Isyarat.....	36

4.3.2 Pembahasan Aplikasi Pembelajaran Bahasa Isyarat	40
BAB V KESIMPULAN DAN SARAN.....	43
5.1 Kesimpulan.....	43
5.2 Saran	43
DAFTAR PUSTAKA	45
LAMPIRAN-LAMPIRAN.....	47
Proses <i>Training</i> Uji Coba Skenario 1 Tanpa Data Augmentasi.....	47
Proses <i>Training</i> Uji Coba Skenario 2 Tanpa Data Augmentasi.....	47
Proses <i>Training</i> Uji Coba Skenario 3 Tanpa Data Augmentasi.....	47
Proses <i>Training</i> Uji Coba Skenario 4 Tanpa Data Augmentasi.....	48
Proses <i>Training</i> Uji Coba Skenario 5 Tanpa Data Augmentasi.....	48
Proses <i>Training</i> Uji Coba Skenario 6 Tanpa Data Augmentasi.....	48
Proses <i>Training</i> Uji Coba Skenario 7 Tanpa Data Augmentasi.....	49
Proses <i>Training</i> Uji Coba Skenario 8 Tanpa Data Augmentasi.....	49
BIODATA PENULIS	51

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

Gambar 2.1 Contoh Ekstraksi Fitur Keypoints (Sumber : DebuggerCaffe)	6
Gambar 2.2 MediaPipe Holistic Solution (Sumber : MediaPipe).....	6
Gambar 2.3 Landmark yang ada pada bagian tubuh dan telapak tangan (Sumber : MediaPipe)	7
Gambar 2.4 <i>Confusion Matrix</i> untuk mengevaluasi kebenaran prediksi	8
Gambar 2.5 Arsitektur Deep Learning dengan Multilayer Neural Network (LeCun et al, 2015).....	9
Gambar 2.6 Arsitektur CNN VGG-16 (Le, 2017)	10
Gambar 2.7 Arsitektur Umum RNN Terkompresi (kiri) dan Tidak Terkompresi (kanan) (Sumber : wikipedia.org/wiki/Recurrent_neural_network).....	10
Gambar 2.8 Arsitektur Transformer (Vaswan, 2017).	11
Gambar 2.9 Arsitektur BERT yang merupakan Transformer Encoder Only (Sumber : https://medium.com/thelocalminima/are-bert-features-interbertible-250a91eb9dc).....	12
Gambar 2.10 Arsitektur GPT yang merupakan Transformer Decoder Only (Sumber : https://huggingface.co/docs/transformers/tasks_explained).....	13
Gambar 2.11 Contoh Antarmuka Tkinter (Sumber : GeeksForGeeks)	14
Gambar 3.1 Diagram Alir Pembuatan Model	15
Gambar 3.2 Visualisasi keypoint yang diekstrak (warna merah) dan keypoint yang dinormalisasi (warna biru).....	16
Gambar 3.3 Visualisasi Proses Augmentasi	16
Gambar 3.4 Diagram Alir Aplikasi Pembelajaran Bahasa Isyarat.....	17
Gambar 3.5 Contoh Dataset dengan 3 Pemeraga pada Class “Akan”	18
Gambar 3.6 Visualisasi layer model	23
Gambar 4.1 (a) Grafik Akurasi Train dan Validation, (b) Grafik Train dan Validation Loss dari Skenario Uji Coba 1	28
Gambar 4.2 (a) Grafik Akurasi Train dan Validation, (b) Grafik Train dan Validation Loss dari Skenario Uji Coba 2	28
Gambar 4.3 (a) Grafik Akurasi Train dan Validation, (b) Grafik Train dan Validation Loss dari Skenario Uji Coba 3	29
Gambar 4.4 (a) Grafik Akurasi Train dan Validation, (b) Grafik Train dan Validation Loss dari Skenario Uji Coba 4	30
Gambar 4.5 (a) Grafik Akurasi Train dan Validation, (b) Grafik Train dan Validation Loss dari Skenario Uji Coba 5	30
Gambar 4.6 (a) Grafik Akurasi Train dan Validation, (b) Grafik Train dan Validation Loss dari Skenario Uji Coba 6	31
Gambar 4.7 (a) Grafik Akurasi Train dan Validation, (b) Grafik Train dan Validation Loss dari Skenario Uji Coba 7	32
Gambar 4.8 (a) Grafik Akurasi Train dan Validation, (b) Grafik Train dan Validation Loss dari Skenario Uji Coba 8	32
Gambar 4.9 Tampilan Antarmuka Menu Utama Aplikasi Pembelajaran Bahasa Isyarat.....	33
Gambar 4.10 Tampilan Antarmuka Menu Modul Belajar Aplikasi Pembelajaran Bahasa Isyarat	34
Gambar 4.11 Jendela Pemutar Video Peragaan Bahasa Isyarat.....	34
Gambar 4.12 Tampilan Antarmuka Menu Modul Evaluasi Aplikasi Pembelajaran Bahasa Isyarat	35
Gambar 4.13 Tampilan Jendela Proses Evaluasi Aplikasi Pembelajaran Bahasa Isyarat	35
Gambar 4.14 Tampilan Jendela Hasil Evaluasi Aplikasi Pembelajaran Bahasa Isyarat	36
Gambar 4.15 Tampilan Jendela Latihan Aplikasi Pembelajaran Bahasa Isyarat	36

Gambar 4.16 Perbandingan Kinerja Antar Model.....	38
Gambar 4.17 Kondisi Optimal Deteksi Keypoints.....	38
Gambar 4.18 Hasil Prediksi Kondisi Optimal.....	39
Gambar 4.19 Kondisi Tidak Optimal Deteksi Keypoints	39
Gambar 4.20 Hasil Prediksi Kondisi Tidak Optimal.....	40
Gambar 4.21 (a) Video peraga class Tiga, (b) Video peraga class “Dia”	40
Gambar 4.22 (a) Hasil prediksi peraga class Tiga, (b) Hasil prediksi peraga class “Dia”	40

DAFTAR TABEL

Tabel 2.1 Perbandingan Penelitian Terkait	3
Tabel 3.1 Daftar Kata yang Digunakan	17
Tabel 3.2 Spesifikasi Peralatan Pendukung	18
Tabel 4.1 Tabel Skenario dan Hasil Uji Coba Dengan Data Augmentasi	27
Tabel 4.2 Tabel Skenario dan Hasil Uji Coba Tanpa Data Augmentasi	37

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SEMU

Kode Semu 3.1	Kode Semu Pelabelan Video	19
Kode Semu 3.2	Kode Semu Ekstraksi Fitur	19
Kode Semu 3.3	Kode Semu Normalisasi Data	19
Kode Semu 3.4	Kode Semu Augmentasi Data	20
Kode Semu 3.5	Kode Semu Train dan Evaluasi Model	20
Kode Semu 3.6	Kode Semu Modul Belajar.....	21
Kode Semu 3.7	Kode Semu Modul Evaluasi	21

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

Kode Sumber 3.1 Arsitektur Model.....	23
Kode Sumber 3.2 Fungsi Utama pada Modul Belajar	25
Kode Sumber 3.3 Fungsi Utama pada Modul Evaluasi	26

[Halaman ini sengaja dikosongkan]

BAB I PENDAHULUAN

1.1 Latar belakang

Manusia merupakan makhluk ciptaan tuhan yang berakal. Dalam kehidupan sehari-hari, manusia berkomunikasi satu sama lain untuk menyampaikan informasi. Komunikasi umumnya disampaikan melalui lisan, tulisan, serta visual. Akan tetapi, manusia tidak sempurna, beberapa memiliki kekurangan yang menghambat untuk berkomunikasi dengan sesama. Untuk mengatasi hambatan tersebut diciptakan metode lain untuk menggantikan cara komunikasi umum seperti dengan menggunakan bahasa isyarat. Masyarakat pada umumnya kesulitan dalam menggunakan bahasa isyarat karena tidak terbiasa. Untuk mengatasi permasalahan tersebut diperlukan adanya media pembelajaran bahasa isyarat.

Bahasa isyarat adalah bahasa yang menggunakan gestur anggota badan – pada umumnya tangan – untuk menyampaikan maksud. Bahasa isyarat sendiri memiliki standar yang berbeda, sesuai dengan bahasa daerah sekitarnya. Di Indonesia, standar baku bahasa isyarat tercantum dalam Standar Isyarat Bahasa Indonesia (SIBI). Pembakuan bahasa isyarat bertujuan untuk mempermudah penggunaan bahasa isyarat sehingga makna yang disampaikan tepat. SIBI memberikan panduan dan petunjuk sistematis dalam menggunakan bahasa isyarat yang baku dalam bentuk buku. Namun pembelajaran dengan media buku kurang efektif karena gambar pada buku tidak dapat memberikan contoh riil seperti praktek langsung. Selain itu buku kurang praktis untuk dibawa karena memiliki halaman tebal.

Untuk itu diperlukan sebuah sarana aplikasi yang dapat membantu dan memudahkan pembelajaran bahasa isyarat bagi masyarakat umum atau penyandang tunarungu. Diperlukan juga pembelajaran yang interaktif sehingga peserta belajar lebih mudah dalam memahami. Salah satu sarana yang memenuhi syarat tersebut adalah gim, yang menyediakan interaksi dengan peserta ajar dan memberikan umpan balik secara *real-time*.

Hingga saat ini sudah ada beberapa aplikasi yang dikembangkan untuk memudahkan pembelajaran bahasa isyarat. Salah satunya adalah Aplikasi Bahasa Isyarat untuk Tuna Rungu yang menggunakan platform android. Aplikasi ini menyediakan materi dan evaluasi pembelajaran bahasa isyarat dengan media ponsel. Adapun materi yang disajikan dalam bentuk gambar dan video. Untuk kategori materi bahasa isyarat yang tersedia antara lain abjad, angka, benda-benda umum, serta kelompok kata. Modul evaluasi pada aplikasi berupa tebak gambar dan tebak video, yang mana pengguna aplikasi akan memilih gambar atau video yang sesuai dengan soal yang diberikan, atau sebaliknya (Mandarani et al, 2020).

Aplikasi Pembelajaran Bahasa Isyarat (PemBais), adalah aplikasi pembelajaran bahasa isyarat serupa yang memberikan materi gambar dan video penggunaan bahasa isyarat juga telah dikembangkan. Materi pada PemBais memiliki beberapa kategori seperti kosa kata, angka, abjad, serta video gerakan bahasa isyarat. Aplikasi ini tidak menyajikan modul evaluasi bagi pengguna (Pradikja et al, 2019).

Dari contoh yang diberikan, aplikasi pembelajaran bahasa isyarat dapat menyampaikan pembelajaran melalui gambar dan video. Namun pembelajaran bahasa isyarat tersebut belum dapat memberikan evaluasi bagi pengguna tentang ketepatan penggunaan bahasa isyarat. Salah satu cara untuk mengevaluasi ketepatan tersebut adalah dengan memberikan aplikasi pembelajaran kemampuan untuk mengenali bahasa isyarat.

Dalam mengenali bahasa isyarat, solusi yang dapat digunakan adalah *deep learning*. *Convolutional Neural Network* (CNN) merupakan salah satu metode *deep learning* yang umum digunakan ketika *dataset* berupa gambar statis (Suharjito et al, 2018). Jika *dataset* berupa video atau urutan gambar yang saling terhubung, *Recurrent Neural Network* (RNN) dapat menghasilkan akurasi yang lebih baik (Lee et al, 2021).

Pada pengembangan aplikasi pembelajaran bahasa isyarat ini, metode *deep learning* yang

akan digunakan untuk mengenali bahasa isyarat adalah arsitektur transformer. Arsitektur tersebut memiliki kelebihan dalam mengolah data sekuensial seperti RNN. Namun, transformer dapat mengolah semua *input* dalam satu waktu yang bersamaan.

1.2 Rumusan Permasalahan

Berikut adalah beberapa hal yang menjadi rumusan masalah dalam tugas akhir ini:

1. Bagaimana rancangan model pengenalan bahasa isyarat dengan arsitektur transformer?
2. Bagaimana rancangan aplikasi pembelajaran bahasa isyarat yang memiliki modul belajar dan evaluasi?
3. Bagaimana rancangan modul evaluasi pada aplikasi pembelajaran bahasa isyarat dengan mempertimbangkan kesesuaian gestur tangan?
4. Bagaimana rancangan modul belajar pada aplikasi pembelajaran bahasa isyarat?
5. Bagaimana cara mengevaluasi aplikasi pembelajaran bahasa isyarat?

1.3 Batasan Masalah

Batas permasalahan yang dibahas dalam tugas akhir sebagai berikut:

1. Aplikasi yang akan dibuat dijalankan pada laptop atau komputer dengan sistem operasi Windows.
2. Target pengguna dari aplikasi pembelajaran ini adalah masyarakat umum dan penyandang tunarungu umur 10 hingga 35 tahun.
3. Bahasa isyarat menggunakan Standar Isyarat Bahasa Indonesia (SIBI).
4. Bahasa pemrograman yang digunakan dalam pengembangan model dan aplikasi yaitu Python, dengan *library* pendukung seperti OpenCV, MediaPipe dan Tensorflow.
5. Modul evaluasi mandiri hanya menerima masukan isyarat per kata, bukan kalimat utuh.
6. Kondisi uji coba *real time* menggunakan jarak pemeraga dan kamera diantara 0.75-1 meter, pencahayaan merata dengan cahaya latar belakang tidak terlalu terang, serta resolusi minimal kamera 720p.

1.4 Tujuan

Berikut adalah tujuan dari pembuatan tugas akhir ini:

1. Merancang sebuah model pengenalan bahasa isyarat dengan arsitektur transformer.
2. Merancang sebuah aplikasi pembelajaran bahasa isyarat yang menyediakan modul belajar dan modul evaluasi.
3. Merancang modul evaluasi aplikasi pembelajaran bahasa isyarat yang mempertimbangkan kesesuaian gestur tangan.
4. Merancang modul belajar pada aplikasi pembelajaran bahasa isyarat.
5. Menentukan cara-cara evaluasi untuk mengukur keberhasilan aplikasi pembelajaran bahasa isyarat yang telah dirancang.

1.5 Manfaat

Manfaat dari hasil pembuatan tugas akhir ini antara lain:

1. Memberikan sarana praktek penggunaan bahasa isyarat.
2. Memberikan pengalaman baru dalam belajar bahasa isyarat bagi pengguna.
3. Mengasah kemampuan pengguna dalam menggunakan dan mengenal bahasa isyarat.

BAB II TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Dalam pengenalan bahasa isyarat atau *sign language recognition* (SLR) alat penangkap *input* yang biasa digunakan adalah kamera. Selain kamera, ada cara lain untuk mendapatkan *input* seperti dengan menggunakan *accelerometer*, sarung tangan yang dilengkapi sensor gerak, serta Microsoft Kinect. Dengan penggunaan kamera sederhana seperti kamera ponsel atau *webcam*, *input* yang dihasilkan berupa video yang memiliki *frame rate* 30 FPS. Sedangkan dengan penggunaan Kinect, dapat memberikan data *color*, *depth* dan *skeleton* (Suharjito et al, 2017).

Pada saat pemrosesan data, metode umum yang digunakan adalah *Hidden Markov Model* (HMM) yang digabungkan dengan algoritma lain seperti *Fuzzy K-Means* untuk melakukan klasifikasi bentuk tangan (Suharjito et al, 2017). Selain HMM, *deep learning* juga digunakan dalam pemrosesan data. *Convolutional Neural Network* merupakan salah satu jenis neural network yang umum digunakan pada data citra. CNN terdiri dari *neuron* yang memiliki bobot, bias, dan fungsi aktivasi. Secara umum, CNN terdiri dari: dua bagian, yaitu *feature extraction layer* dan *fully connected layer* (Daniels et al, 2021).

Dalam merancang tugas akhir, diperlukan adanya referensi untuk membantu pengerjaan. Tabel 2.1 berisikan penelitian yang terkait dengan tugas akhir. Adapun isi dari tabel seperti nama penelitian, dataset dan spesifikasinya, serta kelebihan dan kekurangan dari penelitian tersebut.

Tabel 2.1 Perbandingan Penelitian Terkait

No.	Peneliti	Keterangan	Kelebihan	Kekurangan
1	Suharjito. Gunawan, Herman Thiracitta, Narada. Nugroho, Ariadi, 2018	<i>Dataset</i> LSA64, 10 <i>vocabulary</i> , 10 pemeraga, 5 kali percobaan per <i>vocabulary</i> (500 video)	Metode pengolahan data dan pengujian dijelaskan dengan detail. Hasil akurasi model yang tinggi dengan sedikit <i>class</i> .	Akurasi model menurun dengan <i>class</i> yang banyak.
2	C.K.M.Lee, Kam K.H.Ng, Chun-Hsien Chen, H.C.W.Lau, S.Y.Chung, Tiffany Tsoi., 2021	26 data alfabet, 100 sampel per alfabet (2600 data)	Penggunaan RNN untuk 26 alfabet memiliki performa rata-rata yang lebih tinggi dibandingkan dengan metode lain dalam <i>cross</i> <i>validation</i> .	Menggunakan <i>Leap Motion</i> <i>Controller</i> untuk mendeteksi gestur. Aplikasi pembelajaran bahasa isyarat yang akan dikembangkan tidak menggunakan <i>Leap Motion</i> <i>Controller</i> .

No.	Peneliti	Keterangan	Kelebihan	Kekurangan
3	Neto, G. M. R., Junior, G. B., de Almeida, J. D. S., & de Paiva, A. C., 2018	<i>Dataset</i> LSA64, 3200 video, 10 pemeraga, 64 <i>class</i>	Arsitektur model 3D CNN dapat mengenali 64 <i>class</i> bahasa isyarat dengan akurasi 93.9%.	Variasi data yang kurang beragam.
4	Camgoz, N. C., Koller, Oscar. Hadfield, Simon. Bowden, Richard, 2020	<i>Dataset</i> RWTH- PHOENIX -Weather-2014T (PHOENIX14T), 1066 <i>class</i> , 9 pemeraga	Pengenalan dan penerjemahan bahasa isyarat menggunakan <i>transformer</i> dengan skor terbaik 21.80 BLEU-4.	Ekstraksi fitur untuk wajah, tangan dan badan belum ada.
5	Fang, Biyi. Co, Jillian. Zhang, Mi, 2018	<i>Dataset</i> kata ASL, 56 <i>class</i> kata dan 100 <i>class</i> kalimat, 11 pemeraga, 7306 sampel	Memiliki akurasi 94.5% pada tingkat pengenalan kata. <i>Error rate</i> pada tingkat pengenalan kalimat adalah 8.2% dengan kalimat yang belum pernah dikenali.	Data yang digunakan belum mencakup percakapan yang umum digunakan sehari-hari.
6	Krishna, Akhil G., Sandeep, G., 2022	Menggunakan <i>dataset</i> bahasa isyarat standar India (ISL), 200 <i>class</i> , 5 variasi <i>view angle</i>	Hasil akurasi model yang tinggi yaitu 90%. Model sesuai untuk tipe data yang sekuensial.	Dengan menerapkan Distance measure classifier (DMC) akurasi prediksi menurun.
7	Ko, S.-K., Son, J. G., & Jung, H., 2018	<i>Dataset</i> KETI, 10840 video, Resolusi 1920x1080, 30 fps	Video diperagakan oleh ahli bahasa isyarat Korea. Hasil akurasi akhir yang tinggi sebesar 89.5%.	Model bergantung pada <i>dataset</i> yang terbatas dengan penutur bahasa isyarat profesional dalam lingkungan tertentu.

No.	Peneliti	Keterangan	Kelebihan	Kekurangan
8	Daniels, Steve. Suciati, Nanik. Fathichah, Chastine, 2021	<i>Dataset</i> yang digunakan diambil secara manual menggunakan standar BISINDO.	Arsitektur model CNN menggunakan YOLO. Skor akurasi sebesar 93.1%, presisi 77.14% dan <i>recall</i> 72.97%.	Model mengalami kesulitan untuk mengenali isyarat pada saat perpindahan atau pergerakan antar gestur.
9	Mandarani, Putri. Putra, Yogi, 2020	Aplikasi pembelajaran bahasa isyarat <i>mobile</i> untuk tunarungu dan tunawicara.	Modul pembelajaran bahasa isyarat menggunakan video peraga. Modul ujian/evaluasi memberikan <i>feedback</i> langsung melalui skor benar dan salah.	Modul evaluasi masih berupa gambar, bukan video seperti modul pembelajaran.
10	Pradikja, Maharoni Hendra. Tolle, Herman. Brata, Kumang Candra, 2019	Aplikasi pembelajaran bahasa isyarat <i>mobile</i> untuk tunarungu dan tunawicara.	Memiliki modul belajar dengan materi kosa kata, angka, abjad, serta video peraga bahasa isyarat.	Tidak memiliki modul evaluasi untuk mengukur pemahaman bahasa isyarat.

2.2 Dasar Teori

Dasar teori merupakan landasan utama dalam merancang tugas akhir. Dasar teori berisikan penjelasan lebih lanjut mengenai teori, alat, serta *library* yang akan digunakan. Pada sub bab 2.2 akan menjelaskan mengenai *deep learning*, *Convolutional Neural Network*, *Recurrent Neural Network*, *Transformer* dan variasinya, TensorFlow, OpenCV, MediaPipe, serta Tkinter.

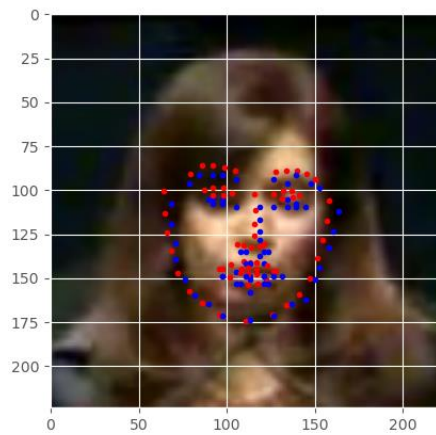
2.2.1 Ekstraksi Fitur

Ekstraksi fitur adalah proses mengubah data mentah menjadi representasi yang lebih sederhana dan bermakna yang dapat digunakan untuk mempelajari pola dan menjalankan tugas tertentu. Dalam bahasa Indonesia, ekstraksi fitur dapat diartikan sebagai "pengambilan fitur" atau "ekstraksi ciri."

Ekstraksi fitur adalah langkah penting dalam analisis data dan pembelajaran mesin. Tujuannya adalah untuk mengurangi dimensi data, mengidentifikasi pola yang relevan, dan membuat data lebih mudah dipahami oleh algoritma pembelajaran mesin. Dengan representasi fitur yang tepat, model dapat mempelajari informasi yang relevan dan menghasilkan hasil yang lebih baik.

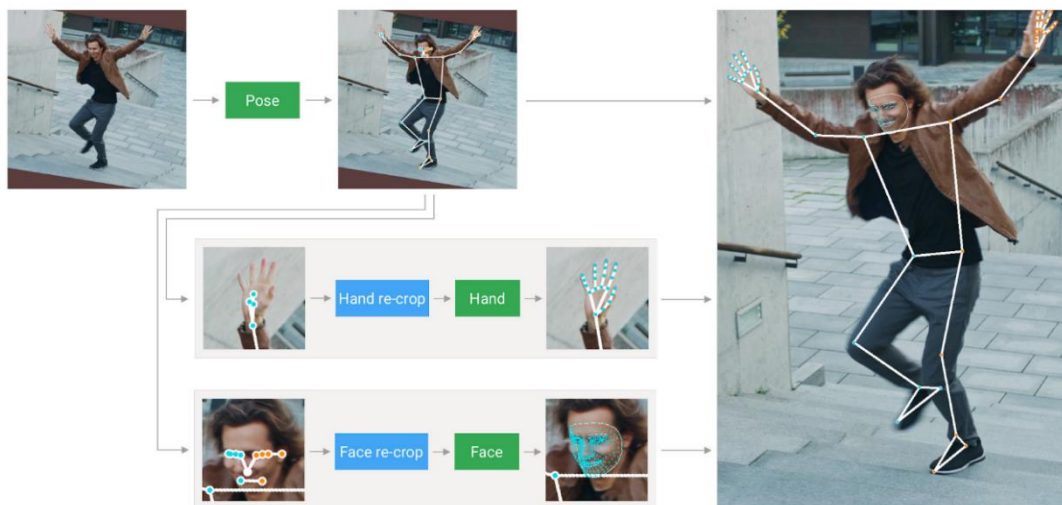
Fitur dalam konteks ini merujuk pada atribut atau karakteristik tertentu yang menggambarkan data. Misalnya, dalam pengenalan tulisan tangan, fitur-fitur mungkin mencakup ukuran huruf, bentuk garis, sudut, dll. Dimensi data merujuk pada jumlah fitur yang ada dalam setiap sampel data. Ekstraksi fitur membantu mengurangi dimensi data dengan menghilangkan fitur yang kurang relevan atau redundan. Terdapat berbagai metode untuk ekstraksi fitur, tergantung pada jenis data dan tujuan analisis. Beberapa teknik umum meliputi ekstraksi fitur statistik, transformasi Fourier, reduksi dimensi, ekstraksi *keypoints* dan metode

domain khusus. Dalam tugas akhir ini metode ekstraksi fitur yang digunakan adalah ekstraksi keypoints.



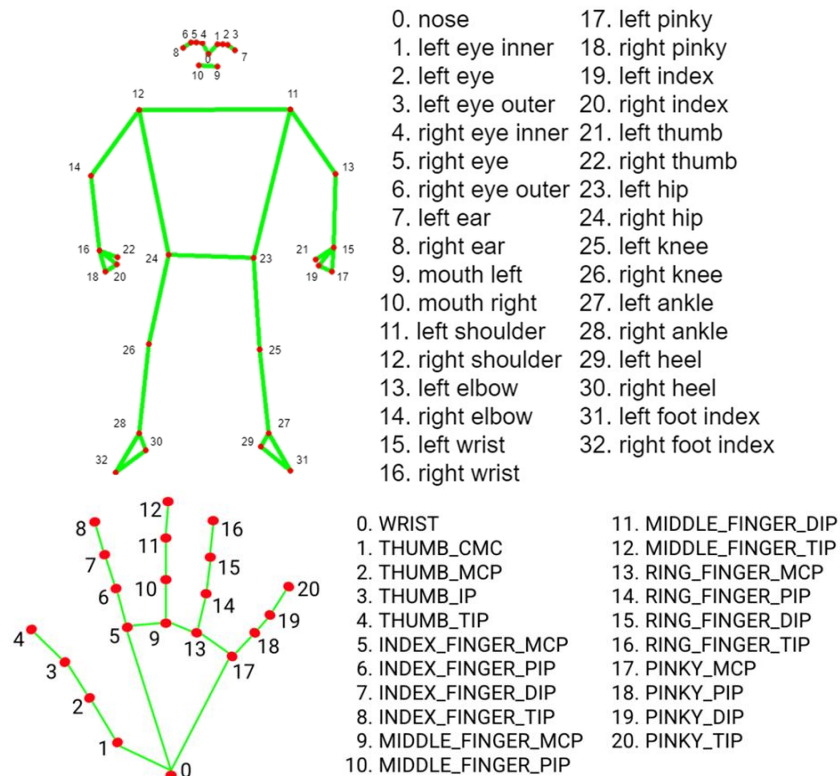
Gambar 2.1 Contoh Ekstraksi Fitur *Keypoints* (Sumber : DebuggerCaffe)

Ekstraksi fitur *keypoints* (titik kunci) adalah proses untuk mengidentifikasi dan mengekstrak lokasi atau posisi titik-titik penting dalam sebuah gambar atau citra. Titik-titik kunci ini sering kali mewakili fitur-fitur yang penting atau menonjol dari objek yang ada di dalam gambar. Fitur-fitur ini dapat berupa sudut, ujung, titik pusat, atau bagian-bagian lain yang memiliki informasi penting untuk tujuan analisis lebih lanjut, seperti pengenalan objek, pelacakan objek, atau analisis pola.



Gambar 2.2 MediaPipe Holistic Solution (Sumber : MediaPipe)

Dalam mengekstraksi *keypoints* terdapat beberapa *tools* yang dapat digunakan, salah satunya adalah MediaPipe. MediaPipe adalah *framework open-source* yang digunakan untuk mendeteksi objek secara *real-time*. MediaPipe menyediakan solusi untuk berbagai masalah deteksi objek seperti *tracking*, *segmentation*, serta *object detection*. Salah satu solusi MediaPipe yakni Holistic, memungkinkan pengguna untuk menggabungkan beberapa model *tracking* menjadi kesatuan yang utuh seperti kerangka. Pada kerangka tersebut terdapat *keypoints* yang dapat diekstrak posisinya dalam bentuk koordinat tiga dimensi (koordinat XYZ).



Gambar 2.3 Landmark yang ada pada bagian tubuh dan telapak tangan (Sumber : MediaPipe)

2.2.2 Augmentasi Data

Augmentasi data adalah teknik yang umum digunakan dalam pembelajaran mesin untuk meningkatkan jumlah data pelatihan dengan membuat variasi dari data yang ada. Tujuannya adalah untuk meningkatkan performa dan generalisasi model.

Augmentasi data bertujuan untuk memperluas keragaman dataset pelatihan tanpa mengumpulkan data tambahan secara manual. Dengan menciptakan variasi dari data yang ada, model pembelajaran mesin dapat belajar dari berbagai sudut pandang dan situasi, sehingga dapat meningkatkan kemampuan model untuk mengenali pola yang lebih umum dan menggeneralisasi dengan baik pada data yang belum pernah dilihat sebelumnya. Terdapat berbagai teknik augmentasi data yang dapat diterapkan, tergantung pada jenis data dan jenis tugas pembelajaran mesin yang sedang dihadapi. Beberapa contoh teknik augmentasi data adalah rotasi, *cropping*, *shifting*, *scaling*, perubahan *brightness*, *scaling*, penambahan *noise* dan transformasi geometris.

Augmentasi data sering digunakan ketika jumlah data pelatihan terbatas atau tidak seimbang (*imbalance*). Dengan mengaplikasikan augmentasi data pada kelas minoritas, dapat membantu mengurangi masalah ketidakseimbangan kelas dalam dataset. Selain itu, augmentasi data juga berguna dalam tugas-tugas pengenalan pola, visi komputer, pemrosesan bahasa alami, dan berbagai bidang pembelajaran mesin lainnya.

Meskipun augmentasi data sangat bermanfaat, augmentasi tidak menyebabkan perubahan mendasar pada sifat data atau menghasilkan data yang tidak realistis. Augmentasi yang tidak sesuai dapat menyebabkan model menghasilkan prediksi yang tidak relevan atau tidak akurat pada data asli.

2.2.3 Metrik Evaluasi

Metrik evaluasi kinerja model *deep learning* adalah ukuran-ukuran yang digunakan untuk mengukur seberapa baik model tersebut melakukan tugasnya, seperti klasifikasi atau

regresi, pada data yang belum pernah dilihat sebelumnya. Metrik evaluasi *accuracy* (akurasi) dan *precision* (presisi) adalah dua ukuran kinerja yang umum digunakan dalam tugas klasifikasi, seperti pada model *deep learning*. Kedua metrik ini memberikan informasi yang berbeda mengenai kualitas prediksi model terhadap data uji.

		Nilai Sebenarnya	
		Positif	Negatif
Nilai Prediksi	Positif	<i>True Positive (TP)</i>	<i>False Positive (FP)</i>
	Negatif	<i>False Negative (FN)</i>	<i>True Negative (TN)</i>

Gambar 2.4 *Confusion Matrix* untuk mengevaluasi kebenaran prediksi

Akurasi adalah metrik yang mengukur sejauh mana model dapat memprediksi dengan benar pada semua kelas. Akurasi memberikan gambaran tentang seberapa baik model secara keseluruhan dalam melakukan klasifikasi. Namun, akurasi bisa menyesatkan jika data memiliki ketidakseimbangan kelas, yaitu ketika salah satu kelas lebih dominan daripada yang lain. Pada kasus ketidakseimbangan, akurasi yang tinggi mungkin terjadi karena model cenderung memprediksi mayoritas kelas, tetapi performa sebenarnya bisa buruk untuk kelas minoritas yang mungkin lebih penting. Secara matematis, akurasi didefinisikan sebagai jumlah prediksi yang benar (*true positives* dan *true negatives*) dibagi dengan jumlah total sampel data.

$$\text{Akurasi} = \frac{TP+TN}{TP + FP + TN+ FN} \quad (2.1)$$

Presisi adalah metrik yang mengukur sejauh mana prediksi positif model adalah benar. Presisi menghitung berapa persen prediksi positif yang benar (*true positives*) dari total prediksi positif yang dibuat oleh model (*true positives* dan *false positives*).

$$\text{Presisi} = \frac{TP}{TP + FP} \quad (2.2)$$

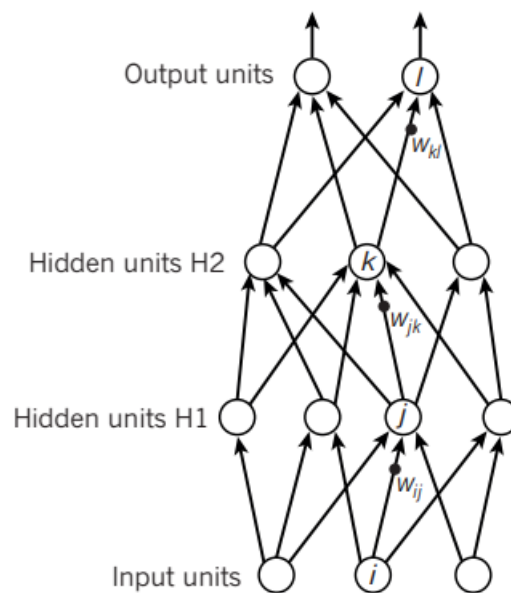
2.2.4 Deep Learning

Deep learning atau *deep structured learning* adalah bagian dari metode *machine learning* dengan basis *neural network* (jaringan saraf) buatan. *Deep learning* memungkinkan model yang terdiri dari beberapa lapisan pemrosesan untuk mempelajari representasi data dengan berbagai tingkat abstraksi. Metode ini menemukan struktur rumit dalam *data set* besar dengan menggunakan algoritma *backpropagation* untuk menunjukkan bagaimana mesin harus mengubah parameter yang digunakan untuk menghitung representasi di setiap lapisan dari representasi di lapisan sebelumnya (LeCun et al, 2015).

Metode *deep learning* adalah metode *representation-learning* dengan beberapa tingkat representasi, diperoleh dengan menyusun modul non-linier sederhana yang masing-masing mengubah representasi pada satu tingkat (dimulai dengan *input* mentah) menjadi representasi pada tingkat yang lebih tinggi, dengan sedikit tambahan abstrak. Dengan komposisi transformasi yang cukup, fungsi yang sangat kompleks bisa dipelajari. Untuk tugas klasifikasi,

lapisan representasi yang lebih tinggi memperkuat aspek *input* yang penting untuk membedakan dan menekan variasi yang tidak relevan. Sebagai contoh, sebuah gambar muncul dalam bentuk *array* nilai *pixel*, dan fitur yang dipelajari di bagian pertama lapisan representasi biasanya mewakili ada atau tidak adanya tepi pada orientasi dan lokasi tertentu dalam gambar. Kedua lapisan biasanya mendeteksi motif dengan melihat pengaturan tertentu dari tepi, terlepas dari variasi kecil dalam posisi tepi. Ketiga lapisan dapat merakit motif menjadi kombinasi yang lebih besar yang sesuai ke bagian objek yang sudah dikenal, dan lapisan berikutnya akan mendeteksi objek sebagai kombinasi dari bagian-bagian ini (LeCun et al, 2015).

Arsitektur *deep learning* seperti *deep neural network* (DNN), *recurrent neural network* (RNN), *convolutional neural network* (CNN), dan *transformer* telah diterapkan di berbagai bidang seperti citra computer, pengenalan suara, serta *natural language processing* (NLP).

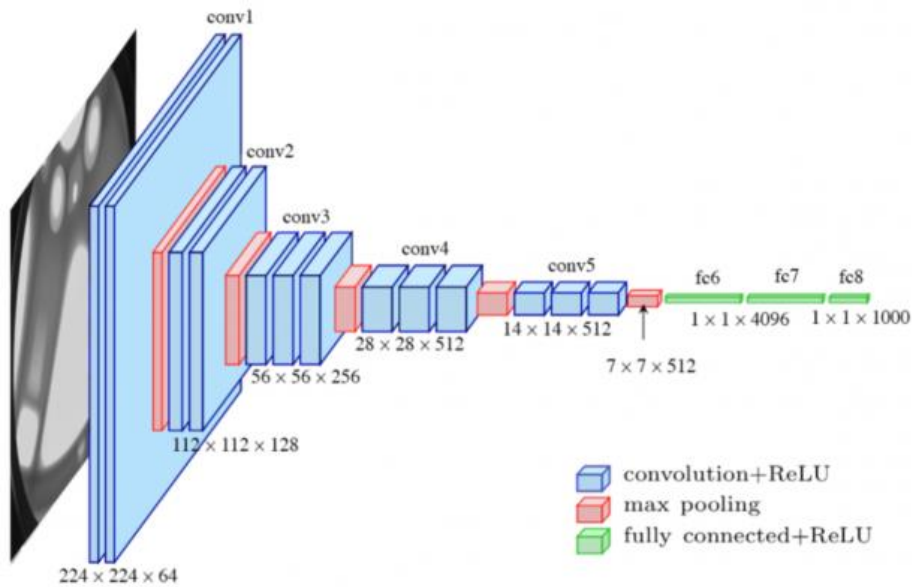


Gambar 2.5 Arsitektur Deep Learning dengan Multilayer Neural Network (LeCun et al, 2015)

2.2.5 CNN

CNN (*Convolutional Neural Network* atau *ConvNet*) adalah salah satu arsitektur *deep learning*. CNN merupakan versi regular dari *multilayer perceptron* yang merupakan jaringan yang terhubung penuh. Pada *multilayer perceptron* setiap *neuron* dalam satu lapisan terhubung dengan semua *neuron* yang ada di lapisan berikutnya (Valueva et al, 2020).

Sebuah *convolutional neural network* terdiri atas *input layer*, *hidden layer*, dan *output layer*. Pada *feed-forward neural network*, setiap lapisan tengah disebut sebagai lapisan tersembunyi (*hidden layer*) karena *input* dan *output* ditutupi oleh fungsi aktivasi dan konvolusi akhir. Dalam *convolutional neural network*, lapisan tersembunyi berisi lapisan yang melakukan konvolusi. Umumnya, lapisan ini mencakup lapisan yang melakukan *dot product* dari kernel konvolusi dengan *input matrix* lapisan. Saat kernel konvolusi berjalan di sepanjang *input matrix* pada lapisan, operasi konvolusi menghasilkan peta fitur, yang kemudian berkontribusi pada *input layer* berikutnya. Hal ini diikuti oleh lapisan lain seperti *pooling layer*, *fully connected layer*, dan *normalization layer*.

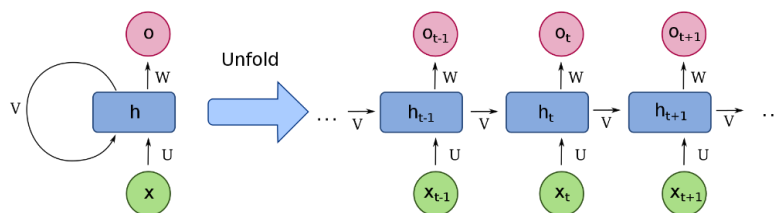


Gambar 2.6 Arsitektur CNN VGG-16 (Le, 2017)

2.2.6 RNN

Recurrent Neural Network adalah bagian dari *deep learning* yang memiliki *self-connected hidden layer*. Salah satu kelebihan dari RNN adalah "memori" atau ingatan dari *input* di masa lalu tersimpan dalam *network internal state*. Dari "memori" tersebut RNN dapat memanggil atau mendapatkan konteks *input* masa lalu. Kelebihan lain yang dimiliki RNN, tingkat perubahan (*rate of change*) dari *internal state* dapat diatur melalui *recurrent weights*. Tingkat perubahan yang modular tersebut memberikan kekokohan terhadap distorsi lokal dari *input* (Graves et al, 2009).

RNN merupakan jaringan node mirip neuron yang diatur ke dalam "lapisan" yang berurutan. Setiap node dalam lapisan tertentu terhubung dengan koneksi terarah ke setiap node lain di lapisan berikutnya. Setiap node (neuron) memiliki aktivasi bernilai riil yang bervariasi waktu. Setiap koneksi (sinaps) memiliki bobot bernilai riil yang dapat dimodifikasi. Node adalah node input (menerima data dari luar jaringan), node output (menghasilkan hasil), atau node tersembunyi (yang mengubah data dalam perjalanan dari input ke output).



Gambar 2.7 Arsitektur Umum RNN Terkompresi (kiri) dan Tidak Terkompresi (kanan)
(Sumber : wikipedia.org/wiki/Recurrent_neural_network)

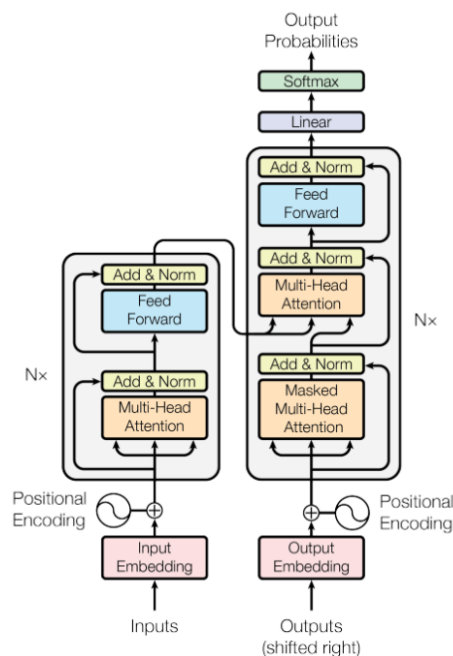
2.2.7 Transformer

Transformer merupakan implementasi mekanisme *self-attention* pada sebuah model *deep learning*, yang dapat memfokuskan bagian *input* yang diberikan. Model transformer memiliki kesamaan dengan *Recurrent Neural Network* (RNN) yang didesain untuk memproses *input* data sekuensial. Perbedaan yang dimiliki oleh transformer dengan model RNN adalah

model transformer dapat memproses semua *input* secara bersamaan. Mekanisme *attention* yang diberikan pada transformer memungkinkan model untuk mendapatkan konteks dari setiap bagian *input*. Sebagai contoh, ketika model diberikan *input* berupa sebuah kalimat bahasa alami, transformer akan memproses seluruh kalimat secara utuh (Vaswan, 2017).

Transformer terdiri dari dua komponen utama: satu set *encoder* yang terhubung dan satu set *decoder* yang terhubung. Fungsi setiap *encoder* adalah memproses vektor inputnya untuk menghasilkan *encoding*, yang berisi informasi tentang bagian-bagian input yang relevan satu sama lain. *Encoding* ini melewati set pengkodean yang dihasilkan ke *encoder* berikutnya sebagai *input*. Setiap *decoder* melakukan yang sebaliknya, mengambil semua pengkodean dan memprosesnya, menggunakan informasi kontekstual yang digabungkan untuk menghasilkan sekuens *output*. Untuk mencapai hal ini, setiap *encoder* dan *decoder* menggunakan mekanisme *self-attention*, yang untuk setiap input, menimbang relevansi setiap input dan menarik informasi yang sesuai saat menghasilkan output. Setiap *decoder* juga memiliki mekanisme *attention* tambahan yang menarik informasi dari *output decoder* sebelumnya, sebelum *decoder* mengambil informasi dari pengkodean. Baik *encoder* dan *decoder* memiliki *feed-forward neural network* akhir untuk pemrosesan *output* tambahan, dan juga berisi koneksi residual dan langkah normalisasi lapisan (Vaswan, 2017).

Terdapat beberapa variasi arsitektur Transformer yang telah dikembangkan untuk berbagai tugas pemrosesan bahasa alami dan komputasi visual. Salah satunya adalah arsitektur Transformer *Encoder-Decoder*, yang mencakup *encoder* dan *decoder*. *Encoder* bertanggung jawab untuk mengambil input teks atau gambar dan menghasilkan representasi kontekstual, sementara *decoder* digunakan untuk menghasilkan output berdasarkan representasi tersebut. Selain itu, ada juga variasi arsitektur seperti *Encoder Only*, yang mengadopsi struktur hanya dengan menggunakan blok *encoder*, di mana setiap blok terdiri dari lapisan *multi-head attention* dan lapisan *feed-forward*.



Gambar 2.8 Arsitektur Transformer (Vaswan, 2017).

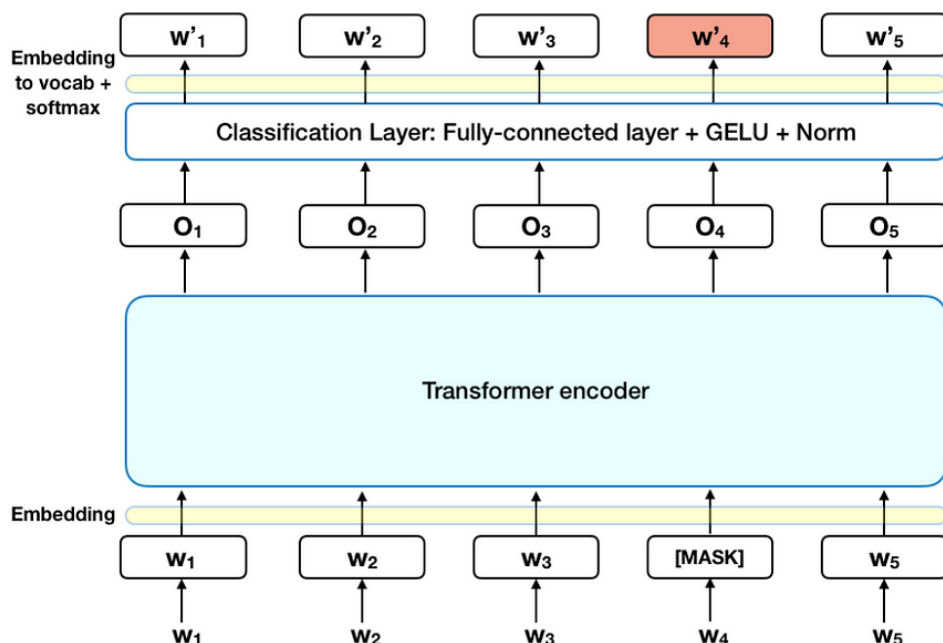
2.2.4.1 Encoder-Decoder

Variasi arsitektur *Encoder-Decoder* adalah pendekatan yang terdiri dari dua bagian utama, yakni *encoder* dan *decoder*. *Encoder* bertanggung jawab untuk mengambil *input* data dan menghasilkan representasi yang lebih abstrak dan kontekstual. Setelah representasi tersebut

terbentuk, *decoder* menggunakan representasi tersebut untuk menghasilkan *output* yang sesuai dengan tugas tertentu, seperti penerjemahan bahasa, generasi teks, atau pengenalan gambar. Tujuan utama dari arsitektur *Encoder-Decoder* adalah memfasilitasi pemahaman yang lebih baik dari data *input*, dan kemudian menghasilkan *output* yang lebih tepat dan relevan. Pendekatan ini telah terbukti sangat efektif dalam berbagai tugas pemrosesan bahasa alami dan komputasi visual, karena memungkinkan model untuk memahami hubungan kompleks antara input dan output serta mengatasi masalah tugas yang berbeda dengan satu arsitektur yang sama. Selain itu, arsitektur *Encoder-Decoder* juga memungkinkan untuk melatih model secara *end-to-end*, yang dapat meningkatkan efisiensi dan kualitas performa dalam tugas-tugas pemrosesan data yang kompleks dan berdimensi tinggi. Gambar 2.4 adalah arsitektur Transformer variasi *encoder-decoder*.

2.2.4.2 Encoder Only

Variasi arsitektur *Encoder-Only* merupakan pendekatan yang hanya menggunakan bagian *encoder* dari arsitektur Transformer. Dalam model ini, *input* data diolah melalui serangkaian lapisan *encoder*, yang terdiri dari mekanisme *self-attention* dan jaringan *feed-forward*. *Encoder* bertanggung jawab untuk mengambil data masukan, menganalisisnya secara mendalam, dan menghasilkan representasi yang lebih abstrak dan kontekstual dari data tersebut. Tujuan utama dari variasi *Encoder-Only* ini adalah untuk menghasilkan representasi yang kaya informasi dari data *input*, yang dapat digunakan dalam berbagai tugas pemrosesan bahasa dan komputasi visual, seperti klasifikasi teks atau pengenalan gambar. Keuntungan dari arsitektur ini adalah kesederhanaannya dan efisiensinya dalam mengatasi berbagai masalah pemrosesan data. Dengan menghilangkan bagian *decoder*, model dapat lebih fokus pada pengolahan data masukan dan menghasilkan representasi yang lebih kuat tanpa memerlukan komputasi tambahan untuk menghasilkan output.



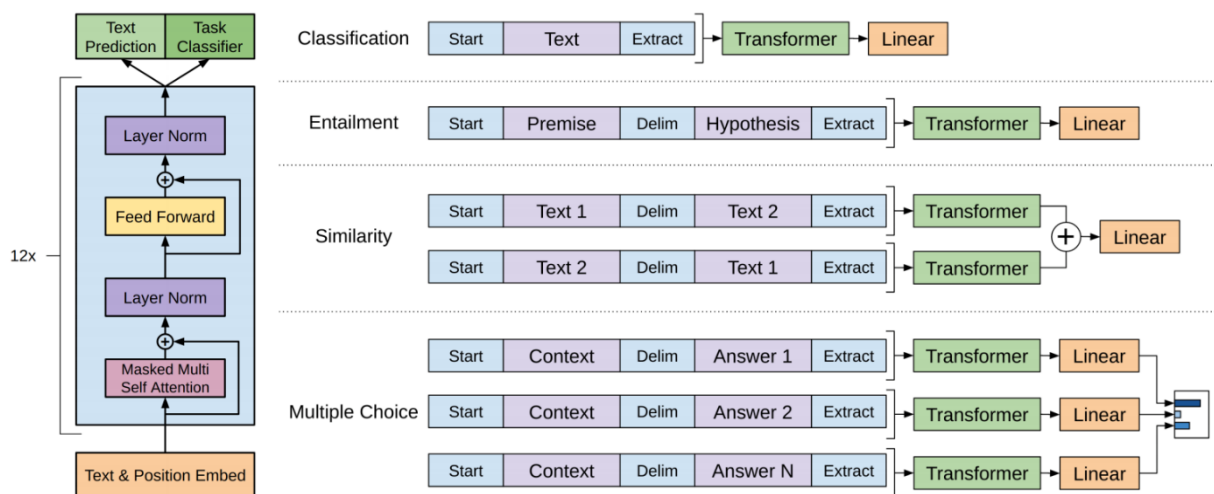
Gambar 2.9 Arsitektur BERT yang merupakan *Transformer Encoder Only*

(Sumber : <https://medium.com/thelocalminima/are-bert-features-interbertible-250a91eb9dc>)

2.2.4.3 Decoder Only

Variasi arsitektur *Decoder-Only* merupakan pendekatan yang hanya menggunakan

bagian *decoder* dari arsitektur Transformer. Dalam model ini, representasi yang sudah diolah sebelumnya (biasanya melalui proses *encoder*) diumpangkan ke dalam serangkaian lapisan *decoder*. *Decoder* bertanggung jawab untuk mengambil representasi tersebut dan menghasilkan output yang sesuai dengan tugas tertentu, seperti penerjemahan bahasa atau generasi teks. Tujuan utama dari variasi *Decoder-Only* ini adalah untuk menghasilkan *output* yang akurat dan relevan berdasarkan representasi yang diberikan. Model ini cocok digunakan dalam tugas-tugas yang memerlukan pemrosesan data berdasarkan representasi kontekstual yang sudah terbentuk sebelumnya, sehingga *decoder* dapat berfokus pada generasi hasil akhir dengan memanfaatkan informasi yang sudah ada. Keuntungan dari arsitektur ini adalah kesederhanaannya dan efisiensinya dalam menghasilkan output dengan memanfaatkan representasi yang sudah diproses sebelumnya.



Gambar 2.10 Arsitektur GPT yang merupakan Transformer *Decoder Only*
(Sumber : https://huggingface.co/docs/transformers/tasks_explained)

2.2.5 TensorFlow

TensorFlow merupakan *library* yang difokuskan untuk mengerjakan kegiatan yang berhubungan dengan *machine learning* dan *artificial intelligence*. Pada awalnya, TensorFlow dikembangkan oleh tim Google Brain untuk keperluan riset dan produksi internal Google. Pada tahun 2017, versi 1.0.0 TensorFlow dirilis untuk umum. *Library* ini tersedia dalam beberapa bahasa pemrograman seperti Python, C++, JavaScript, dan Java. Dalam penggunaannya, TensorFlow menyediakan API yang menggunakan Keras. API tersebut memungkinkan pengguna untuk membuat model *machine learning* sesuai dengan kebutuhan. TensorFlow dapat memanfaatkan beberapa CPU dan GPU yang tersedia pada komputer untuk menjalankan *model implementation*.

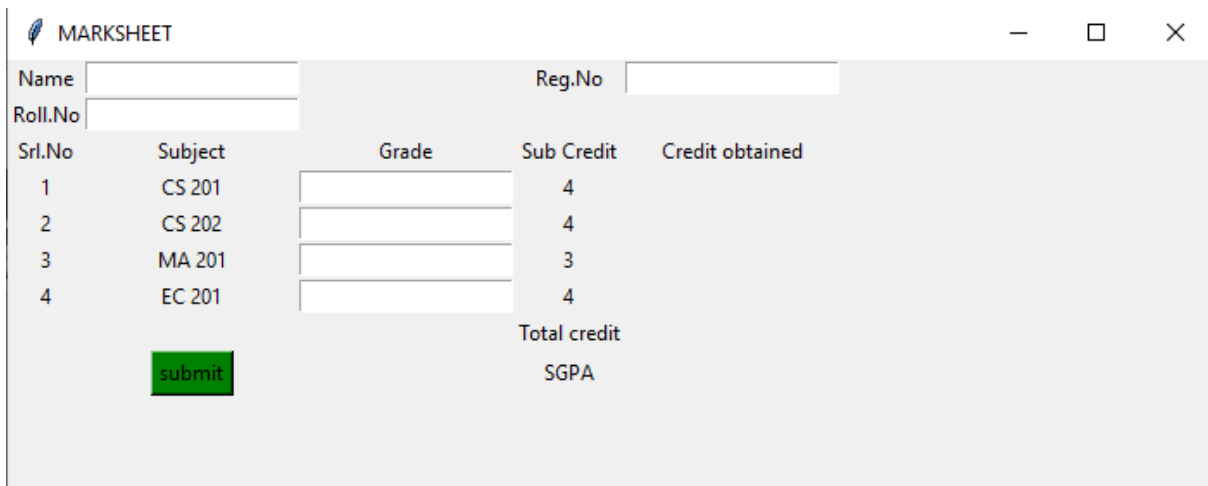
2.2.6 OpenCV

OpenCV (*Open Source Computer Vision*) adalah *library* visi komputer dan pengolahan citra yang bersifat *open source* dan dikembangkan terutama untuk bahasa pemrograman Python, tetapi juga mendukung bahasa pemrograman lain seperti C++, Java, dan lainnya. Pustaka ini menyediakan berbagai alat dan fungsi untuk mengolah gambar dan video, seperti membaca, menulis, dan memanipulasi gambar, mendeteksi objek dan wajah, mengukur jarak, mengenali pola, serta menerapkan teknik pengolahan citra seperti transformasi, filtrasi, dan lainnya. OpenCV juga mendukung pemrosesan citra secara *real-time* untuk aplikasi kecerdasan buatan, robotika, pengenalan pola, dan berbagai bidang lain yang memerlukan analisis citra.

Pustaka ini memiliki performa yang tinggi, berkat implementasi algoritma yang dioptimalkan menggunakan teknologi multi-threading dan SIMD (*Single Instruction, Multiple Data*).

2.2.7 Tkinter

Tkinter adalah pustaka (*library*) GUI (*Graphical User Interface*) standar yang terdapat dalam bahasa pemrograman Python. Pustaka ini menyediakan beragam *widget* dan alat yang memungkinkan para pengembang untuk membuat antarmuka grafis dalam aplikasi mereka dengan mudah dan cepat. Tkinter didasarkan pada toolkit Tk, yang awalnya ditulis dalam bahasa Tcl, dan diintegrasikan dengan Python melalui binding. Tkinter menyediakan elemen-elemen GUI seperti jendela (*window*), tombol (*button*), kotak teks (*text box*), dan elemen GUI lainnya yang memudahkan interaksi dengan pengguna. Penggunaan Tkinter cukup sederhana dan intuitif, sehingga cocok untuk pemula dan para pengembang yang ingin membangun aplikasi berbasis GUI tanpa terlalu rumit. Dengan Tkinter, pengembang dapat mengatur tata letak (*layout*) elemen GUI secara fleksibel, menambahkan fungsi interaktif, dan memberikan respons atas aksi pengguna. Selain itu, Tkinter juga memungkinkan para pengembang untuk membuat aplikasi desktop lintas platform yang kompatibel dengan berbagai sistem operasi.



Srl.No	Subject	Grade	Sub Credit	Credit obtained
1	CS 201	<input type="text"/>	4	
2	CS 202	<input type="text"/>	4	
3	MA 201	<input type="text"/>	3	
4	EC 201	<input type="text"/>	4	

Total credit SGPA

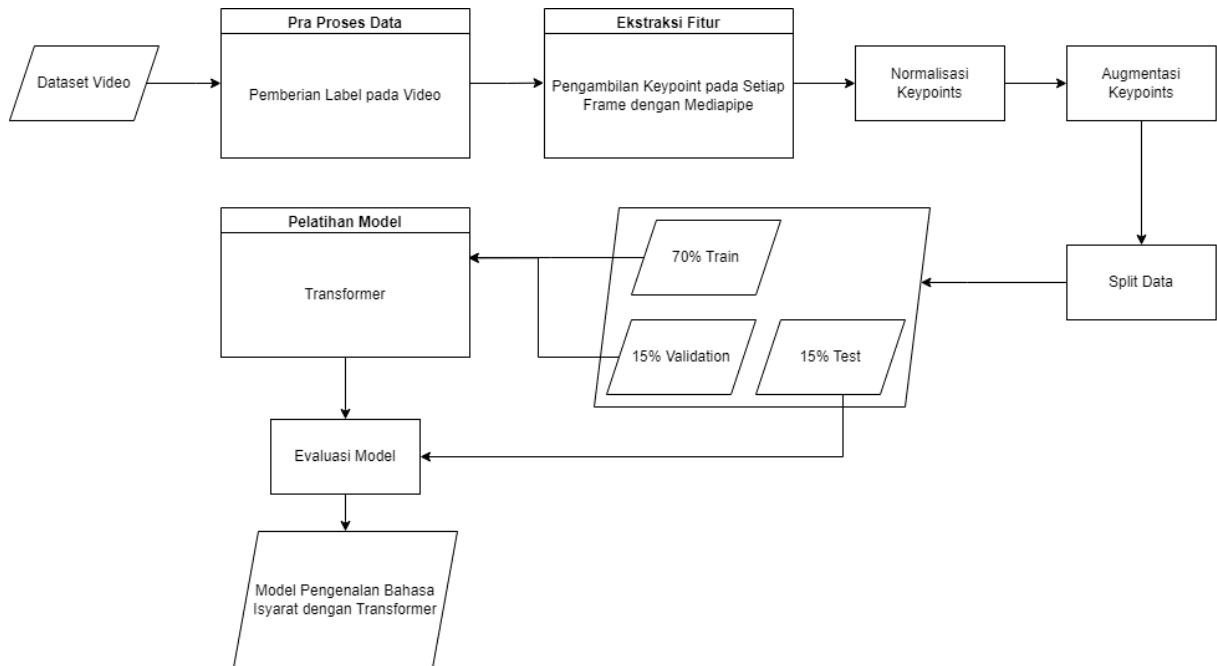
Gambar 2.11 Contoh Antarmuka Tkinter (Sumber : GeeksForGeeks)

BAB III METODOLOGI

3.1 Metode yang dirancang

Pada sub bab 3.1, akan membahas mengenai perancangan pembuatan model, pembuatan aplikasi, serta *dataset* yang akan digunakan. Adapun diagram alir (*flowchart*) sebagai visualisasi proses.

3.1.1 Perancangan Model Pengenalan Bahasa Isyarat Menggunakan Transformer



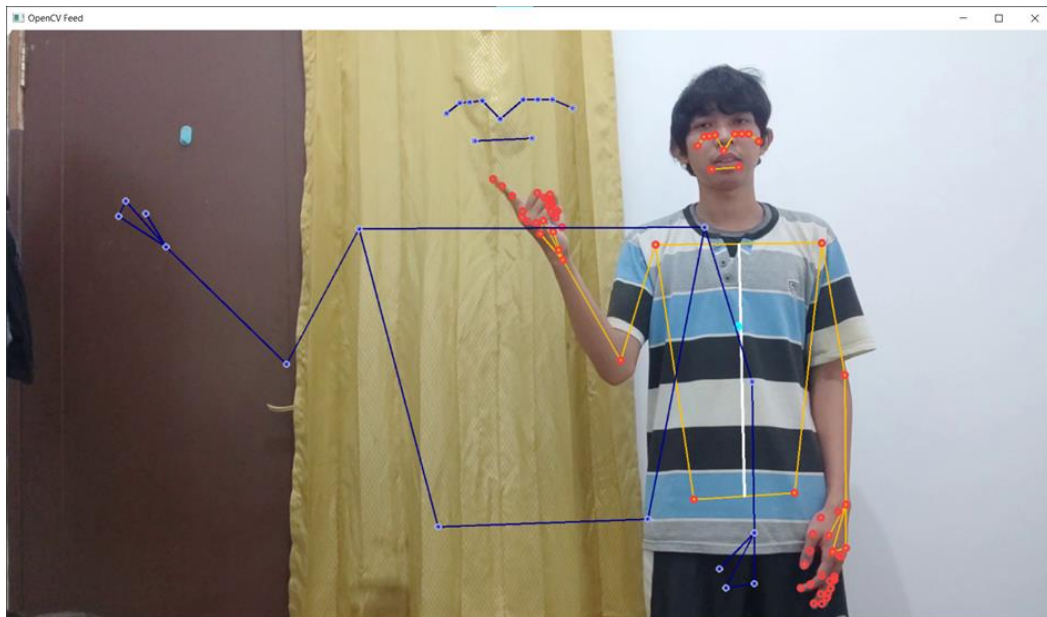
Gambar 3.1 Diagram Alir Pembuatan Model

Pada Gambar 3.1 menunjukkan rancangan pembuatan model pengenalan bahasa isyarat dengan transformer. Tahap pertama yang dilakukan adalah melakukan *preprocessing* pada *dataset* yang digunakan dengan memberikan label pada video. Proses pemberian label dilakukan dengan mengelompokkan video pada folder dengan nama label sesuai *class* yang ada. *Preprocess* tersebut dilakukan untuk mempermudah proses pengenalan oleh model dan pengolahan *dataset*. Setelah *preprocess* akan dilakukan ekstraksi fitur dengan mengambil *keypoint* setiap *frame* menggunakan MediaPipe. *Keypoints* yang diekstrak adalah 33 *keypoints* bagian lengan, 21 *keypoints* bagian telapak tangan kanan, dan 21 *keypoints* bagian telapak tangan kiri. Dari masing-masing *keypoints* tersebut akan disimpan koordinat tiga dimensinya (koordinat XYZ). Total fitur yang diekstrak adalah 225 fitur.

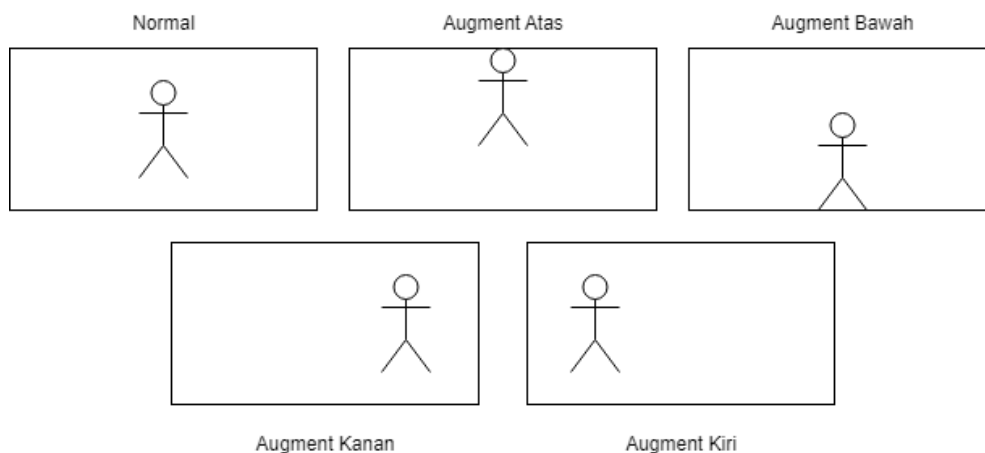
Proses berikutnya adalah normalisasi data *keypoints* untuk meningkatkan akurasi model dan mempermudah augmentasi. Tahap pertama yang dilakukan dalam melakukan normalisasi adalah mencari *center* atau bagian tengah badan dengan menggunakan fungsi *euclidean* dengan titik bahu kiri, bahu kanan, pinggang kanan dan pinggang kiri. Setelah itu, *keypoints* lain akan diatur kembali dengan menyesuaikan resolusi dari video yang digunakan. *Keypoints* yang telah dinormalisasi akan menjadi seragam nilai posisi (posisi XYZ) dan jarak dari *center* ke *keypoints*.

Setelah normalisasi, dilakukan data diaugmentasi dengan menambahkan data yang menggeser posisi *keypoints*. Data digeser ke arah atas, bawah, kiri dan kanan dengan translasi (perpindahan) pada sumbu X atau Y untuk setiap *keypoints* yang telah dinormalisasi. Adanya

data augmentasi dapat menambah akurasi model pengenalan bahasa isyarat. Selain itu, augmentasi data dilakukan untuk menghindari *overfitting* pada saat *training* model.



Gambar 3.2 Visualisasi keypoint yang diekstrak (warna merah) dan keypoint yang dinormalisasi (warna biru)



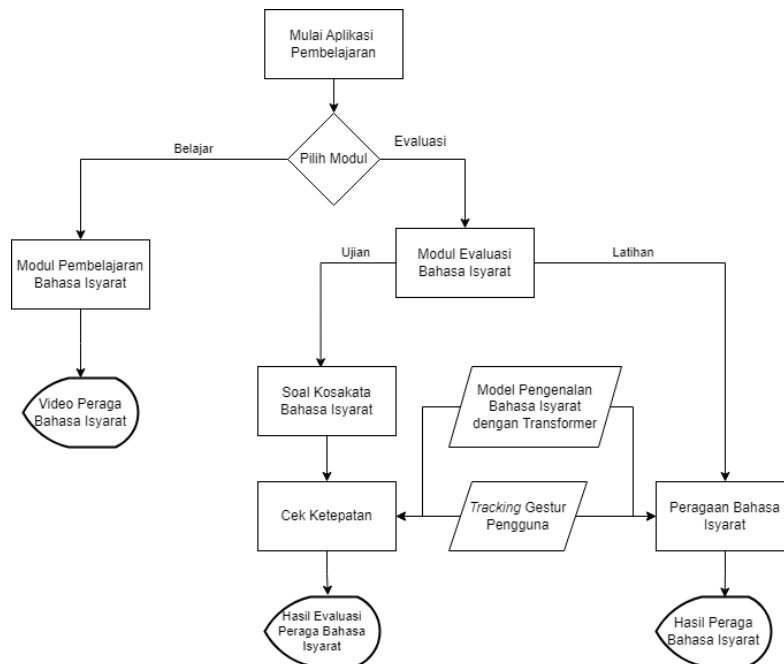
Gambar 3.3 Visualisasi Proses Augmentasi

Setelah augmentasi, dilakukan *split* data dengan rasio *train* : *validation* : *test* sebesar 70:15:15. Model transformer akan dilatih menggunakan data *train* dan divalidasi dengan *validation data*, lalu dievaluasi dengan data *test*. Metrik yang akan digunakan adalah *accuracy* dan *precision* dari evaluasi data *test*. Setelah evaluasi, model akan digunakan pada aplikasi pembelajaran.

3.1.2 Perancangan Aplikasi Pembelajaran Bahasa Isyarat

Pada Gambar 3.4 menunjukkan rancangan aplikasi pembelajaran bahasa isyarat. Saat memulai aplikasi, akan terdapat dua pilihan modul, Belajar dan Evaluasi. Pada modul belajar terdapat video yang memperagakan penggunaan bahasa isyarat dengan Standar Isyarat Bahasa Indonesia (SIBI). Pada modul evaluasi, terdapat ujian soal kosakata bahasa isyarat yang dapat

diselesaikan. Untuk melakukan pengecekan ketepatan dalam penggunaan, aplikasi akan melakukan *tracking* gestur menggunakan kamera. Gestur akan dievaluasi dengan model yang telah dilatih menggunakan transformer. Hasil evaluasi akan ditampilkan setelah proses cek ketepatan. Modul evaluasi juga menyediakan tempat untuk latihan peraga bahasa isyarat. Pada bagian latihan, diberikan kebebasan memperagakan bahasa isyarat tanpa adanya soal kosakata bahasa isyarat. Hasil deteksi peraga pada latihan akan ditampilkan secara *real time*.



Gambar 3.4 Diagram Alir Aplikasi Pembelajaran Bahasa Isyarat

3.1.3 Dataset

Untuk melatih model pengenalan bahasa isyarat menggunakan transformer, diperlukan adanya *dataset*. Pada tugas akhir ini, *dataset* bahasa isyarat dengan Standar Isyarat Bahasa Indonesia yang digunakan berisi 50 *class* video berdurasi 4 detik dengan 6 variasi posisi pemeraga, dan 3 pemeraga, serta total 20 video dalam masing-masing *class*. Jumlah video yang ada pada dataset adalah 1000 video dengan 225 *keypoints* yang diekstrak. Setiap *class* merupakan kata dasar yang sering digunakan sehari-hari. *Dataset* diambil dalam posisi *landscape* dengan resolusi 1920x1080 *pixel* dan *frame rate* 30fps. Tabel 3.1 berisikan *class* kata yang ada pada dataset.

Tabel 3.1 Daftar Kata yang Digunakan

No	Kata	No	Kata	No	Kata	No	Kata	No	Kata
1	Akan	11	Buah	21	Juga	31	Makan	41	Sebagai
2	Anda	12	Dan	22	Kami	32	Masing	42	Tambah
3	Apa	13	Dari	23	Kata	33	Mereka	43	Tangan
4	Atau	14	Dengan	24	Kecil	34	Milik	44	Tetapi
5	Baca	15	Dia	25	Kumpul	35	Minum	45	Tidak
6	Bagaimana	16	Haus	26	Labuh	36	Oleh	46	Tiga
7	Bahwa	17	Ingin	27	Lain	37	Pada	47	Udara
8	Beberapa	18	Ini	28	Laku	38	Rumah	48	Untuk
9	Besar	19	Itu	29	Lapar	39	Satu	49	Waktu
10	Bisa	20	Jadi	30	Main	40	Saya	50	Yang



Gambar 3.5 Contoh *Dataset* dengan 3 Pemeraga pada *Class* “Akan”

3.2 Peralatan pendukung

Dalam pengembangan aplikasi, diperlukan alat yang mendukung pengerjaannya. Adapun peralatan yang mendukung pengembangan aplikasi dengan rincian dan spesifikasi yang terdapat pada Tabel 3.2.

Tabel 3.2 Spesifikasi Peralatan Pendukung

No	Perangkat	Spesifikasi
1	Sistem Operasi	Windows 10 Home 64-bit
2	RAM	16GB
3	CPU	AMD Ryzen 5 4600H
4	GPU	NVIDA GeForce GTX 1650 Ti
5	Kamera	20 MP, f/2.0, (wide), 1/3", 0.9µm 1080p@30fps

3.3 Rancangan dan Implementasi Aplikasi

Pada sub bab 3.3 membahas mengenai rancangan dan implementasi aplikasi pembelajaran. Adapun hal yang dibahas seperti *pseudocode*, rancangan dan implementasi model, serta implementasi aplikasi.

3.3.1 Pseudocode

Pseudocode adalah representasi cara kerja sistem dalam bentuk *syntax* yang disederhanakan. Berikut merupakan *pseudocode* pada sistem :

3.3.1.1 Pelabelan Video

Pada Kode Semu 3.1, langkah pertama, pada baris 1 adalah membaca nilai *folder_path* dari pengguna. Selanjutnya, terdapat dua perulangan di baris 2 dan 3: pertama akan mengiterasi setiap *class_name* dalam *folder_path*, dan yang kedua akan mengiterasi setiap *videos* dalam

class_name tersebut. Di dalam perulangan kedua, nilai *video_num* akan dibaca dari pengguna, dan kemudian akan dilakukan operasi JOIN antara *class_name* dan *videos* di baris 5.

```
1 READ folder_path
2 for class_name in folder_path:
3     for videos in class_name:
4         READ video_num
5         JOIN class_name, videos
```

Kode Semu 3.1 Kode Semu Pelabelan Video

3.3.1.2 Ekstraksi Fitur

```
1 READ video
2 while video.isOpened:
3     Detect landmark from video
4     FeatureExtract landmark as keypoints_data
5     WRITE keypoints_data
```

Kode Semu 3.2 Kode Semu Ekstraksi Fitur

Kode Semu 3.2 merupakan deskripsi algoritma untuk mengolah data video dan mendeteksi *landmark* dari setiap *frame* dalam video tersebut. Algoritma dimulai dengan membaca video dari suatu sumber (misalnya file video atau perangkat kamera) pada baris 1. Selanjutnya, dalam perulangan *while* di baris 2, setiap *frame* video akan diolah untuk mendeteksi *landmark*. Hasil deteksi *landmark* dari baris 3 tersebut akan dijadikan data *keypoints_data* yang mencatat informasi tentang letak *landmark* pada setiap *frame* di baris 4. Selanjutnya, data *keypoints_data* akan disimpan dalam suatu bentuk penyimpanan yang sesuai dengan perintah *WRITE* di baris 5. Perulangan akan berlanjut selama *video.isOpened* menghasilkan nilai *True*, artinya ada *frame* yang belum diproses. Proses ini berhenti ketika seluruh *frame* video telah selesai diproses atau jika terjadi kesalahan dalam membuka video.

3.3.1.3 Normalisasi Data

```
1 READ keypoints_data
2 Normalize keypoints_data as normalized_data
3 WRITE normalized_data
```

Kode Semu 3.3 Kode Semu Normalisasi Data

Kode Semu 3.3 adalah langkah-langkah algoritma untuk memproses data *keypoints* yang sebelumnya telah disimpan dalam file atau sumber data lainnya. Langkah pertama yang terdapat pada baris 1 adalah membaca data *keypoints* dari sumber yang disebut *keypoints_data*. Setelah data *keypoints* terbaca, langkah selanjutnya adalah melakukan normalisasi terhadap *keypoints_data* tersebut. Normalisasi pada baris 2 dilakukan untuk membawa data *keypoints* ke dalam skala atau rentang nilai tertentu sehingga dapat dibandingkan atau digunakan lebih mudah dalam analisis atau pemrosesan selanjutnya. Hasil normalisasi tersebut disimpan dalam variabel *normalized_data*. Terakhir, setelah proses normalisasi selesai, hasil *normalized_data* akan disimpan kembali dengan *WRITE* pada baris 3.

3.3.1.4 Augmentasi Data

Kode Semu 3.4 adalah langkah-langkah algoritma untuk melakukan augmentasi data dengan menghasilkan berbagai variasi data baru dari data asli yang telah dinormalisasi sebelumnya. Langkah pertama yang ada di baris 1 adalah membaca data yang telah dinormalisasi dari sumber yang disebut *normalized_data*. Setelah data tersebut terbaca, algoritma melakukan proses augmentasi data dengan melakukan berbagai transformasi pada data asli untuk menghasilkan data baru.

Proses augmentasi dilakukan dengan menghasilkan empat variasi data baru, yaitu *augment_Up_data* (data diaugmentasi dengan pergeseran ke atas di baris 2), *augment_Down_data* (data diaugmentasi dengan pergeseran ke bawah di baris 3), *augment_Left_data* (data diaugmentasi dengan pergeseran ke kiri di baris 4), dan *augment_Right_data* (data diaugmentasi dengan pergeseran ke kanan di baris 5).

Setelah proses augmentasi selesai, langkah selanjutnya di baris 6 adalah menggabungkan semua data hasil augmentasi tersebut, bersama dengan data asli yang telah dinormalisasi, menjadi satu kesatuan yang disebut *dataset*. Data dalam *dataset* ini memiliki berbagai variasi yang dihasilkan dari proses augmentasi, serta data asli yang telah dinormalisasi. Terakhir, hasil *dataset* tersebut akan ditulis (WRITE) pada baris 7 untuk keperluan analisis atau pemrosesan lebih lanjut.

```
1 READ normalized_data
2 Augment normalized_data as augment_Up_data
3 Augment normalized_data as augment_Down_data
4 Augment normalized_data as augment_Left_data
5 Augment normalized_data as augment_Right_data
6 JOIN normalized_data, augment_Up_data, augment_Down_data,
  augment_Left_data, augment_Right_data as dataset
7 WRITE dataset
```

Kode Semu 3.4 Kode Semu Augmentasi Data

3.3.1.5 Train dan Evaluasi Model

```
1 READ dataset
2 Split dataset to train_data, validation_data, test_data
3 Define Transformer
4 Train train_data with Transformer as model
5 Evaluate model
6 Export model
```

Kode Semu 3.5 Kode Semu Train dan Evaluasi Model

Kode Semu 3.5 adalah deskripsi langkah-langkah algoritma untuk melatih dan menguji sebuah model menggunakan dataset yang telah dibaca sebelumnya. Langkah pertama di baris 1 adalah membaca *dataset* dari sumber data yang disebut *dataset*. Setelah dataset terbaca, langkah selanjutnya pada baris 2 adalah membagi *dataset* menjadi tiga bagian, yaitu *train_data*, *validation_data* dan *test_data*. Bagian *train_data* dan *validation_data* akan digunakan untuk melatih model, sedangkan *test_data* digunakan untuk menguji model.

Setelah *dataset* dibagi, algoritma mendefinisikan sebuah model yang disebut Transformer di baris 3. Kemudian langkah berikutnya pada baris 4 adalah melatih model menggunakan *train_data* dan *validation_data*. Proses pelatihan ini akan mengoptimalkan parameter-parameter dalam model agar dapat melakukan prediksi yang akurat berdasarkan data yang diberikan. Setelah pelatihan selesai, langkah selanjutnya adalah mengevaluasi kinerja model menggunakan *test_data*. Evaluasi di baris 5 akan memberikan gambaran tentang seberapa baik model berperforma pada data yang belum pernah dilihat sebelumnya.

Terakhir, jika model telah memberikan hasil yang memuaskan, algoritma akan mengekspor (*export*) model tersebut dengan kode pada baris 6. Proses ini menghasilkan file atau representasi yang dapat digunakan untuk memuat model ke dalam aplikasi.

3.3.1.6 Modul Belajar

Kode Semu 3.6 merupakan deskripsi algoritma untuk memainkan dan menutup video yang telah dipilih oleh pengguna. Langkah pertama adalah membaca dua *input* dari pengguna,

yaitu *video_path* di baris 1 dan *video_belajar* di baris 2. Variabel *video_path* berisi lokasi atau alamat dari video yang ingin diputar, sedangkan variabel *video_belajar* berisi nama file video yang ingin diputar sebagai video pembelajaran atau referensi.

Setelah dua *input* tersebut terbaca, langkah selanjutnya adalah memanggil fungsi *PlayVideo* pada baris 4 dengan argumen *video_path* dan *video_belajar*. Setelah proses pemutaran selesai, pemutaran video akan ditutup secara otomatis dengan fungsi *CloseVideo* di baris 5.

```
1 READ video_path
2 READ video_belajar
4 PlayVideo video_path,video_belajar
5 CloseVideo
```

Kode Semu 3.6 Kode Semu Modul Belajar

3.3.1.7 Modul Evaluasi

```
1 Import model
2 while camera.READ(video) :
3     Detect frame_landmark
4     Extract frame_landmark
5     Predict frame_landmark to model
6
7     if frame_landmark == evaluated_model
8         if Cek_Ketepatan == True
9             PRINT 'Benar !'
10    else
11        PRINT 'Salah !'
```

Kode Semu 3.7 Kode Semu Modul Evaluasi

Kode Semu 3.7 adalah deskripsi algoritma yang menggambarkan penggunaan sebuah model untuk mengenali suatu objek atau fitur dalam frame video dari kamera secara *real-time* pada modul evaluasi aplikasi pembelajaran.

Langkah pertama yang ada pada baris 1 adalah meng-*import* atau memuat model yang akan digunakan untuk melakukan prediksi pada *frame* video. Model ini sebelumnya telah dilatih untuk mengenali objek atau fitur yang diinginkan. Selanjutnya, algoritma akan memulai perulangan (*while loop*) dari baris 2 hingga 11 untuk membaca setiap *frame* dari video yang diambil oleh kamera. Di dalam setiap iterasi akan dilakukan deteksi *landmark* seperti pada baris 3, kemudian dilanjutkan dengan ekstraksi data *landmark* di baris 4. Lalu, data *landmark* akan diprediksi di baris 5 menggunakan model yang di-*import*. Setelah itu pada baris 7 hingga 11 akan dilakukan pengecekan hasil prediksi dari model dengan objek atau fitur yang dievaluasi.

Jika hasil prediksi model (*frame_landmark*) sama dengan objek yang dievaluasi (*evaluated_model*), maka variabel *Cek_Ketepatan* di-*set* sebagai True, dan akan mencetak pesan "Benar!" dengan menggunakan perintah PRINT. Jika hasil prediksi model tidak sama dengan objek yang dievaluasi, maka algoritma akan mencetak pesan "Salah!" menggunakan perintah PRINT.

3.3.2 Model Pengenalan Bahasa Isyarat

Pada subbab ini, rincian perancangan dan implementasi model pengenalan bahasa isyarat yang menggunakan arsitektur transformer akan dibahas. Model dirancang untuk dapat mengenali dan menerjemahkan gestur tangan dalam bahasa isyarat menjadi kata.

Dalam perancangan model, langkah awal adalah menentukan arsitektur transformer yang sesuai untuk pengenalan bahasa isyarat. Pada umumnya arsitektur transformer memiliki dua lapisan utama yang memproses data masukan. Lapisan pertama – *Encoder* – berfungsi

untuk mengambil *input* sekuensial dan menghasilkan *output* yang mengikuti konteks pada sekuens. Kemudian lapisan berikutnya yakni *Decoder*, bertanggung jawab untuk menghasilkan keluaran berdasarkan informasi yang diterima dari lapisan *encoder* dan *output* sebelumnya (dalam kasus model sekuensial). *Decoder* melakukan generasi *output* dengan memperhatikan konteks dari urutan masukan serta urutan keluaran yang telah dihasilkan sebelumnya.

Terdapat variasi Transformer yang disebut sebagai "Transformer *Encoder-Only*". Pada variasi ini hanya lapisan *encoder* yang digunakan untuk memproses masukan dan menghasilkan representasi yang diperlukan. Pada Transformer *Encoder-Only*, tidak memerlukan generasi *output* dari lapisan *decoder*. Dalam kasus ini, lapisan *decoder* dihilangkan untuk mengurangi kompleksitas dan mengurangi jumlah parameter dalam model.

Arsitektur yang digunakan pada model pengenalan bahasa isyarat tidak memerlukan generasi *output*. Sebagai gantinya, arsitektur tersebut berfokus pada pemrosesan dan pengenalan urutan gerakan atau isyarat dalam bahasa isyarat. Pada *dataset*, hasil yang diekstrak berupa posisi *keypoints* sebanyak 225 titik. Model pengenalan bahasa isyarat akan dilatih untuk mengenali perubahan posisi *keypoints* dari satu *frame* ke *frame* berikutnya. Kode Sumber 3.1 merupakan fungsi yang digunakan untuk membuat model dengan arsitektur Transformer *Encoder-Only*.

```

1 def create_transformer_model(sequence_length, num_features, num_classes,
num_layers, hidden_units, num_heads, dropout_rate):
2     # Input Layer
3     inputs = layers.Input(shape=(sequence_length, num_features))
4     x = inputs
5
6     # Positional Encoding Layer
7     positional_encoding = layers.Embedding(input_dim=sequence_length,
output_dim=num_features)(tf.range(sequence_length))
8     x += positional_encoding
9
10    # Encoder Layers
11    for _ in range(num_layers):
12        # Multi-Head Attention
13        attention = layers.MultiHeadAttention(num_heads=num_heads,
key_dim=num_features)(x, x)
14        attention = layers.Dropout(rate=dropout_rate)(attention)
15        attention = layers.LayerNormalization(epsilon=1e-6)(attention + x)
16
17        # Feed Forward Neural Network
18        ffn = layers.Dense(units=hidden_units, activation='relu')(attention)
19        ffn = layers.Dropout(rate=dropout_rate)(ffn)
20        ffn = layers.LayerNormalization(epsilon=1e-6)(ffn + attention)
21
22        x = ffn
23
24    # Global Average Pooling
25    x = layers.GlobalAveragePooling1D()(x)
26
27    # Output Layer
28    outputs = layers.Dense(units=num_classes, activation='softmax')(x)
29
30    # Create and compile the model
31    model = tf.keras.Model(inputs=inputs, outputs=outputs)

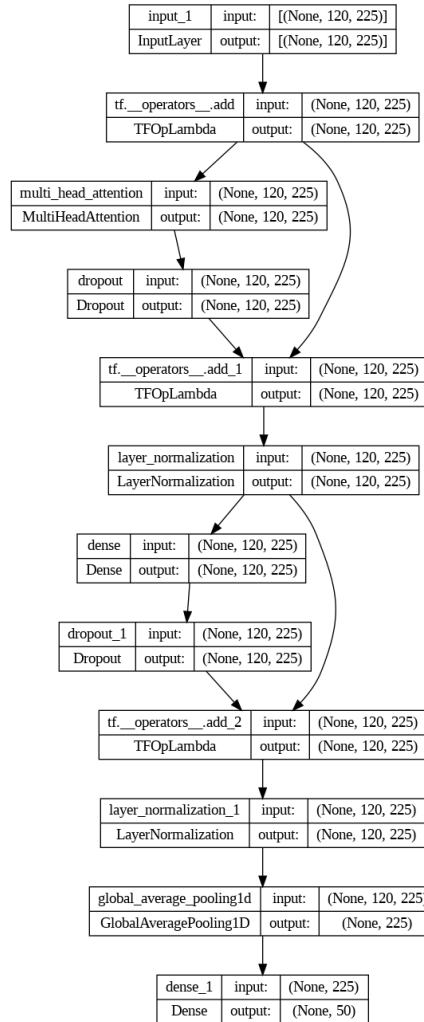
```

```

32 model.compile(optimizer='adam', loss='categorical_crossentropy',
33               metrics=['accuracy'])
34 return model
35

```

Kode Sumber 3.1 Arsitektur Model



Gambar 3.6 Visualisasi *layer* model

Lapisan pertama pada arsitektur model adalah *Input Layer* yang ada pada baris 3. Lapisan ini menentukan ukuran input yang akan digunakan pada model. *Input* ini berupa urutan (*sequence*) dengan panjang '*sequence_length*' dan '*num_features*' fitur dalam setiap langkah urutan. Dalam model pengenalan bahasa isyarat, '*sequence_length*' yang digunakan adalah 120 – sesuai panjang video 30 fps x 4 detik. Kemudian '*num_features*' sejumlah 225, sesuai dengan *keypoints* yang ada pada *dataset*.

Lapisan kedua adalah *Positional Encoding Layer* yang ada pada baris 7 hingga 8. Lapisan ini menggunakan *layer Embedding* untuk mengodekan informasi posisional dalam urutan dengan dimensi yang sama seperti fitur *input*. Ini bertujuan untuk memberikan informasi posisional ke model agar dapat memperhatikan urutan.

Lapisan berikutnya adalah *Encoder Layer* yang ada pada baris 11 sampai 22. Lapisan ini dapat diberikan secara majemuk atau tunggal untuk model. Lapisan *Encoder* terdiri dari beberapa sub lapisan yang menjadi komponen penting dalam arsitektur model. Pertama, *Multi-*

Head Attention untuk memperhatikan hubungan kontekstual antara gerakan-gerakan isyarat dalam urutan. Kemudian, *Dropout* untuk mengurangi *overfitting*, diikuti dengan normalisasi menggunakan *Layer Normalization*. Setelah itu, *Feed Forward Neural Network* (FFN) untuk memperoleh pemahaman yang lebih kompleks pada urutan. Selanjutnya, diterapkan lagi *Dropout* dan normalisasi menggunakan *Layer Normalization*.

Lapisan setelah *encoder* adalah *Global Average Pooling* di baris 25. Pada lapisan ini, menggunakan *GlobalAveragePooling1D* untuk menghasilkan representasi global dari urutan setelah lapisan *encoder*. Lapisan ini mengurangi dimensi ruang dan mengambil rata-rata dari setiap fitur dalam urutan.

Lapisan terakhir yakni *Output Layer* di baris 28, menggunakan lapisan *Dense* dengan fungsi aktivasi *softmax* untuk melakukan klasifikasi pada output dari *Global Average Pooling*. Jumlah unit pada lapisan ini sesuai dengan jumlah kelas yang akan diprediksi.

3.3.3 Aplikasi Pembelajaran Bahasa Isyarat

Pada sub bab ini, akan dijelaskan mengenai implementasi aplikasi pembelajaran bahasa isyarat menggunakan bahasa pemrograman Python. Aplikasi ini menggunakan *library* Tkinter, OpenCV, dan MediaPipe. Terdapat dua modul utama pada aplikasi pembelajaran bahasa isyarat, yakni modul Belajar dan modul Evaluasi.

Dalam implementasinya, aplikasi ini menggunakan *framework* Tkinter sebagai antarmuka grafis, OpenCV untuk memanipulasi dan memutar video, serta mediapipe untuk mendeteksi pose tubuh dan tangan. Modul Belajar memiliki fitur untuk memutar video peragaan bahasa isyarat. Sedangkan pada modul Evaluasi, terdapat fitur untuk memfasilitasi pengguna dalam mempraktekan bahasa isyarat. Praktek bahasa isyarat tersebut memiliki soal, dimana pengguna harus mencocokkan bahasa isyarat yang diperagakan dengan soal yang diberikan.

3.3.3.1 Modul Belajar

```
1 def handle_video(self, file_path, window_width=800, window_height=600):
2     # Load the video file
3     video = cv2.VideoCapture(file_path)
4
5     # Check if video file is opened successfully
6     if not video.isOpened():
7         print("Error opening video file")
8         return
9
10    # Get the frames per second (fps) of the video
11    fps = video.get(cv2.CAP_PROP_FPS)
12
13    # Set the desired window width and height
14    cv2.namedWindow('Video Player', cv2.WINDOW_NORMAL)
15    cv2.resizeWindow('Video Player', window_width, window_height)
16
17    while True:
18        # Read the current frame from the video
19        ret, frame = video.read()
20
21        if not ret:
22            # End of video
23            print("Video playback finished")
24            break
25        # Display the frame in the window named 'Video Player'
```

```

26         cv2.imshow('Video Player', frame)
27         # Check if the 'q' key is pressed
28         if cv2.waitKey(int(1000 / fps)) & 0xFF == ord('q'):
29             break
30
31     # Release the video object and close all windows
32     video.release()
33     cv2.destroyAllWindows()
34
35     # Close the window if it's still open
36     if cv2.getWindowProperty('Video Player', cv2.WND_PROP_VISIBLE) > 0:
37         cv2.waitKey(1)
38         cv2.destroyWindow('Video Player')

```

Kode Sumber 3.2 Fungsi Utama pada Modul Belajar

Kode Sumber 3.2 merupakan fungsi utama yang digunakan dalam menangani proses pada modul Evaluasi. Pada modul belajar, fungsi “*handle_video*” pada akan membuka file video yang diberikan, membaca *frame* per detik (fps), dan membuat jendela pemutar video. Selama dalam loop, kode akan membaca *frame* dari video dan menampilkannya dalam jendela “*Video Player*” hingga akhir video atau tombol 'q' ditekan. Setelah *loop* selesai, objek video dilepaskan, jendela ditutup, dan kode memeriksa keberadaan jendela pemutar video untuk ditutup secara eksplisit. Dengan demikian, kode tersebut menyediakan kerangka dasar untuk memutar video dengan OpenCV, termasuk pengaturan ukuran jendela dan pengendalian pemutaran menggunakan masukan pengguna. Video diputarkan dari *path* yang sesuai dengan *class* video.

3.3.3.2 Modul Evaluasi

```

1  def run_detection (model,array):
2      shuffle_soal(array)
3
4      # Set mediapipe model
5      with mp_holistic.Holistic(min_detection_confidence=0.6,
6      min_tracking_confidence=0.5) as holistic:
7          for frame_num in range(sequence_length):
8              # Read feed
9              ret, frame = cap.read()
10
11              # Make detections
12              coordinates = normalize(holistic, mp_holistic, frame)
13
14              # NEW Apply wait logic
15              if frame_num == 0:
16                  Display_WaitScreen(2500)
17              else:
18                  Display_DetectScreen()
19                  Detect_keypoints()
20                  cv2.imshow('OpenCV Feed', cv2.resize(frame, (800,
21                  600)))
22
23                  if cv2.waitKey(1) & 0xFF == ord('q'):
24                      break
25
26      cap.release()
27      cv2.destroyAllWindows()
28      # Open a new window after the main loop
29      #Predict and Detect

```

```
27     if len(frame_num) == 120:
28         StartPrediction()
29     Display_EvaluasiScreen()
30     cv2.imshow('New Window', cv2.resize(new_window, (800, 600)))
31     cv2.waitKey(0) # Wait until a key is pressed
32     cv2.destroyAllWindows()
```

Kode Sumber 3.3 Fungsi Utama pada Modul Evaluasi

Kode Sumber 3.3 merupakan fungsi utama yang digunakan dalam menangani proses pada modul Evaluasi. Ketika proses evaluasi dimulai, maka *window* baru akan terbuka. *Window* tersebut digunakan sebagai media visual evaluasi. Saat proses evaluasi berlangsung, *webcam* akan menyala untuk merekam video pengguna dalam memperagakan bahasa isyarat sesuai soal acak yang diberikan. Video yang direkam dengan *frame rate* 30fps ditampilkan pada *window* evaluasi yang telah dibuat secara *real-time*. Selain itu, aplikasi akan menangkap *keypoints* yang terdeteksi dari *frame* video dan mengumpulkan data *keypoints* dalam sebuah *array*. Terdapat jeda 2,5 detik serta pesan ‘Bersiap’ pada awal proses evaluasi sebagai pertanda untuk bersiap-siap. *Window* proses evaluasi akan tertutup setelah *array* memiliki 120 elemen, lalu *array* akan digunakan untuk prediksi gerakan dengan model yang telah dilatih.

Hasil prediksi akan ditampilkan pada *window* baru. Pada *window* tersebut akan ditampilkan soal yang diberikan pada proses evaluasi, hasil prediksi gerakan yang direkam, serta evaluasi dalam bentuk ‘Benar’ atau ‘Salah’. Jendela hasil prediksi dapat ditutup dengan tombol ‘q’. Ketika ditutup, pengguna akan kembali pada menu utama modul evaluasi.

BAB IV HASIL DAN PEMBAHASAN

4.1 Analisis Model Pengenalan Bahasa Isyarat

Dalam mengimplementasikan model, terdapat beberapa skenario uji coba dengan mengganti beberapa parameter untuk menemukan model dengan kinerja terbaik. Metrik yang digunakan untuk mengukur kinerja adalah *test accuracy* (akurasi tes), *test precision* (presisi tes). Kedua metrik dipilih karena merepresentasikan kemampuan model untuk memprediksi dengan benar dan konsisten. Parameter yang diubah pada uji coba adalah jumlah *epoch*, jumlah *multihead*, dan jumlah *encoder layer*. Adapun skenario yang digunakan dalam mengimplementasikan model diantaranya sebagai berikut.

Tabel 4.1 Tabel Skenario dan Hasil Uji Coba Dengan Data Augmentasi

No	Skenario	<i>Epoch</i>	<i>Multihead</i>	<i>Encoder Layer</i>	<i>Test Accuracy (%)</i>	<i>Test Precision (%)</i>
1	Normal	50	4	1	100,00	100,00
2	Normal	40	4	1	98,67	100,00
3	Penambahan <i>Multihead</i>	50	8	1	93,33	95,24
4	Penambahan <i>Multihead</i>	40	8	1	96,00	96,64
5	Penambahan <i>Encoder Layer</i>	50	4	2	97,33	97,33
6	Penambahan <i>Encoder Layer</i>	40	4	2	96,67	96,67
7	Penambahan <i>Multihead</i> dan <i>Encoder Layer</i>	50	8	2	86,00	87,16
8	Penambahan <i>Multihead</i> dan <i>Encoder Layer</i>	40	8	2	88,67	90,48

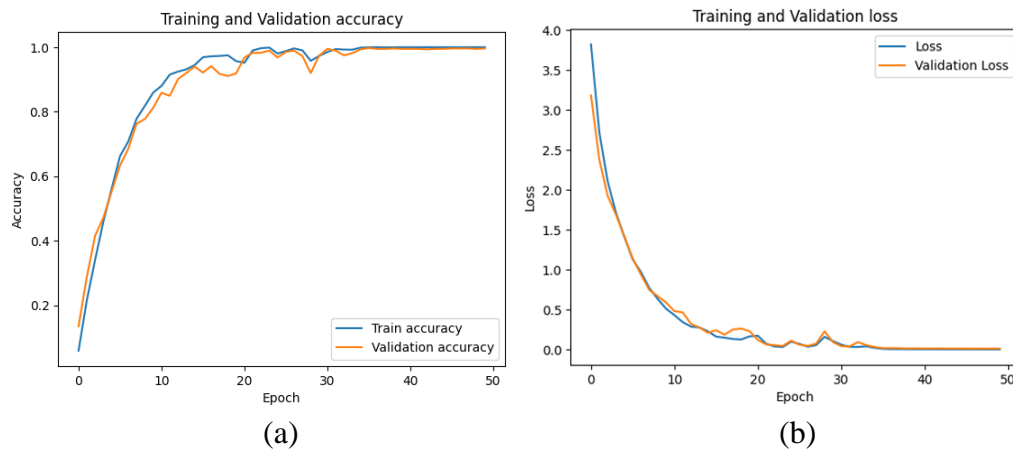
Tabel 4.1 memuat informasi mengenai skenario dan hasil uji coba. Pada skenario normal, parameter yang digunakan adalah jumlah *epoch* dengan variasi 50 dan 40 *epoch*. Skenario penambahan *multihead* menggunakan parameter *epoch* 50 dan 40, serta *multihead* sejumlah 8. Skenario penambahan *encoder layer* terdapat peningkatan pada jumlah *encoder layer* menjadi 2, sementara parameter *epoch* tetap menggunakan 50 dan 40. Terakhir, skenario penambahan *multihead* dan *encoder layer* menggabungkan perubahan pada kedua skenario sebelumnya, dengan *multihead* sejumlah 8 dan *encoder layer* sejumlah 2, serta parameter *epoch* 50 dan 40.

4.1.1 Analisis Hasil Skenario 1

Gambar 4.1a dan 4.1b berisi grafik akurasi dan loss pelatihan model dengan skenario uji coba 1. Pada uji coba ini menggunakan 50 epoch, 4 *multihead*, dan 1 *encoder layer*. Gambar 4.1a menunjukkan perubahan akurasi model seiring dengan bertambahnya epoch. Di awal training, model memiliki akurasi yang rendah, lalu terjadi peningkatan akurasi secara pesat pada pertengahan epoch 0 dan 10. Pada epoch antara 10 dan 20, akurasi validation naik menyusul akurasi train meskipun terjadi beberapa drop. Di pertengahan epoch 20 dan 30 akurasi masih bersifat fluktuatif. Akurasi train dan validation konvergen kembali mulai epoch 30.

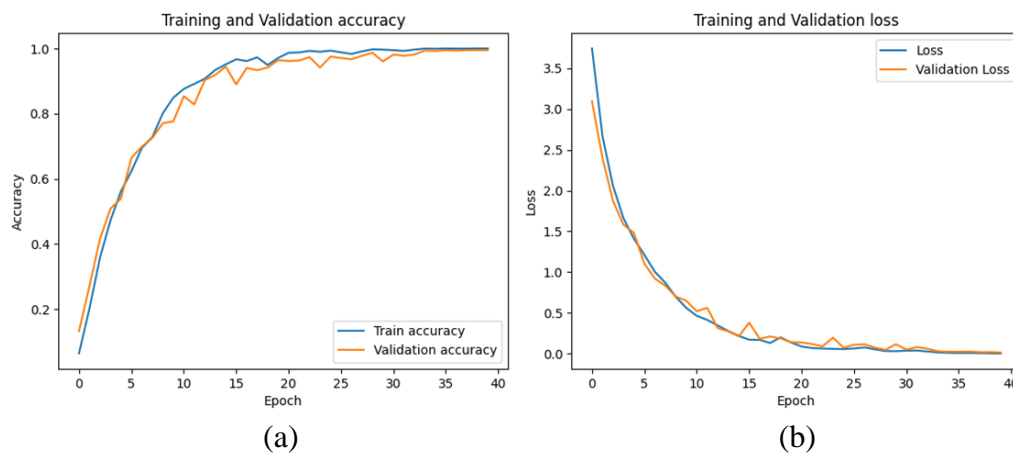
Gambar 4.1b menunjukkan perubahan loss model seiring dengan bertambahnya epoch. Di awal training, model memiliki loss yang tinggi, lalu terjadi penurunan loss secara pesat pada pertengahan epoch 0 dan 10. Pada epoch antara 10 dan 20, validation loss turun menyusul train loss. Loss model mulai konvergen di antara epoch 20 dan 30. Di epoch 30, train dan validation loss mulai konvergen hingga akhir epoch.

Dari kedua grafik tersebut, didapatkan bahwa model dapat mengenali data yang diberikan. Proses pelatihan model skenario uji coba 1 memakan waktu 28 detik untuk setiap epoch, dengan total waktu pelatihan 1402 detik atau 23 menit. Model pertama memiliki akurasi test 100,00% dengan presisi test 100,00%.



Gambar 4.1 (a) Grafik Akurasi *Train* dan *Validation*, (b) Grafik *Train* dan *Validation Loss* dari Skenario Uji Coba 1

4.1.2 Analisis Hasil Skenario 2



Gambar 4.2 (a) Grafik Akurasi *Train* dan *Validation*, (b) Grafik *Train* dan *Validation Loss* dari Skenario Uji Coba 2

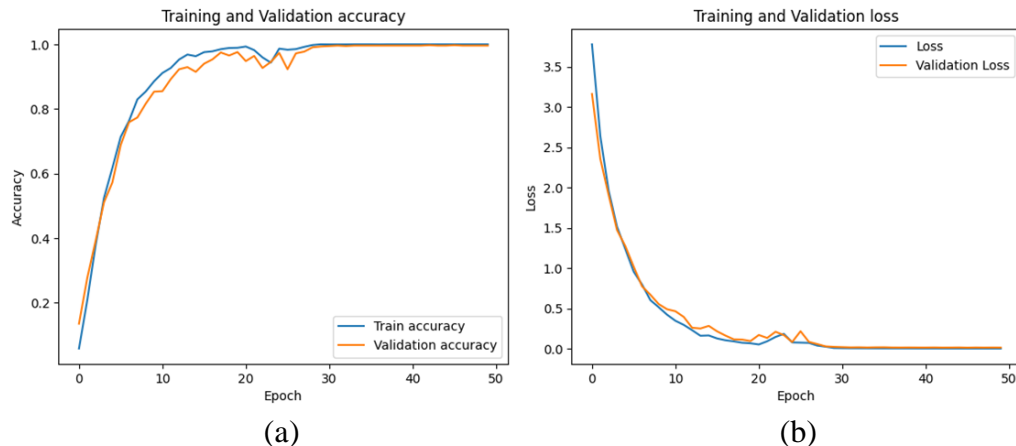
Gambar 4.2a dan 4.2b merupakan grafik akurasi dan *loss* pelatihan model dengan skenario uji coba 2. Pada uji coba ini menggunakan 40 *epoch*, 4 *multihead*, dan 1 *encoder layer*. Gambar 4.2a menunjukkan perubahan akurasi model seiring dengan bertambahnya *epoch*. Di awal *training*, model memiliki akurasi yang rendah, lalu terjadi peningkatan akurasi secara pesat pada pertengahan *epoch* 0 dan 10. Pada *epoch* 20, akurasi *validation* menyusul akurasi *train*. Akurasi *validation* dan *train* tidak stabil hingga *epoch* 35. Setelahnya akurasi *validation* dan *train* menjadi konvergen hingga *epoch* 40.

Gambar 4.2b menunjukkan perubahan *loss* model seiring dengan bertambahnya *epoch*.

Di awal *training*, model memiliki *loss* yang tinggi, lalu terjadi penurunan *loss* secara pesat pada pertengahan *epoch* 0 dan 10. Pada *epoch* antara 10 dan 20, *validation loss* turun menyusul *train loss* dan mulai konvergen pada *epoch* 35. *Loss* model tetap stabil hingga *epoch* 40.

Dari kedua grafik tersebut, didapatkan bahwa model dapat mengenali data yang diberikan dengan baik. Proses pelatihan model skenario uji coba 2 memakan waktu 28 detik untuk setiap *epoch*, dengan total waktu pelatihan 1138 detik atau 18 menit. Model kedua memiliki akurasi *test* 98,67% dengan presisi *test* 100,00%.

4.1.3 Analisis Hasil Skenario 3



Gambar 4.3 (a) Grafik Akurasi *Train* dan *Validation*, (b) Grafik *Train* dan *Validation Loss* dari Skenario Uji Coba 3

Gambar 4.3a dan 4.3b merupakan grafik akurasi dan *loss* pelatihan model dengan skenario uji coba 3. Pada uji coba ini menggunakan 50 *epoch*, 8 *multihead*, dan 1 *encoder layer*. Gambar 4.3a menunjukkan perubahan akurasi model seiring dengan bertambahnya *epoch*. Di awal *training*, model memiliki akurasi yang rendah, lalu terjadi peningkatan akurasi secara pesat pada pertengahan *epoch* 0 dan 10. Pada pertengahan *epoch* 20 dan 30, akurasi *validation* mengalami sedikit penurunan. Akurasi *validation* dan *train* mulai konvergen dari *epoch* 30 dan tetap stabil hingga akhir *epoch* 50.

Gambar 4.3b menunjukkan perubahan *loss* model seiring dengan bertambahnya *epoch*. Di awal *training*, model memiliki *loss* yang tinggi, lalu terjadi penurunan *loss* secara pesat pada pertengahan *epoch* 0 dan 10. Pada pertengahan *epoch* 20 dan 30, *validation loss* mengalami sedikit kenaikan. *Validation* dan *train loss* mulai konvergen dari *epoch* 30 dan tetap stabil hingga akhir *epoch* 50.

Dari kedua grafik tersebut, didapatkan bahwa model dapat mengenali data yang diberikan dengan baik. Proses pelatihan model skenario uji coba 3 memakan waktu 52 detik untuk setiap *epoch*, dengan total waktu pelatihan 2633 detik atau 44 menit. Model ketiga memiliki akurasi *test* 93,33% dengan presisi *test* 95,24%.

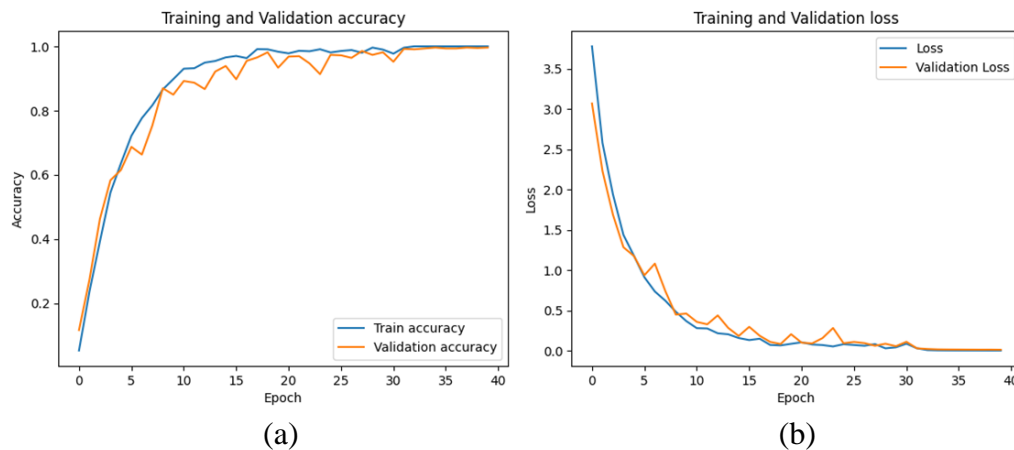
4.1.4 Analisis Hasil Skenario 4

Gambar 4.4a dan 4.4b merupakan grafik akurasi dan *loss* pelatihan model dengan skenario uji coba 4. Pada uji coba ini menggunakan 40 *epoch*, 8 *multihead*, dan 1 *encoder layer*. Gambar 4.4a menunjukkan perubahan akurasi model seiring dengan bertambahnya *epoch*. Di awal *training*, model memiliki akurasi yang rendah, lalu terjadi peningkatan akurasi secara pesat pada pertengahan *epoch* 0 dan 10. Pada *epoch* 35, akurasi *validation* mulai konvergen dengan akurasi *train* dan tetap stabil hingga *epoch* 40.

Gambar 4.4b menunjukkan perubahan *loss* model seiring dengan bertambahnya *epoch*.

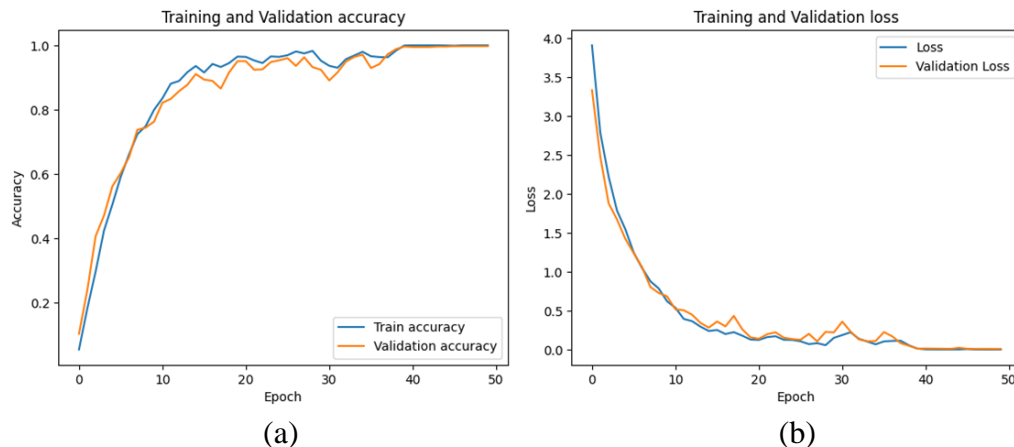
Di awal *training*, model memiliki *loss* yang tinggi, lalu terjadi penurunan *loss* secara pesat pada pertengahan *epoch* 0 dan 10. Pada pertengahan *epoch* 20 dan 30, *validation loss* masih belum stabil dibandingkan dengan *train loss*. *Validation* dan *train loss* mulai konvergen dari *epoch* 35 dan tetap stabil hingga *epoch* 40.

Dari kedua grafik tersebut, didapatkan bahwa model dapat mengenali data yang diberikan dengan baik. Proses pelatihan model skenario uji coba 4 memakan waktu 52 detik untuk setiap *epoch*, dengan total waktu pelatihan 2060 detik atau 34 menit. Model keempat memiliki akurasi 96,00% dengan presisi *test* 96,64%.



Gambar 4.4 (a) Grafik Akurasi *Train* dan *Validation*, (b) Grafik *Train* dan *Validation Loss* dari Skenario Uji Coba 4

4.1.5 Analisis Hasil Skenario 5



Gambar 4.5 (a) Grafik Akurasi *Train* dan *Validation*, (b) Grafik *Train* dan *Validation Loss* dari Skenario Uji Coba 5

Gambar 4.5a dan 4.5b adalah gambar grafik akurasi dan *loss* pelatihan model dengan skenario uji coba 5. Pada uji coba ini menggunakan 50 *epoch*, 4 *multihead*, dan 2 *encoder layer*. Gambar 4.5a menunjukkan perubahan akurasi model seiring dengan bertambahnya *epoch*. Di awal *training*, model memiliki akurasi yang rendah, lalu terjadi peningkatan akurasi secara pesat pada pertengahan *epoch* 0 dan 10. Dari *epoch* 10 hingga 20 akurasi *train* dan *validation* mengalami perkembangan fluktuatif. Kemudian pada *epoch* 40 hingga 50, akurasi *train* dan *validation* menjadi konvergen.

Pada Gambar 4.5b, grafik menunjukkan perubahan *loss* model seiring dengan bertambahnya *epoch*. Di awal *training*, model memiliki *loss* yang tinggi, lalu terjadi penurunan

loss secara pesat pada pertengahan *epoch* 0 dan 10. Grafik perubahan *train* dan *validation loss* cenderung lebih stabil dibandingkan dengan perubahan akurasi *train* dan *validation*. *Validation* dan *train loss* mulai konvergen dari *epoch* 40 dan tetap stabil hingga *epoch* 50.

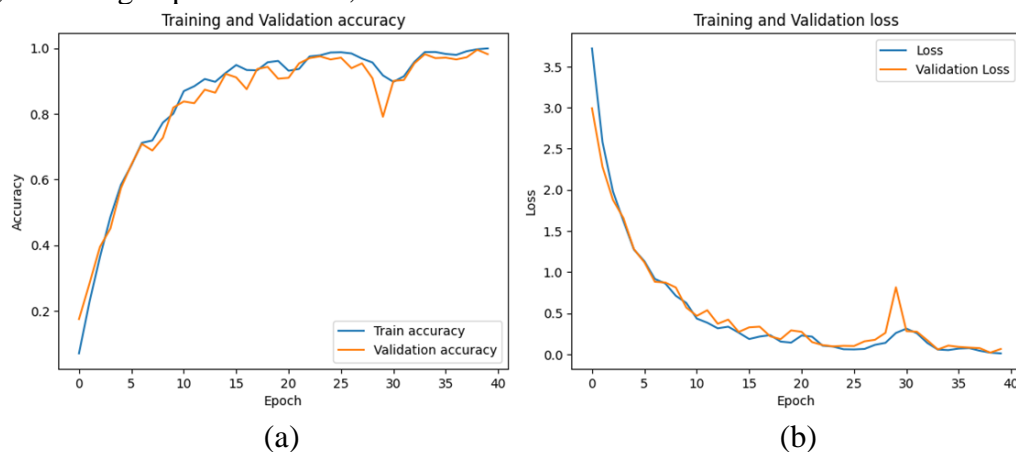
Dari kedua grafik tersebut, didapatkan bahwa model dapat mengenali data yang diberikan meskipun belum akurat. Proses pelatihan model skenario uji coba 5 memakan waktu 59 detik untuk setiap *epoch*, dengan total waktu pelatihan 2938 detik atau 48 menit. Model kelima memiliki akurasi *test* 97,33% dengan presisi *test* 97.33%.

4.1.6 Analisis Hasil Skenario 6

Gambar 4.6a dan 4.6b adalah gambar grafik akurasi dan *loss* pelatihan model dengan skenario uji coba 6. Pada uji coba ini menggunakan 40 *epoch*, 4 *multihead*, dan 2 *encoder layer*. Gambar 4.6a menunjukkan perubahan akurasi model seiring dengan bertambahnya *epoch*. Di awal *training*, model memiliki akurasi yang rendah, lalu terjadi peningkatan akurasi secara pesat pada pertengahan *epoch* 0 dan 10. Dari *epoch* 10 hingga 20 akurasi *train* dan *validation* mengalami perkembangan fluktuatif. Kemudian pada *epoch* 20 hingga 50, perkembangan akurasi *train* dan *validation* relatif stabil walau tidak dapat konvergen hingga akhir *epoch*. Akurasi *validation* mengalami *drop* tajam pada *epoch* 30.

Pada Gambar 4.6b, grafik menunjukkan perubahan *loss* model seiring dengan bertambahnya *epoch*. Di awal *training*, model memiliki *loss* yang tinggi, lalu terjadi penurunan *loss* secara pesat pada pertengahan *epoch* 0 dan 10. Grafik perubahan *train* dan *validation loss* cenderung lebih stabil dibandingkan dengan perubahan akurasi *train* dan *validation* meskipun tidak stabil hingga *epoch* 40. *Loss validation* mengalami *spike* (loncatan) tajam pada *epoch* 30.

Dari kedua grafik tersebut, didapatkan bahwa model dapat mengenali data yang diberikan. Proses pelatihan model skenario uji coba 6 memakan waktu 59 detik untuk setiap *epoch*, dengan total waktu pelatihan 2400 detik atau 40 menit. Model keenam memiliki akurasi *test* 96,67% dengan presisi *test* 96,67%.



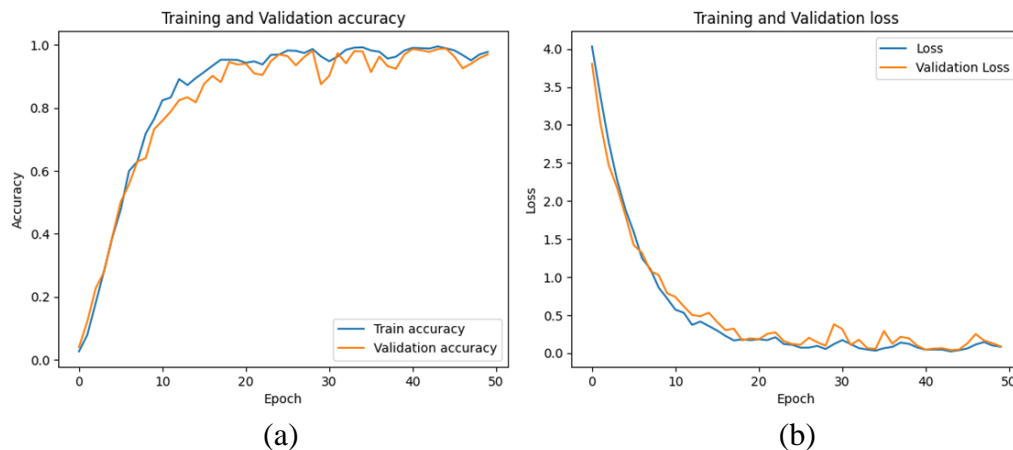
Gambar 4.6 (a) Grafik Akurasi *Train* dan *Validation*, (b) Grafik *Train* dan *Validation Loss* dari Skenario Uji Coba 6

4.1.7 Analisis Hasil Skenario 7

Gambar 4.7a dan 4.7b adalah gambar grafik akurasi dan *loss* pelatihan model dengan skenario uji coba 7. Pada uji coba ini menggunakan 50 *epoch*, 8 *multihead*, dan 2 *encoder layer*. Gambar 4.7a menunjukkan perubahan akurasi model seiring dengan bertambahnya *epoch*. Di awal *training*, model memiliki akurasi yang rendah, lalu terjadi peningkatan akurasi secara pesat pada pertengahan *epoch* 0 dan 10. Dari *epoch* 10 hingga 20 akurasi *train* dan *validation* mengalami perkembangan fluktuatif. Kemudian pada *epoch* 20 hingga 50, akurasi *train* dan *validation* mengalami fluktuasi dengan kisaran akurasi 87% hingga 98%.

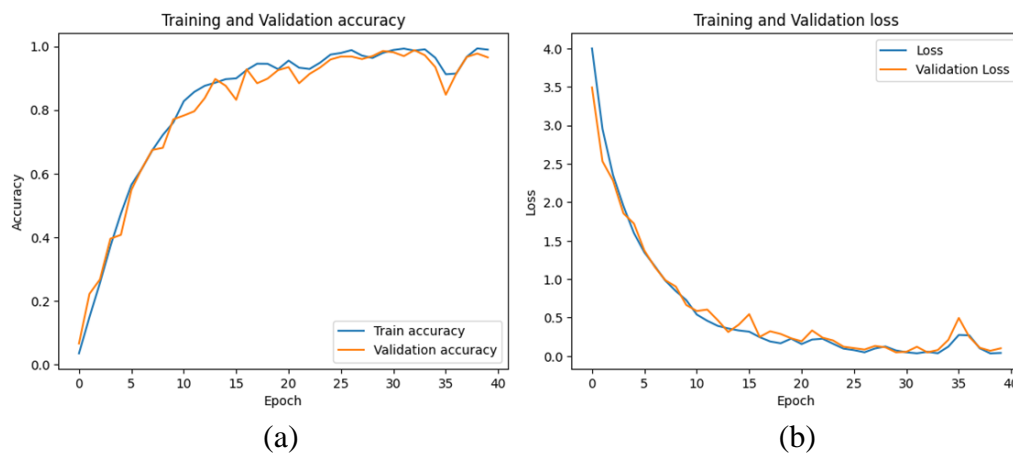
Pada Gambar 4.7b, grafik menunjukkan perubahan *loss* model seiring dengan bertambahnya *epoch*. Di awal *training*, model memiliki *loss* yang tinggi, lalu terjadi penurunan *loss* secara pesat pada pertengahan *epoch* 0 dan 10. Grafik perubahan *train* dan *validation loss* cenderung lebih stabil dibandingkan dengan perubahan akurasi *train* dan *validation* meskipun tetap fluktuatif hingga *epoch* 50.

Dari kedua grafik tersebut, didapatkan bahwa model dapat mengenali data yang diberikan meskipun belum akurat. Proses pelatihan model skenario uji coba 7 memakan waktu 110 detik untuk setiap *epoch*, dengan total waktu pelatihan 5479 detik atau 91 menit. Model ketujuh memiliki akurasi *test* 86,00% dengan presisi *test* 87,16%.



Gambar 4.7 (a) Grafik Akurasi *Train* dan *Validation*, (b) Grafik *Train* dan *Validation Loss* dari Skenario Uji Coba 7

4.1.8 Analisis Hasil Skenario 8



Gambar 4.8 (a) Grafik Akurasi *Train* dan *Validation*, (b) Grafik *Train* dan *Validation Loss* dari Skenario Uji Coba 8

Gambar 4.8a dan 4.8b adalah gambar grafik akurasi dan *loss* pelatihan model dengan skenario uji coba 8. Pada uji coba ini menggunakan 50 *epoch*, 8 *multihead*, dan 2 *encoder layer*. Gambar 4.8a menunjukkan perubahan akurasi model seiring dengan bertambahnya *epoch*. Di awal *training*, model memiliki akurasi yang rendah, lalu terjadi peningkatan akurasi secara pesat pada pertengahan *epoch* 0 dan 10. Dari *epoch* 10 hingga 20 akurasi *train* dan *validation* mengalami perkembangan fluktuatif. Kemudian pada *epoch* 20 hingga 30, akurasi *train* dan *validation* mengalami fluktuasi dengan kisaran akurasi 87% hingga 98%.

Pada Gambar 4.8b, grafik menunjukkan perubahan *loss* model seiring dengan

bertambahnya *epoch*. Di awal *training*, model memiliki *loss* yang tinggi, lalu terjadi penurunan *loss* secara pesat pada pertengahan *epoch* 0 dan 10. Grafik perubahan *train* dan *validation loss* cenderung lebih stabil dibandingkan dengan perubahan akurasi *train* dan *validation* meskipun tetap fluktuatif hingga *epoch* 35. Pada *epoch* 35 hingga 40, *train* dan *validation loss* menjadi konvergen.

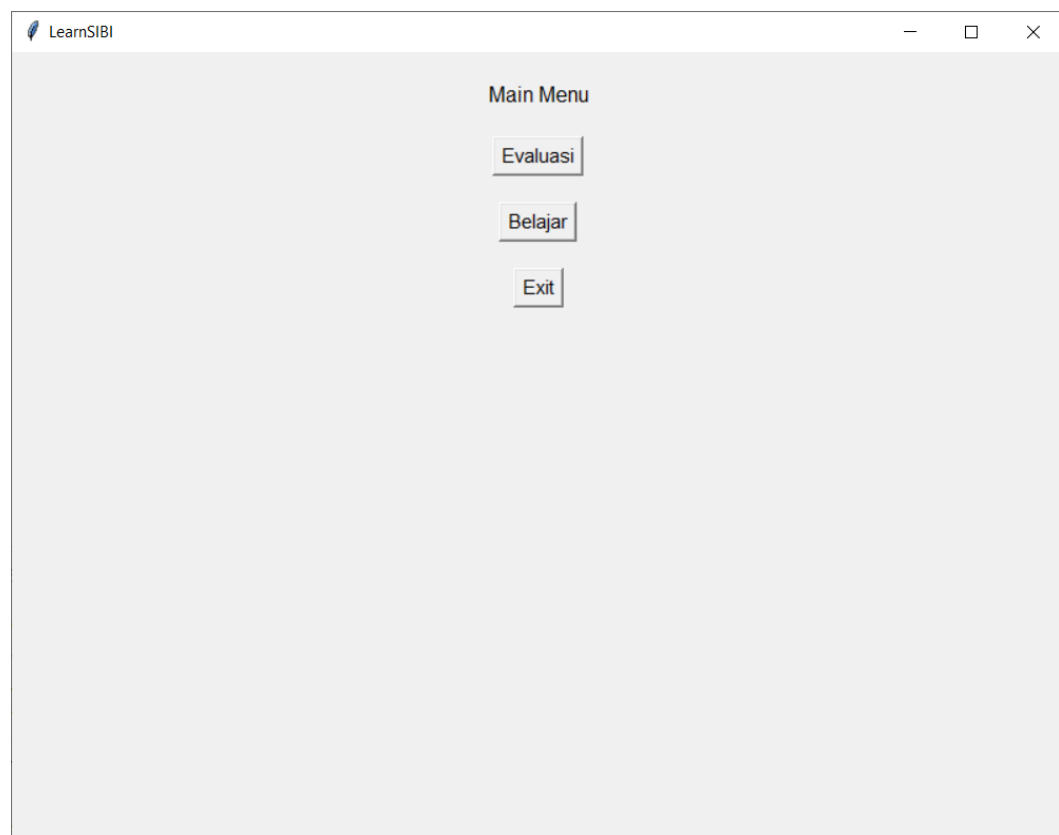
Dari kedua grafik tersebut, didapatkan bahwa model dapat mengenali data yang diberikan meskipun belum akurat. Proses pelatihan model skenario uji coba 8 memakan waktu 110 detik untuk setiap *epoch*, dengan total waktu pelatihan 4453 detik atau 74 menit. Model ketujuh memiliki akurasi *test* 88,67% dengan presisi *test* 90,48%.

4.1.9 Analisis Uji Coba

Dari uji coba yang telah dilakukan, didapatkan bahwa skenario ke-1 memiliki hasil terbaik dengan akurasi tes sebesar 100,00% dan presisi tes 100,00%. Skenario tersebut memiliki parameter *epoch* sebanyak 50 dengan 4 *multihead* dan 1 *encoder layer*. Sedangkan untuk hasil terburuk terdapat pada skenario ketujuh yang memiliki akurasi tes 86,00% dan presisi test 87,16%. Parameter pada skenario tersebut adalah 50 *epoch*, 8 *multihead* dan 2 *encoder layer*.

4.2 Analisis Aplikasi Pembelajaran Bahasa Isyarat

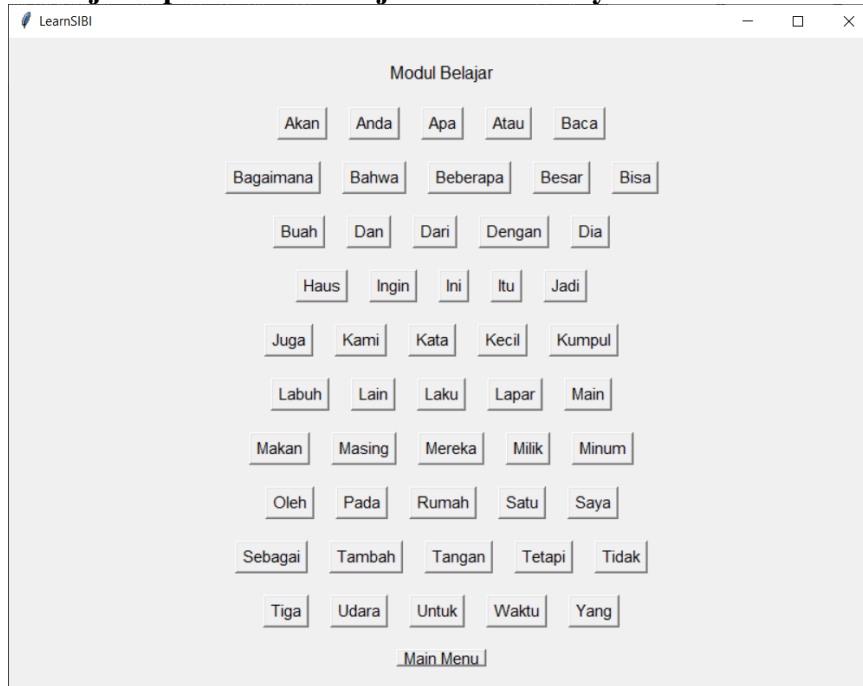
Pada bagian ini akan dilakukan analisis terhadap aplikasi pembelajaran bahasa isyarat. Analisis ini bertujuan untuk menggali informasi tentang berbagai aspek yang terkait dengan aplikasi, termasuk antarmuka pengguna, fitur, dan fungsi yang disediakan. Dengan melakukan analisis yang komprehensif, dapat diketahui sejauh mana aplikasi ini mampu mendukung proses pembelajaran bahasa isyarat.



Gambar 4.9 Tampilan Antarmuka Menu Utama Aplikasi Pembelajaran Bahasa Isyarat

Gambar 4.9 adalah antarmuka untuk menu utama pada aplikasi pembelajaran bahasa isyarat. Menu utama ini berfungsi sebagai penghubung modul evaluasi dan modul belajar. Terdapat *title* menu dengan judul *Main Menu*, serta tombol Evaluasi dan Belajar. Ketika tombol Evaluasi dan Belajar dipencet, pengguna diarahkan ke menu untuk modul evaluasi atau belajar. Aplikasi akan ditutup ketika tombol Exit dipilih.

4.2.1. Modul Belajar Aplikasi Pembelajaran Bahasa Isyarat



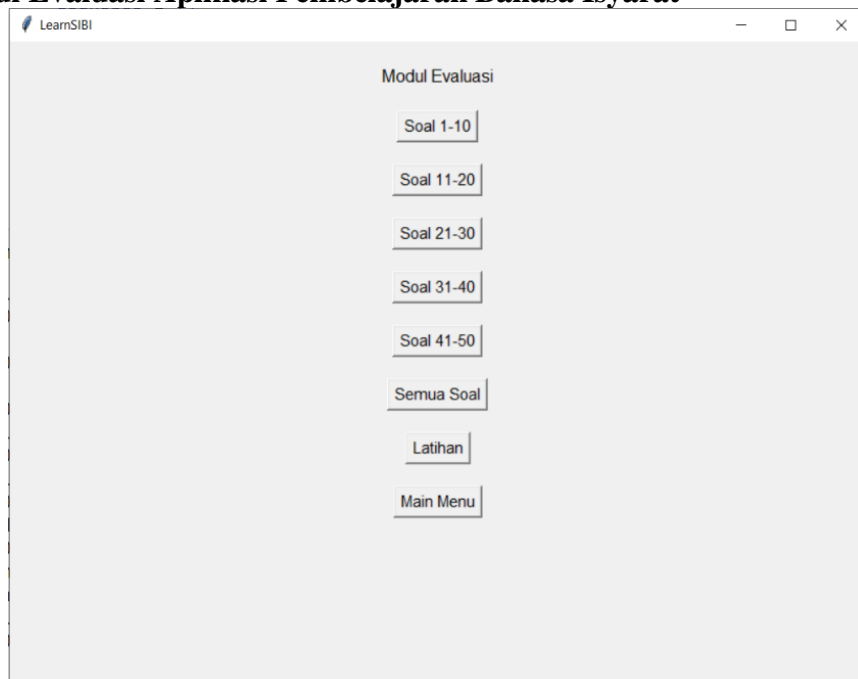
Gambar 4.10 Tampilan Antarmuka Menu Modul Belajar Aplikasi Pembelajaran Bahasa Isyarat

Gambar 4.10 adalah antarmuka untuk menu utama modul belajar. Menu ini berfungsi sebagai menu untuk memutar video peragaan bahasa isyarat. Terdapat *title* menu dengan judul *Modul Belajar*, serta tombol untuk memutar video. Setiap tombol yang ada pada modul merupakan *class* bahasa isyarat yang digunakan dalam aplikasi pembelajaran. Ketika salah satu tombol dipencet, muncul jendela baru yang memutar video seperti pada Gambar 4.11. Jendela pemutaran video ditutup secara otomatis ketika video selesai atau ketika tombol 'q' dipencet. Pengguna dapat kembali ke menu utama dengan tombol Main Menu.

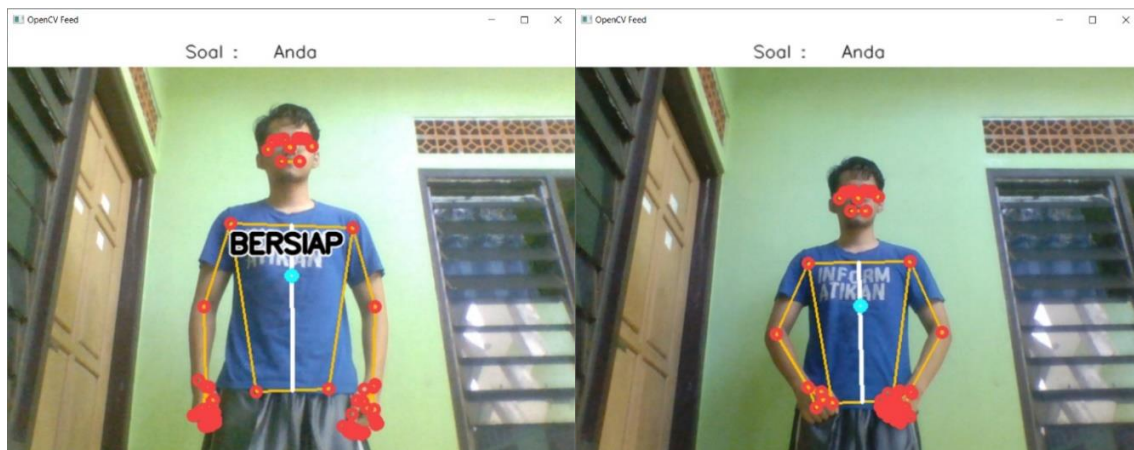


Gambar 4.11 Jendela Pemutar Video Peragaan Bahasa Isyarat

4.2.2. Modul Evaluasi Aplikasi Pembelajaran Bahasa Isyarat



Gambar 4.12 Tampilan Antarmuka Menu Modul Evaluasi Aplikasi Pembelajaran Bahasa Isyarat



Gambar 4.13 Tampilan Jendela Proses Evaluasi Aplikasi Pembelajaran Bahasa Isyarat

Gambar 4.12 adalah antarmuka untuk menu utama modul evaluasi. Menu ini berfungsi sebagai menu untuk memilih soal yang diujikan pada proses evaluasi. Terdapat *title* menu dengan judul *Modul Evaluasi*, serta tombol memilih soal evaluasi. Ketika salah satu tombol soal dipencet, muncul jendela baru untuk proses evaluasi bahasa isyarat. Pengguna dapat kembali ke menu utama dengan tombol Main Menu.

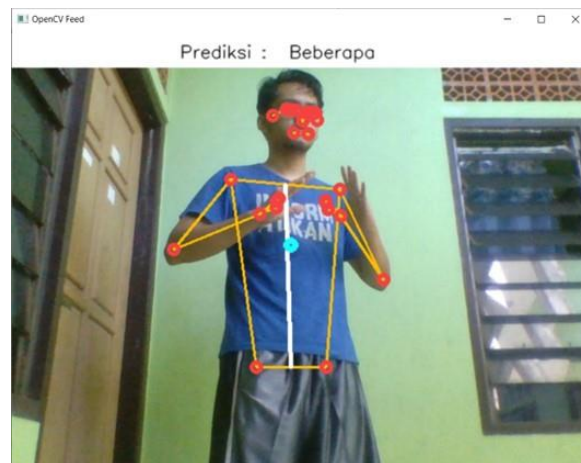
Pada Gambar 4.13 terdapat tampilan video *real time* pengguna. Di bagian atas, terdapat antarmuka untuk soal. Di awal mulainya proses evaluasi, terdapat waktu jeda 2,5 detik untuk pengguna bersiap-siap. Ketika data *keypoints* dari 120 *frame* terkumpul, jendela proses evaluasi akan ditutup, lalu data *keypoints* digunakan untuk memprediksi bahasa isyarat. Hasil evaluasi ditampilkan pada jendela baru seperti pada Gambar 4.14.

Gambar 4.15 merupakan tampilan jendela latihan aplikasi pembelajaran bahasa isyarat. Pada bagian latihan, proses berlangsung secara kontinyu hingga tombol 'q' dipencet. Sama seperti pada proses evaluasi, aplikasi memerlukan data *keypoints* dari 120 *frame* untuk

melakukan prediksi. Untuk prediksi berikutnya, data *keypoints* adalah 120 *frame* terakhir yang disimpan. Hasil prediksi ditampilkan pada bagian atas jendela latihan.



Gambar 4.14 Tampilan Jendela Hasil Evaluasi Aplikasi Pembelajaran Bahasa Isyarat



Gambar 4.15 Tampilan Jendela Latihan Aplikasi Pembelajaran Bahasa Isyarat

4.3 Pembahasan

Pada subbab ini, model dan aplikasi pembelajaran bahasa isyarat dibahas secara mendalam berdasarkan hasil penelitian yang telah diuji. Pembahasan model dan aplikasi mencakup kinerja, kendala, kekurangan serta saran untuk mengembangkan model dan aplikasi pembelajaran bahasa isyarat.

4.3.1 Pembahasan Model Pengenalan Bahasa Isyarat

Uji coba model menggunakan data normal dan data diaugmentasi. Sebelum uji coba ini telah dilakukan percobaan tanpa data augmentasi. Akan tetapi uji coba dengan data normal saja mendapatkan akurasi dan presisi yang kurang baik. Selain itu uji coba tanpa data augmentasi mengalami *overfitting* karena kurangnya data *train*, *validation* dan *test*. Sehingga hasil uji coba yang digunakan pada analisis dan pembahasan tugas akhir adalah hasil uji coba dengan data normal dan augmentasi. Untuk hasil uji coba tanpa data augmentasi dapat dilihat pada tabel 4.2. Sedangkan untuk grafik proses *training* tanpa data augmentasi dapat dilihat pada bagian Lampiran.

Tabel 4.2 Tabel Skenario dan Hasil Uji Coba Tanpa Data Augmentasi

No	Skenario	Epoch	Multihead	Encoder Layer	Test Accuracy (%)	Test Precision (%)
1	Normal	50	4	1	59,33	67,94
2	Normal	40	4	1	56,67	71,43
3	Penambahan Multihead	50	8	1	60,67	67,41
4	Penambahan Multihead	40	8	1	52,67	63,20
5	Penambahan Encoder Layer	50	4	2	59,33	66,42
6	Penambahan Encoder Layer	40	4	2	53,33	59,70
7	Penambahan Multihead dan Encoder Layer	50	8	2	54,00	50,45
8	Penambahan Multihead dan Encoder Layer	40	8	2	48,67	53,28

Dari 8 skenario uji coba yang dilakukan pada tahap pelatihan model dengan data normal dan augmentasi, tidak ditemukan kendala yang mengganggu jalannya proses. Lamanya waktu yang dibutuhkan model dalam pelatihan bergantung pada hardware yang digunakan, tingkat kompleksitas model, serta jumlah data yang digunakan. Pelatihan pada uji coba ini menggunakan CPU bawaan dari laptop. Waktu pelatihan tersebut dapat dikurangi dengan mengganti hardware pelatihan menggunakan GPU. Skenario uji coba Normal memiliki tingkat kompleksitas rendah, sehingga waktu *train* tergolong cepat dengan kisaran 18-23 menit. Skenario Penambahan *Multihead*, dan Penambahan *Encoder Layer* memiliki tingkat kompleksitas sedang dengan waktu *train* 33-48 menit. Untuk Skenario Penambahan *Multihead* dan *Encoder* layer memiliki tingkat kompleksitas tinggi dengan waktu *train* 74-91 menit.

Dari perbandingan kinerja model pada Gambar 4.16 didapatkan bahwa meningkatkan kompleksitas model tidak memberikan kenaikan pada akurasi dan presisi. Pada Skenario 7 dan 8 yang memiliki kompleksitas tinggi justru mendapatkan akurasi dan presisi dibandingkan dengan skenario lain. Model skenario 5 dipilih untuk digunakan pada aplikasi karena memiliki akurasi yang baik dengan waktu prediksi yang dapat ditoleransi.

Meskipun model memiliki akurasi dan presisi yang tinggi pada saat pelatihan, saat diujikan secara *real time* model belum tentu mendapatkan prediksi yang tepat. Hal ini disebabkan oleh kondisi lingkungan yang tidak sama seperti kondisi lingkungan data latih.

Kondisi optimal yang dapat dicapai untuk mendeteksi *keypoints* dengan baik adalah dengan kondisi jarak pemeraga dan kamera diantara 0.75-1 meter, pencahayaan merata dengan cahaya latar belakang tidak terlalu terang, serta resolusi minimal kamera 720p. Untuk *detection* dan *tracking threshold* pada MediaPipe dapat di-set secara manual melalui *source code* yang digunakan, pada pengujian ini *detection* dan *tracking threshold* adalah 0.6 dan 0.5. Ketika kondisi optimal tercapai, salah satu indikatornya adalah *outline* kerangka pada badan terlihat jelas dan sesuai dengan postur badan pemeraga. Gambar 4.17 merupakan contoh kondisi optimal dalam deteksi *keypoints*. Deteksi menggunakan video dari data *test* dengan class “Tangan”.

```

5/5 [=====] - 0s 89ms/step
Model 1 Accuracy: 1.0000
Model 1 Precision: 1.0000

5/5 [=====] - 0s 86ms/step
Model 2 Accuracy: 0.9867
Model 2 Precision: 1.0000

5/5 [=====] - 1s 157ms/step
Model 3 Accuracy: 0.9333
Model 3 Precision: 0.9524

5/5 [=====] - 1s 151ms/step
Model 4 Accuracy: 0.9600
Model 4 Precision: 0.9664

5/5 [=====] - 1s 187ms/step
Model 5 Accuracy: 0.9733
Model 5 Precision: 0.9733

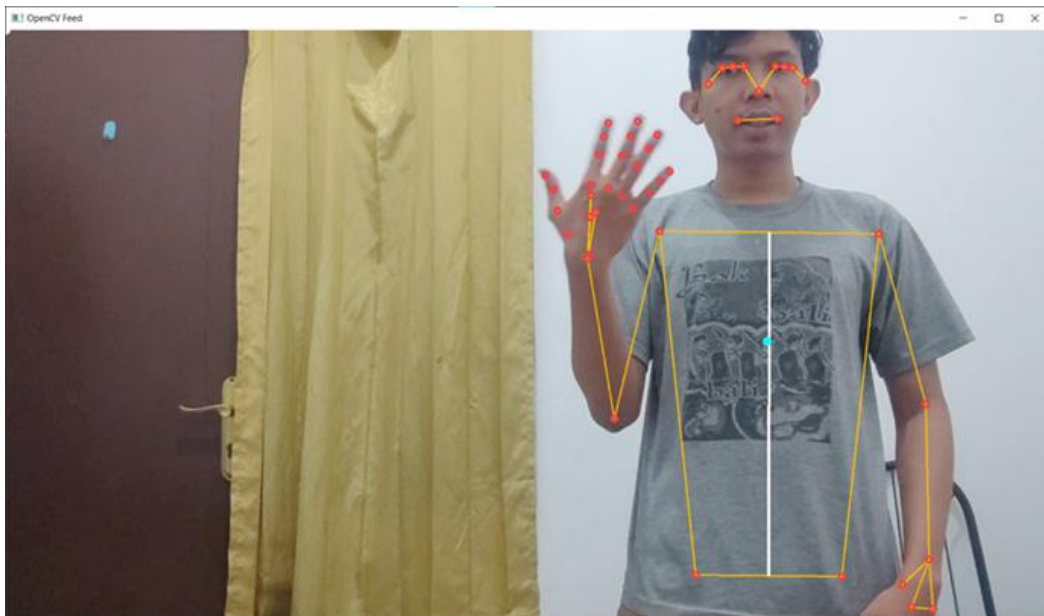
5/5 [=====] - 1s 173ms/step
Model 6 Accuracy: 0.9667
Model 6 Precision: 0.9667

5/5 [=====] - 2s 312ms/step
Model 7 Accuracy: 0.8600
Model 7 Precision: 0.8716

5/5 [=====] - 2s 317ms/step
Model 8 Accuracy: 0.8867
Model 8 Precision: 0.9048

```

Gambar 4.16 Perbandingan Kinerja Antar Model



Gambar 4.17 Kondisi Optimal Deteksi *Keypoints*

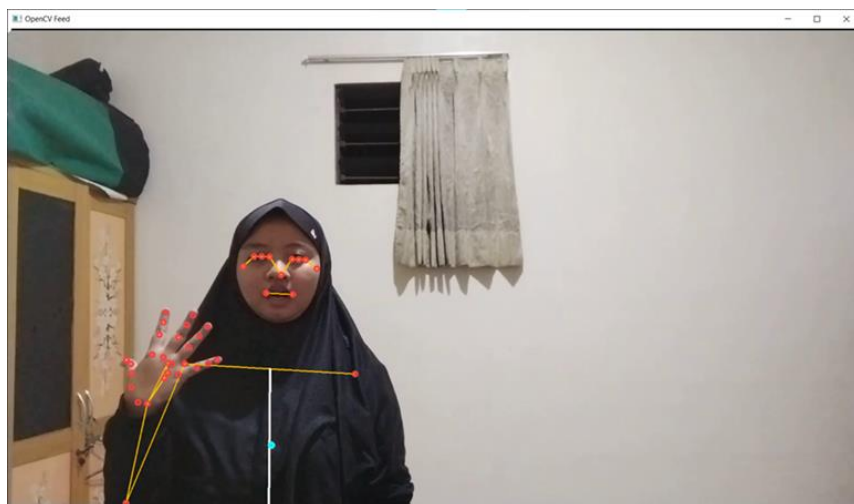
Gambar 4.19 merupakan contoh kondisi yang tidak optimal. Jarak antar kamera dan pemeraga terlalu dekat, sehingga *keypoints* badan dan tangan bagian kiri tidak terdeteksi dengan baik. Hasil prediksi akan bernilai kosong atau tidak ditampilkan ketika model tidak dapat memprediksi *keypoints* yang diberikan.

Model memiliki kekurangan dalam mengenali gestur bahasa isyarat dengan gestur yang

mirip. Sebagai contoh untuk *class* “Tiga” dan “Dia”, pada *class* tersebut memiliki kesamaan gestur mengangkat tangan kanan yang mengarah ke depan. Model juga memiliki kesulitan mengenali *keypoints* pemeraga ketika pakaian pemeraga berwarna gelap dan longgar, ketika *background* pemeraga tidak polos atau ramai, serta ketika pencahayaan terlalu gelap atau terang.



Gambar 4.18 Hasil Prediksi Kondisi Optimal

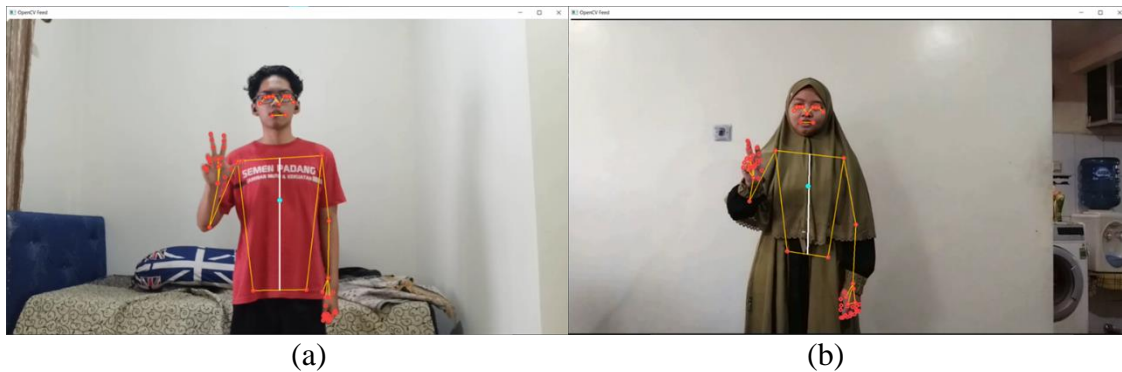


Gambar 4.19 Kondisi Tidak Optimal Deteksi *Keypoints*

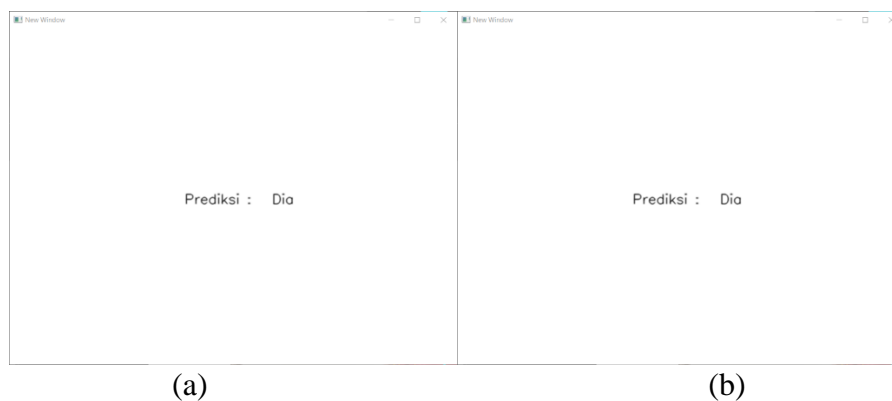
Saran untuk mengatasi masalah tersebut untuk pengembangan di masa depan adalah dengan menambah jumlah data yang digunakan dalam *training*, *validation* dan *testing*. Namun perlu diperhatikan juga untuk penambahan data diperlukan variasi dengan kondisi optimal seperti pada gambar 4.17.



Gambar 4.20 Hasil Prediksi Kondisi Tidak Optimal



Gambar 4.21 (a) Video peraga *class Tiga*, (b) Video peraga *class “Dia”*



Gambar 4.22 (a) Hasil prediksi peraga *class Tiga*, (b) Hasil prediksi peraga *class “Dia”*

4.3.2 Pembahasan Aplikasi Pembelajaran Bahasa Isyarat

Secara keseluruhan, aplikasi sudah berjalan lancar dan sesuai dengan rancangan awal. Model pengenalan bahasa isyarat yang akan digunakan adalah model dari skenario latihan ke-dengan akurasi sebesar 99,87% dan presisi 100%. Model tersebut dapat digunakan dalam pengenalan bahasa isyarat oleh modul evaluasi aplikasi pembelajaran tanpa kendala. Kemudian pada modul belajar, video yang diputarkan tidak memiliki masalah apapun. Masalah kecil yang dijumpai pada proses evaluasi adalah *frame rate* video. Seharusnya *frame rate* video berada pada 30fps, sedangkan pada proses evaluasi video tidak mencapai 30fps. Modul evaluasi

aplikasi dikembangkan dari *source code* pengujian model, sehingga kekurangan modul sama dengan kekurangan model ketika tahap pengujian.

Modul evaluasi memiliki kemampuan untuk mengenali gestur bahasa isyarat secara efektif ketika mayoritas atau seluruh *keypoints* tubuh dapat terdeteksi. Beberapa faktor yang mempengaruhi deteksi *keypoints* meliputi jarak antara pengguna dengan kamera, tingkat pencahayaan di sekitar pengguna, resolusi kamera yang digunakan untuk menangkap *keypoints*, serta ambang batas (*threshold*) *detection* dan *tracking* pada MediaPipe. Untuk mencapai kondisi optimal dalam mendeteksi *keypoints* dengan akurat, disarankan agar jarak antara pengguna dan kamera berada dalam rentang 0.75 hingga 1 meter, pencahayaan merata dengan cahaya latar belakang yang tidak terlalu terang, dan penggunaan kamera dengan resolusi minimal 720p. Selain itu, dalam pengujian ini, *threshold detection* dan *tracking* pada MediaPipe diatur pada nilai 0.6 dan 0.5 secara berurutan.

Saran untuk pengembangan aplikasi adalah dengan memperbaiki antarmuka aplikasi. Antarmuka aplikasi yang sekarang dapat ditenahi tata letak, bentuk, serta tampilannya. Kemudian antarmuka dapat dibuat menjadi lebih *streamline* menjadi satu jendela antarmuka saja tanpa membuat jendela antarmuka baru untuk menampilkan modul. Aplikasi dapat dikembangkan kembali dengan bahasa pemrograman dan platform yang berbeda. Sebagai contoh dengan menggunakan C# untuk platform Android. Terdapat berbagai *library* dan *framework* yang dapat mempercepat waktu pengembangan, contohnya adalah *game engine* Unity, PyGames, dan Construct.

[Halaman ini sengaja dikosongkan]

BAB V KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil perancangan dan uji coba pada tugas akhir ini, didapatkan bahwa arsitektur transformer dapat mengenali data bahasa isyarat dalam bentuk video dengan baik. Aplikasi pembelajaran bahasa isyarat yang memanfaatkan arsitektur transformer dapat mengenali video peragaan bahasa isyarat secara *real time* sebagai sarana evaluasi. Selain itu aplikasi pembelajaran bahasa isyarat dapat memberikan sarana pembelajaran melalui pemutaran video peraga. Dari hasil perancangan dan uji coba pada tugas akhir ini dapat disimpulkan bahwa:

- Rancangan model pengenalan bahasa isyarat dengan arsitektur transformer menggunakan variasi transformer *encoder only*. Model memiliki 4 *multihead*, 1 *encoder layer* dan dilatih selama 50 *epoch*.
- Aplikasi pembelajaran bahasa isyarat memiliki menu utama sebagai penghubung modul belajar dan modul evaluasi. Pada modul belajar terdapat video peraga bahasa isyarat yang dapat diputar sebagai contoh peragaan. Di modul evaluasi terdapat fitur untuk menguji kemampuan menggunakan bahasa isyarat.
- Modul Evaluasi menggunakan model pengenalan bahasa isyarat dengan arsitektur transformer *encoder only* untuk mengenali gestur bahasa isyarat. Saat proses evaluasi berlangsung, aplikasi akan mengumpulkan data *keypoints* setiap *frame* dari kamera secara *real time*. Setelah terkumpul data hingga 120 *frame*, data tersebut digunakan untuk memprediksi gestur bahasa isyarat. Gestur dinilai sesuai bila prediksi gestur sama dengan soal yang diberikan.
- Modul Belajar dapat memutar video peraga bahasa isyarat sesuai kata yang dipilih. Ketika video diputarkan, terdapat jendela baru untuk memutar video peraga. Setelah selesai diputar, jendela akan ditutup secara otomatis.
- Aplikasi pembelajaran bahasa isyarat dievaluasi melalui kinerja model pengenalan bahasa isyarat, serta fitur aplikasi yang menunjang pembelajaran bahasa isyarat. Dari evaluasi yang telah dilakukan, model dapat mengenali bahasa isyarat yang diperagakan. Fitur aplikasi untuk memutar video peraga serta evaluasi gestur telah berjalan dengan baik.

5.2 Saran

Dalam pengerjaan tugas akhir ini, terdapat banyak kekurangan yang dapat diperbaiki dan dikembangkan untuk meningkatkan kinerja aplikasi dan model. Pada bagian model, dapat dilakukan perubahan pada arsitektur atau mengganti *hardware* pelatihan model dengan GPU untuk memangkas waktu pelatihan model. Saran berikutnya yang dapat dilakukan untuk mengembangkan model, arsitektur model dapat diubah dan di-*tuning* untuk menemukan model yang lebih optimal. *Dataset* yang digunakan pada model dapat ditambahkan lagi untuk *train*, *test* dan *validation*. Di bagian aplikasi, *user interface* (antarmuka) dapat dibenahi tata letak, bentuk, serta tampilannya. Aplikasi dapat dikembangkan lagi dengan bahasa pemrograman dan *platform* yang berbeda. Sebagai contoh dengan menggunakan C# untuk *platform* Android.

[Halaman ini sengaja dikosongkan]

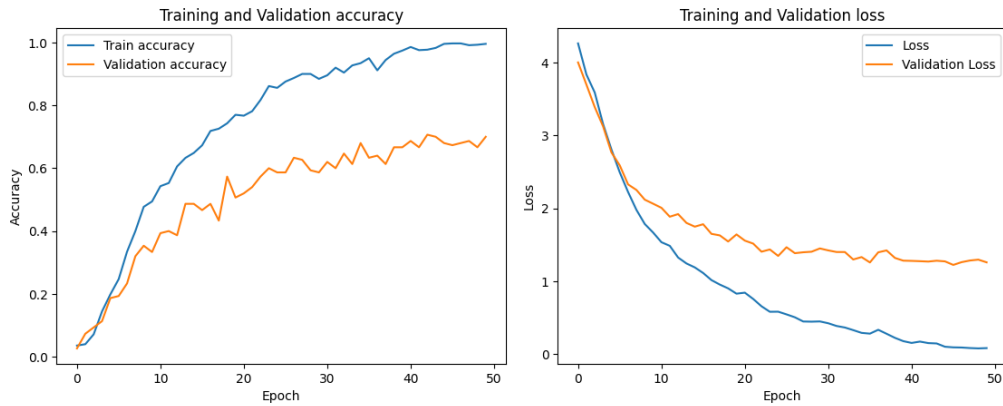
DAFTAR PUSTAKA

- Mandarani, Putri. Putra, Yogi. (2020). "Aplikasi Bahasa Isyarat untuk Tuna Rungu Menggunakan Platform Android". Institut Teknologi Padang. <https://teknoif.itp.ac.id/index.php/teknoif/article/view/40/780>.
- Pradikja, Maharoni Hendra. Tolle, Herman. Brata, Kumang Candra. (2019). "Pengembangan Aplikasi Pembelajaran Bahasa Isyarat Berbasis Android Tablet". Universitas Brawijaya. <https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/1705/645>
- C.K.M.Lee, Kam K.H.Ng, Chun-Hsien Chen, H.C.W.Lau, S.Y.Chung, Tiffany Tsoi. (2021). "American Sign Language Recognition and Training Method with Recurrent Neural Network". <https://www.sciencedirect.com/science/article/pii/S0957417420310745>
- Suharjito. Anderson, Ricky. Wiryana, Fanny. Meita Chandra Ariesta, Gede Putra Kusuma. (2017). Sign Language Recognition Application Systems for Deaf-Mute People: A Review Based on Input-Process-Output. <https://www.sciencedirect.com/science/article/pii/S1877050917320720>
- Suharjito. Gunawan, Herman Thiracitta, Narada. Nugroho, Ariadi. (2018). Sign Language Recognition Using Modified Convolutional Neural Network Model. IEEE. DOI: 10.1109/INAPR.2018.8627014
- Daniels, Steve. Suciati, Nanik. Fathichah, Chastine. (2021). Indonesian Sign Language Recognition using YOLO Method. Institut Teknologi Sepuluh Nopember. https://www.researchgate.net/publication/350082064_Indonesian_Sign_Language_Recognition_using_YOLO_Method/fulltext/608b697e458515d315e6bd95/Indonesian-Sign-Language-Recognition-using-YOLO-Method.pdf
- LeCun. Yann, Bengio. Yoshua, Hinton. Geoffrey (2015). "Deep Learning". Nature. 521 (7553): 436–444.
- Valueva, M.V.; Nagornov, N.N.; Lyakhov, P.A.; Valuev, G.V.; Chervyakov, N.I. (2020). "Application of The Residue Number System to Reduce Hardware Costs of The Convolutional Neural Network Implementation". Mathematics and Computers in Simulation. Elsevier BV. 177: 232–243. doi:10.1016/j.matcom.2020.04.031
- Graves. Alex, Liwicki. Marcus, Fernandez. Santiago, Bertolami. Roman, Bunke, Horst; Schmidhuber, Jürgen (2009). "A Novel Connectionist System for Improved Unconstrained Handwriting Recognition".
- Vaswan. Ashish, Shazeer. Noam, Parmar. Niki, Uszkoreit. Jakob, Jones. Llion, Gomez. Aidan N., Kaiser. Łukasz, Polosukhin. Illia. (2017). "Attention is All You Need". <https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf>
- Le, Khuyen. (2017). *An overview of VGG16 and NiN models*. <https://medium.com/mlearning-ai/an-overview-of-vgg16-and-nin-models-96e4bf398484>. Diakses pada 6 Desember 2022.

- Neto, G. M. R., Junior, G. B., de Almeida, J. D. S., & de Paiva, A. C. (2018). Sign Language Recognition Based on 3D Convolutional Neural Networks. *Image Analysis and Recognition*, 399–407. DOI:10.1007/978-3-319-93000-8_45
- Camgoz, N. C., Koller, Oscar. Hadfield, Simon. Bowden, Richard. (2020). Sign Language Transformers: Joint End-to-end Sign Language Recognition and Translation. <https://doi.org/10.48550/arXiv.2003.13830>
- Fang, Biyi. Co, Jillian. Zhang, Mi. (2018). DeepASL: Enabling Ubiquitous and Non-Intrusive Word and Sentence-Level Sign Language Translation. DOI:10.1145/3131672.3131693
- Krishna, Akhil G., Sandeep, G., (2022). Sign language Recognition using Deep CNN. https://www.researchgate.net/publication/360384816_Sign_language_Recognition_using_Deep_CNN
- Ko, S.-K., Son, J. G., & Jung, H. (2018). Sign language recognition with recurrent neural network using human keypoint detection. *Proceedings of the 2018 Conference on Research in Adaptive and Convergent Systems - RACS '18*. DOI:10.1145/3264746.3264805

LAMPIRAN-LAMPIRAN

Proses *Training* Uji Coba Skenario 1 Tanpa Data Augmentasi

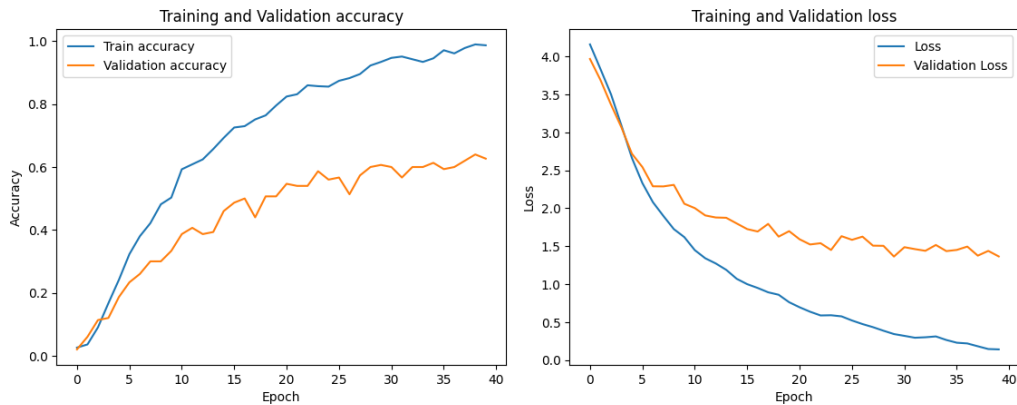


(a)

(b)

(a) Grafik Akurasi *Train* dan *Validation*, (b) Grafik *Train* dan *Validation Loss* dari Skenario Uji Coba 1 Tanpa Data Augmentasi

Proses *Training* Uji Coba Skenario 2 Tanpa Data Augmentasi

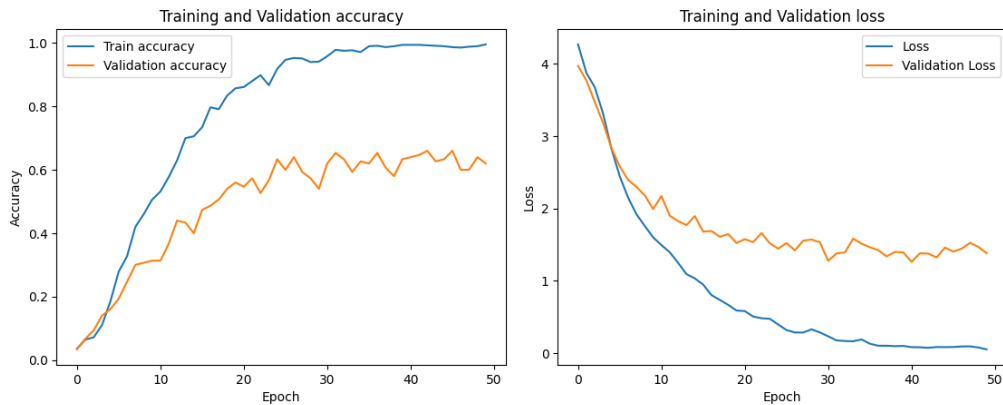


(a)

(b)

(a) Grafik Akurasi *Train* dan *Validation*, (b) Grafik *Train* dan *Validation Loss* dari Skenario Uji Coba 2 Tanpa Data Augmentasi

Proses *Training* Uji Coba Skenario 3 Tanpa Data Augmentasi

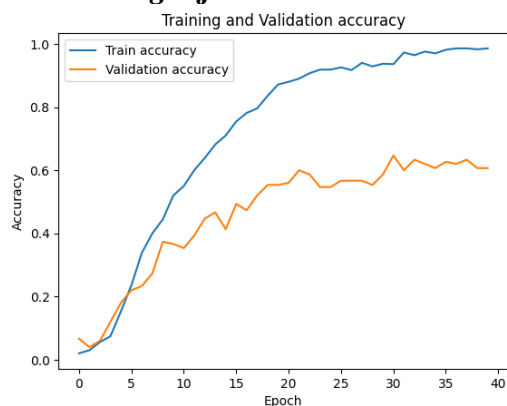


(a)

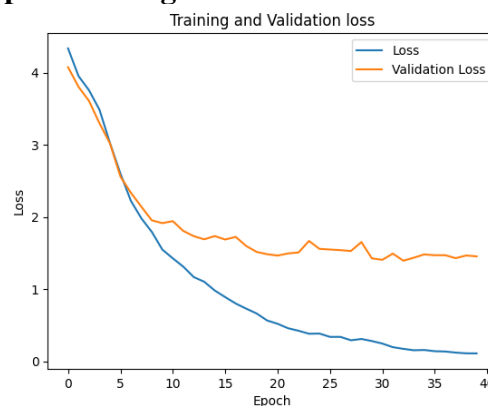
(b)

(a) Grafik Akurasi *Train* dan *Validation*, (b) Grafik *Train* dan *Validation Loss* dari Skenario Uji Coba 3 Tanpa Data Augmentasi

Proses *Training Uji Coba Skenario 4 Tanpa Data Augmentasi*



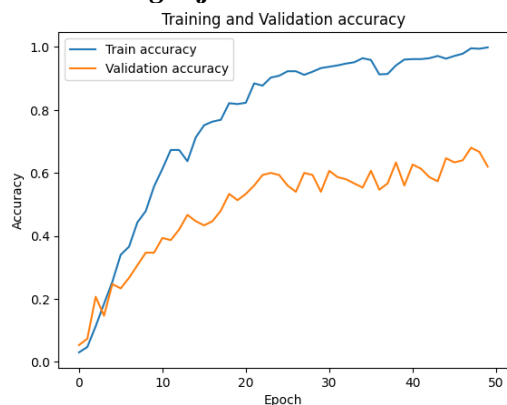
(a)



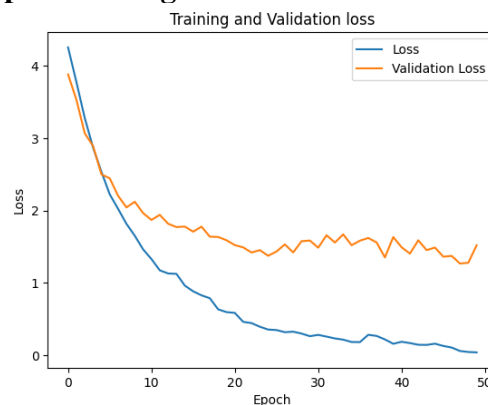
(b)

(a) Grafik Akurasi *Train* dan *Validation*, (b) Grafik *Train* dan *Validation Loss* dari Skenario Uji Coba 4 Tanpa Data Augmentasi

Proses *Training Uji Coba Skenario 5 Tanpa Data Augmentasi*



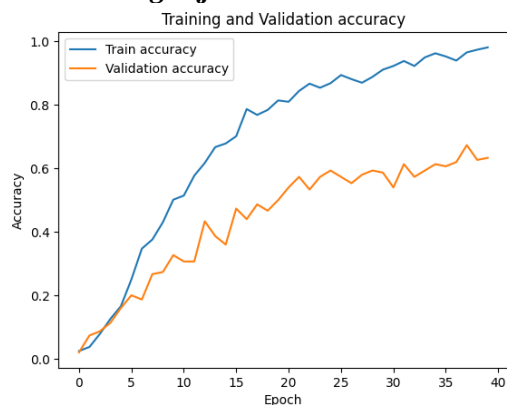
(a)



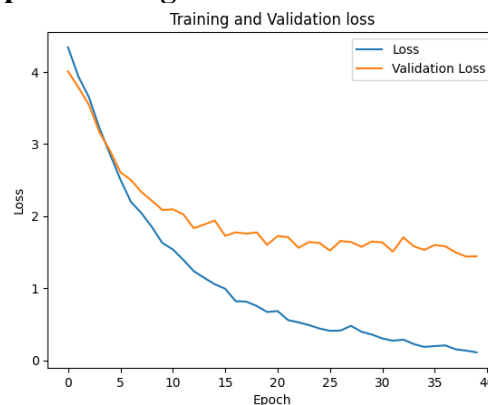
(b)

(a) Grafik Akurasi *Train* dan *Validation*, (b) Grafik *Train* dan *Validation Loss* dari Skenario Uji Coba 5 Tanpa Data Augmentasi

Proses *Training Uji Coba Skenario 6 Tanpa Data Augmentasi*



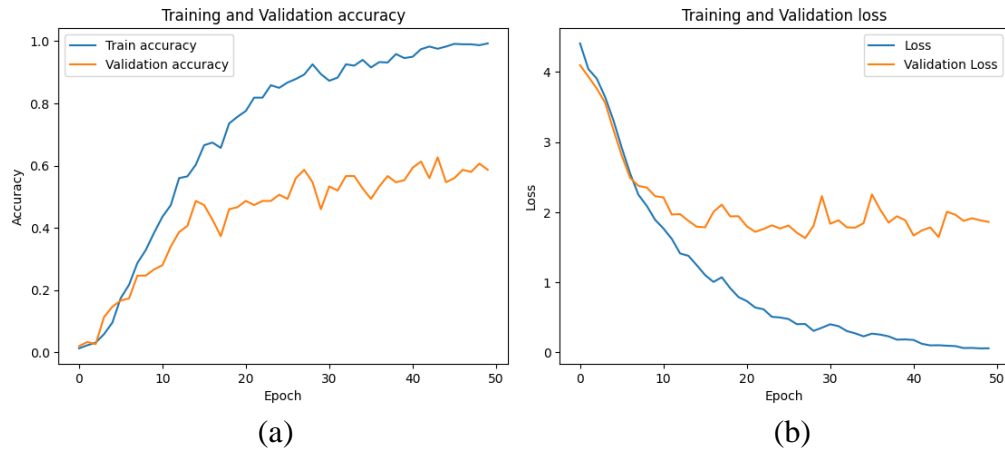
(a)



(b)

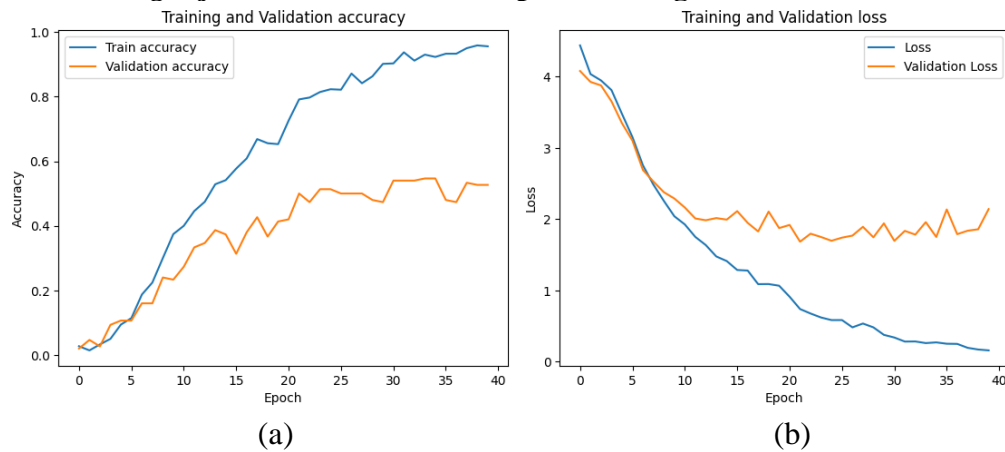
(a) Grafik Akurasi *Train* dan *Validation*, (b) Grafik *Train* dan *Validation Loss* dari Skenario Uji Coba 6 Tanpa Data Augmentasi

Proses *Training* Uji Coba Skenario 7 Tanpa Data Augmentasi



(a) Grafik Akurasi *Train* dan *Validation*, (b) Grafik *Train* dan *Validation Loss* dari Skenario Uji Coba 7 Tanpa Data Augmentasi

Proses *Training* Uji Coba Skenario 8 Tanpa Data Augmentasi



(a) Grafik Akurasi *Train* dan *Validation*, (b) Grafik *Train* dan *Validation Loss* dari Skenario Uji Coba 8 Tanpa Data Augmentasi

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS



Penulis dilahirkan di Padang, 31 Agustus 2001, merupakan anak pertama dari 3 bersaudara. Penulis telah menempuh pendidikan formal yaitu di TK Permata Hati, SD Islam Al Azhar 25 Semarang, SMP Islam Al Azhar 23 Semarang dan SMAN 1 Semarang. Setelah lulus dari SMAN tahun 2019, Penulis mengikuti jalur Mandiri Kemitraan dan diterima di Departemen Teknik Informatika FTEIC - ITS pada tahun 2019 dan terdaftar dengan NRP 05111940000224.

Selama berkuliah di ITS Penulis sempat aktif di UKM Teater Tiyang Alit sebagai pengurus selama satu tahun kepengurusan 2020-2021. Penulis juga aktif di Himpunan Mahasiswa Teknik Computer (HMTC) sebagai anggota pengurus Keprofesian dan Teknologi (Protek) pada tahun 2021. Kemudian pada masa kepengurusan 2022, penulis menjabat sebagai Wakil Kepala Keprofesian dan Teknologi.