Sign Language Translator Using Deep Learning

1st Fikri Putra Hidayat
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
fikzhidayat@student.telkomuniversity.ac.id

2nd Casi Setianingsih
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
setiacasie@telkomuniversity.ac.id

3rd Randy Erfa Saputra
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
resaputra@telkomuniversity.ac.id

***Abstrak*— Proyek akhir ini menyoroti masalah yang mencolok di dalam teman tuli yaitu hambatan komunikasi yang dihadapi oleh teman tuli dengan teman dengar, karena ketidakmampuan dengar, mereka menggunakan bahasa isyarat untuk berkomunikasi. Bentuk komunikasi ini sering kali menyebabkan tantangan besar dalam interaksi mereka dengan populasi pendengar. Untuk mengatasi masalah ini, kami mengusulkan solusi yang tangguh berupa aplikasi yang tidak hanya berfungsi sebagai alat bantu dalam pembelajaran bahasa isyarat SIBI bagi pendengar, tetapi juga sebagai penerjemah bahasa isyarat.**

Solusi yang diusulkan ini memanfaatkan beberapa bidang dalam Teknik Komputer untuk menciptakanaplikasi yang integratif. Dengan menerapkan teknik machine learning LSTM dan framework Mediapipe, sistem secara akan mengenali dan mengartikan bahasa isyarat SIBI.

Kata kunci: SIBI (Sistem Isyarat Bahasa Indonesia), LSTM(Long-Short Term Memory), Mediapipe

I. PENDAHULUAN

Bahasa isyarat memiliki peran penting dalam memfasilitasi komunikasi bagi komunitas teman tuli. Di Indonesia, terdapat dua standar bahasa isyarat, yaitu BISINDO dan SIBI. SIBI dianggap lebih efektif karena telah diresmikan oleh pemerintah dan memiliki struktur kalimat yang lebih terstruktur dengan adanya imbuhan seperti me-, ter-, dan lainnya. Solusi penerjemah manusia untuk berkomunikasi antara teman tuli dan teman dengar tidak selalu praktis dan efektif. Oleh karena itu, di era revolusi industri 4.0, pengembangan sistem translator bahasa isyarat dengan menggunakan *deep learning* menjadi alternatif yang menjanjikan.

Dalam upaya mencapai solusi yang efektif, kami akan memanfaatkan konsep machine learning dengan fokus pada deep learning. Melalui teknologi ini, kami berharap dapat mengembangkan sistem translator bahasa isyarat yang canggih dan akurat. Dengan menggunakan dataset berisi gerakan bahasa isyarat SIBI yang valid, kami akan melatih model deep learning untuk mengenali dan mengklasifikasikan gestur bahasa isyarat dengan presisi tinggi. Implementasi aplikasi ini berbasis Android, sehingga diharapkan akan lebih mudah diakses dan digunakan oleh masyarakat luas, baik oleh teman tuli maupun teman dengar. Pada jurnal ini lebih fokus membahas pembuatan salah satu fitur aplikasi yaitu deteksi SIBI untuk gerakan dinamis.

II. KAJIAN TEORI

Dalam penelitian terkait yang dipublikasikan dalam jurnal IEEE oleh S. Mhatre, S. Joshi, dan H.B. Kulkarni, penggunaan arsitektur Long Short Term Memory (LSTM) telah diusulkan untuk mendeteksi bahasa isyarat. Model ini bertujuan untuk membantu anak-anak dengan gangguan pendengaran atau yang bisu dalam belajar melalui e-learning dengan memanfaatkan American Sign Language (ASL) [1]. Dalam penelitian ini, OpenCV dan Mediapipe Holistic digunakan untuk mengidentifikasi *marker* kunci dari pengguna yang melakukan gestur ASL ke kamera. Data yang dikumpulkan kemudian dilatih pada arsitektur LSTM [2].

Dalam proyek yang dibuat oleh startup SignAll di Amerika, MediaPipe memungkinkan untuk mendeteksi landmark tangan dalam gambar, yang dapat digunakan untuk melokalisir titik kunci dari tangan dan merender efek visual atas tangan tersebut. MediaPipe Hand Landmarker memerlukan model yang telah dilatih dan kompatibel dengan tugas ini. Model ini digunakan untuk mendeteksi landmark tangan dalam gambar dan kemudian digunakan sebagai input untuk jaringan LSTM [3].

Dalam berbagai penelitian, LSTM telah digunakan dalam berbagai aplikasi deteksi selain bahasa isyarat. Salah satu contohnya adalah dalam Human Activity Recognition (HAR). Menurut penelitian yang dipublikasikan dalam jurnal NCBI, berbagai variasi dari jaringan LSTM telah digunakan untuk mengatasi masalah HAR. Misalnya, jaringan LSTM dasar yang disebut Vanilla LSTM digunakan, yang terdiri dari satu layer LSTM tersembunyi dengan dropout dan satu layer dense. Variasi lainnya termasuk menambahkan lebih banyak layer LSTM tersembunyi atau menggabungkan LSTM dengan layer konvolusi untuk menciptakan layer LSTM. Hasilnya, model LSTM dapat dengan akurat mengidentifikasi perilaku dari enam operasi reguler dengan akurasi lebih dari 95% dan skor F1 lebih dari 99% [4].

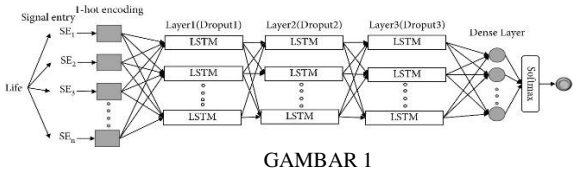
Pada desain sistem akan berfokus pada pengembangan model dan produk akhir berupa *website* Streamlit. Referensi yang dipakai dalam pembuatannya berdasarkan

beberapa penilitian terdahulu seperti penggunaan Streamlit untuk *deploy* deteksi objek dengan OpenCV [5]. Seperti pada proyek yang dikembangkan oleh S. Oluyede Jr. dan N. Landro, proyek mereka memberikan inspirasi dan gambaran bagaimana penggunaan *realtime* kamera dan Mediapipe untuk proses pemberian input ke model *machine learning* [6].

Hal ini menjadi tolak ukur jurnal terutama dalam implementasi model *machine learning* dan integrasi sistem terjemahan SIBI.

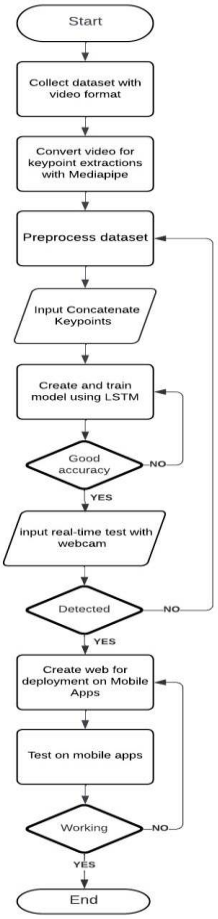
III. METODE

Pada model ini algoritma utamanya adalah LSTM, pada dasarnya LSTM dipakai untuk melatih data gerakan berupa koordinat gestur sesuai dengan label bahasa isyarat yang telah ditentukan. Secara arsitektur LSTM terdiri dari data masukan, proses data, unit LSTM, *dense layer*, *optimizer,* dan output. Berikut gambaran arsitekturnya secara garis besar [7]:



GAMBAR 1
Arsitektur LSTM

A. Alur Pelatihan Model

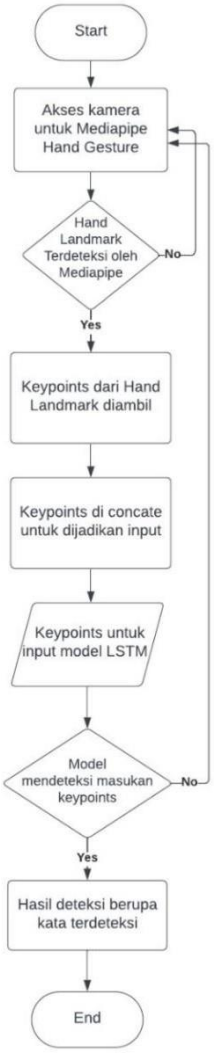


GAMBAR 2
Flowchart Pembuatan Model

Penjelasan *flowchart* :

1. Melakukan koleksi data video dengan PC pengembang dan webcam untuk 26 kata sebanyak 30 kali perkata
2. Mediapipe mengolah video tersebut menjadi koordinat *keypoints* dalam bentuk (x,y,z) sebanyak 126 perframe
3. *Concate* poin kunci tadi menjadi satu kesatuan 30fps menjadi array 30,126
4. Buat model LSTM dan latih sesuai parameter yang dipakai
5. Tes akurasi dengan metode buka kamera langsung dengan Mediapipe sebagai platform untuk praproses data inputan model
6. Apabila bagus maka dilanjutkan dengan pembuatan produk akhir
7. Membuat web dengan Streamlit
8. Uji lokal web streamlit
9. *Deploy* web Streamlit
10. Uji fungsionalitas web Streamlit
11. Uji upload video dan deteksi web Streamlit

B. Alur Sistem Deteksi



GAMBAR 3
Flowchart Sistem Deteksi

Penjelasan *flowchart* :

1. Sistem akan meminta akses kamera baik untuk video
2. Mediapipe mendeteksi gestur video dan mengambil *keypoints* sebagai data masukan
3. Data diproses untuk dimasukan ke model
4. Model menerima masukan
5. Model mendeteksi masukan
6. Model menampilkan hasil deteksi

Proses pengembangan dilakukan melalui perangkat pengembang yang berspesifikasi sebagai berikut:

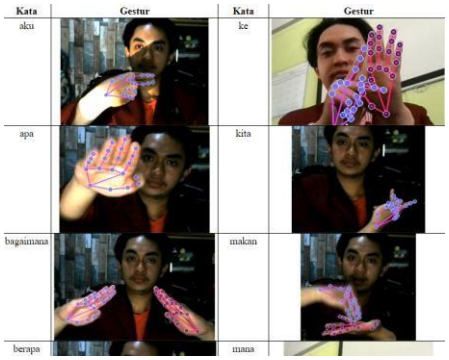
Tabel 1
Spesifikasi Pengembang

Laptop	
OS	Windows 11 64 Bit
Processor	AMD Ryzen 7 5800H 3.2GHz
GPU	RTX 3050Ti 4GB
RAM	16GB
Webcam	
Resolusi	720p
FPS	30 FPS
Software	
Tensorflow	2.7
Anaconda	22.9.0
Mediapipe	0.9.3.0

C. Pengumpulan Data

Pada bagian ini akan berfokus pada proses pembuatan model lebih tepatnya dalam bagian koleksi data. Data diambil menggunakan sebuah *platform* bernama Mediapipe, Mediapipe dapat merubah gambar menjadi koordinat yang nantinya akan dimasukan ke dalam pelatihan model ML.

Dataset diambil dari 26 gestur yang diambil oleh pengembang dari video untuk tiap gestur diambil sebanyak 30 kali masing-masing untuk memperoleh hasil halus 30 fps. Berikut potongan data yang diambil :



GAMBAR 4
Dataset Gestur SIBI

Gestur yang diambil dapat dilihat secara lengkap di *link* berikut : https://docs.google.com/spreadsheets/d/195Mw7cDbuhqSWX-Kd19AyjW-YUppChtT4FYFnVQaFow/edit?usp=sharing

Berikutnya data diolah oleh Mediapipe untuk memastikan tidak ada data yang kosong. Berikut kodenya :

```
# looping subdirektori dalam dataset
for subdir in os.listdir(parent_folder_path):
        subdir_path = os.path.join(parent_folder_path,
subdir)
   if os.path.isdir(subdir_path):
      # Iterasi 30 kali sesuai jumlah data (30fps)
      for i in range(30):
                    file_path = os.path.join(subdir_path,
f"aku_keypoints_{i}.npy")
         # Load data sesuai nama .npy file
         keypoints = np.load(file_path)
         # Cek apakah ada koordinat kosong
        if np.all(keypoints == 0):
      print(f"File {file_path} contains only zero")
      else:
          print(f"No files with all zero values found in
{subdir}.")
```

Dalam kode di atas akan mengiterasi semua data sesuai gesturnya untuk mengecek apakah kosong apabila ada yang kosong maka akan diisi secara random oleh koordinat yang lain dari salah satu 30 data koordinatnya.

D. Pelatihan Model

Pada proses pelatihan model algortima yang dipakai adalah LSTM karena perlu mendeteksi gerakan (*motion*) bahasa isyarat, pelatihan akan menggunakan 26 kata terlebih dahulu ['aku', 'apa', 'bagaimana', 'berapa', 'di', 'dia', 'F', 'halo', 'I', 'J', 'K', 'kamu', 'kapan', 'ke', 'kita', 'makan', 'mana', 'mereka', 'minum', 'nama', 'R', 'saya', 'siapa', 'Y', 'yang', 'Z']. Pembagian *train* test adalah 80:20:20. Berikut potongan kode untuk menginisiasi pelatihannya :

```
log_dir = os.path.join('Logs')
tb_callback = TensorBoard(log_dir=log_dir)

class StopTrainingCallback(Callback):
   def on_epoch_end(self, epoch, logs={}):
      if epoch > 200 and logs.get('loss') < 0.09 and
     (logs.get('categorical_accuracy') < 0.99 and
     logs.get('categorical_accuracy') > 0.91):
         print("Stopping training")
         self.model.stop_training = True
model.add(LSTM(256, return_sequences=True,
    activation='relu', input_shape=(30,126)))
model.add(Dense(128, activation='relu'))
```

Pada kode di atas akan diatur LSTM *unit* dan *layer*

density untuk melihat hasil akhir pelatihannya. Untuk pengujian akan dipaparkan secara rinci terhadap beberapa parameter termasuk *unit* dan*density*.

Setelah dilakukan pelatihan model akan dicek akurasi dan performanya dalam *realtime* sebagai berikut :

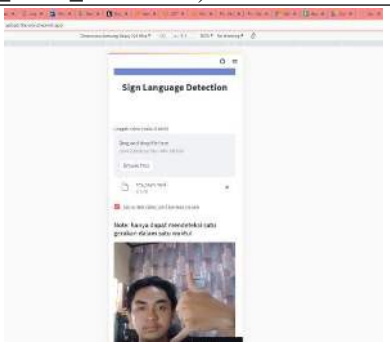
GAMBAR 5
Hasil Pelatihan Model

Untuk analisa dan pengujian rinci akan dicantumkan pada bagian IV.

E. Pembuatan Web Streamlit

Streamlit adalah sebuah platform yang dibuat pada akhir 2021 untuk mendeploy python untuk menjadi sebuah web, saat ini Streamlit masih dalam tahap pengembangan dan eksperimental. Website yang dibuat masih terpaut pada template dan masih harus dideploy pada platform resminya di https://streamlit.io/. Hasilnya dapat diakses oleh siapapun dan ini dapat dijadikan platform produksi, sehingga nanti aplikasi mobile bisa melakukan webview dari fungsi Kotlinnya untuk menampilkan webnya dalam aplikasi Android. Berikut kodenya untuk membuat Streamlit Web Apps :

```
    """
    <style>
    .bar {background-color: rgb(114, 134, 211);
       height: 35px}
    .sentence {font-size: 21px}
    .subheader {font-size: 14px;}
    </style>
    """,
    unsafe_allow_html=True
st.markdown('<div class="bar"></div>',
unsafe_allow_html=True)
st.markdown('<h1 style="text-align: center;">Sign
Language Detection</h1>',
unsafe_allow_html=True)
uploaded_file = st.file_uploader("Unggah video
(maks. 5 detik)", type=["mp4"])
flip_the_video = st.checkbox("cek untuk video
dari kamera depan")
st.markdown('<h2 style="font-size: 21px;">Note:
hanya efektif mendeteksi satu gerakan dalam satu
waktu!</h2>', unsafe_allow_html=True)
st.markdown('<h2 style="font-size: 27px;">Hasil
Deteksi:</h2>', unsafe_allow_html=True)
st.write('<div class="sentence">' + '
'.join(video_transformer.sentence) + '</div>',
unsafe_allow_html=True)
```



GAMBAR 6
Hasil Pembuatan Streamlit Web

Web yang dikembangkan di Streamlit memiliki kelemahan dalam segi proses asinkronus. Tidak seperti React.js yang dapat menyimpan dan menjalankan fungsi asinkronous secara bebas dan tidak terikat. Pada Streamlit masih menjalankan kode secara berurutan terutama apabila ada proses tiap fungsi. Dalam kode di atas ada beberapa fungsi yang asinkronus artinya ada yang dipanggil setelah dilewati terutama dalam proses deteksi dan menampilkan deteksi kata. Apabila dibuat *realtime* kamera stream bisa, akan tetapi katanya tidak muncul di layar karena tidak diproses secara langsung, untuk penampilan katanya.

IV. HASIL DAN PEMBAHASAN

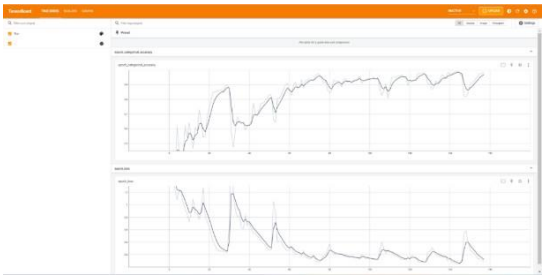
Pada bagian ini akan dilakukan uji dan analisa terhadap temuan yang dialami pada pembuatan system oleh pengembang, Adapun batasannnya adalah parameter uji yang akan dilakukan untuk melihat pengaruhnya terhadap pengembangan system.

A. *Epoch* Pelatihan

Pada bagian *epoch* pelatihan akan berfokus pada hasil temuan yang terjadi ketikan model dilatih pada *epoch* 50, 100, dan 150. Berikut grafiknya :

GAMBAR 7
Pelatihan Epoch 50

GAMBAR 8
Pelatihan Epoch 100



GAMBAR 9
Pelatihan Epoch 150

Hasil model yang dipilih adalah yang menggunakan 100 *epoch*. Berikut hasil dari akurasipelatihannya :

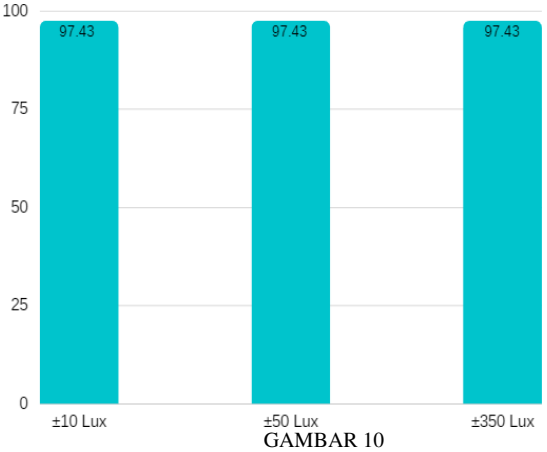
TABEL 2
Akurasi Pelatihan 100 *Epoch*

Parameter	Nilai
Accuracy	95.56%
Precision	99.04%
Recall	98.89%
F1 Score	98.87%

Dari hasil uji *epoch* di atas dapat dilihat bahwa performanya tidak begitu jauh berbeda, tentnunya hal ini tidak menjamin performanya di kenyataan. Berikutnya akan dilakukan pengujian lebih lanjut untuk mengetahui performanya

B. Cahaya

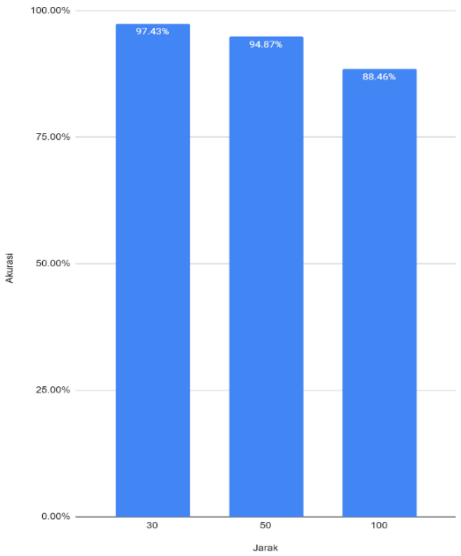
Cahaya yang diuji adalah dari jarak ±10, ±50 dan ±350 lux sesuai keadaan ruangan pengembang. Dalam tes ini sudah dilakukan tes secara waktu nyata untuk melihat performa aslimodel. Berikut hasilnya :



GAMBAR 10
Hasil Pengujian Cahaya

C. Jarak

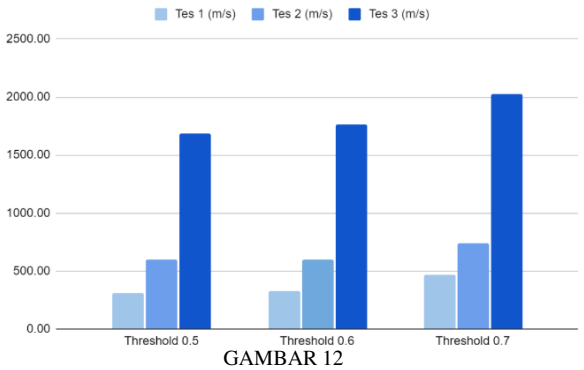
Jarak yang diuji adalah dari jarak 30, 50 dan 100cm dari perangkat tes pengembang. Dalam tes ini sudah dilakukan tes secara waktu nyata untuk melihat performa asli model. Berikut gambaran hasil ujinya :



GAMBAR 11
Hasil Pengujian Jarak

D. Uji Waktu Deteksi

Pada pengujian ini waktu deteksi diatur melalui *threshold* deteksi dari Mediapipe yang diatur sebagai berikut 0.5, 0.6, dan 0.7, karena Mediapipe lah yang berperan mengubah video menjadi masukan model untuk deteksi. Berikut hasilnya :



GAMBAR 12
Hasil Uji Waktu Deteksi

E. Kemampuan Streamlit

Pengujian dilakukan dengan mengunggah video pada web yang telah dibuat dari Streamlit untuk melihat seberapa banyak kata yang dapat dideteksidari satu video unggahan pengguna. Adapula batasan deteksi yang disematkan pada kode untuk melihat pengaruhnya terhadap gestur actual yang ada di video, parameter yang akan digunakan adalah gestur kata dalam satu video, *output* kata, dan kata yang benar dari *output* kata tersebut. Analisa yang digunakan adalah dengan tabel kebenaran, halini dirasa paling cocok untuk merepresentasikan pengujian ini. Berikut hasilnya:

TABEL 3
Hasil Uji Web Streamlit

Batasan Kata	Gestur Aktual	Output Kata	Kata Benar	Analisa
1	1	1	1	*True Positive*
2	2	2	2	*True Positive*
3	3	3	3	*True Positive*
4	4	4	3	*False Positive*
5	5	5	4	*False Positive*
6	6	6	3	*False Positive*
10	10	10	4	*False Positive*
tidak dibatasi	10	13	4	*False Negative* *False Positive*

Hasil dari uji analisa ini menunjukan adanya True Positive (TP) atau hasil deteksi benar dari gestur yang sebenarnya dan False Positive (FP) yakni hasil deteksi model secara tidak tepat mengidentifikasi gestur yang sebenarnya tidak ada. Ada pula di baris akhir model gagal mendeteksi gestur 10 yang sebenarnya ada karena output nya ada 13, maka ini adalah False Negative(FN).

V. KESIMPULAN

Berdasarkan hasil analisa dan grafik yang telah disajikan, dapat disimpulkan bahwa jarak tidak memiliki pengaruh yang signifikan terhadap hasil deteksi gestur. Meskipun ada beberapa kata yang tidakterdeteksi dalam beberapa tes, namun grafik menunjukkan bahwa cahaya tidak mempengaruhi hasil deteksi secara nyata. Perbedaan dalam deteksi gestur terlihat lebih mencolok ketika pencahayaan sangat terang dan ada banyak noise di kamera. Durasi video juga berpengaruh terhadap hasil deteksi, dan hal ini mungkin terkait dengan penggunaan platform Streamlit yang masih dalam tahap eksperimental dan prosesnya sinkronus. Meski demikian, hasil deteksi tetap efektif dan aman untuk mendeteksi satu gestur per video, seperti dalam video pendek yang hanya menampilkan satu gerakan SIBI saja. Adanya oversight untuk lebih dari 1-3 gestur dalam satu video tidak terlalu mengganggu secara signifikan dalam deteksi.

REFERENSI

[1] S. Mhatre, S. Joshi, dan H.B. Kulkarni, "Sign Language Detection using LSTM," Current Development in Engineering and Technology (CCET), IEEE, hal. 13-17, 2022.

[2] Google Developer, "Hand Landmarks Detection Guide," Google Mediapipe Documentation, [Online]. Tersedia: https://developers.google.com/mediapipe/solutions/vision/hand_landmarker. [Diakses: Apr 11, 2023].

[3] Engineering Team at SignAll, "MediaPipe is Now Available for Developers," Google Developers Blog, https://developers.googleblog.com/2021/04/signall-sdk-sign-language-interface-using-mediapipe-now-available.html, [Diakses 30-Juli-2023].

[4] S. Mekruksavanich, A. Jitpattanakul, dan G. Pau, "LSTM Networks Using Smartphone Data for Sensor-Based Human Activity Recognition in Smart Homes," Publikasi 2021 Feb 26, hal. 5-12.

[5] S. Oluyede Jr., "STREAMLIT(Part-1): BUILDING A SIMPLE STREAMLIT APPLICATION FOR COMPUTER VISION," https://juniorboyboy2.medium.com/streamlit-part-1-building-a-simple-streamlit-application-for-computer-vision-49f820a0938d. [Diakses: 30-Juli-2023].

[6] N. Landro, "Live Webcam with Streamlit," https://medium.com/mlearning-ai/live-webcam-with-streamlit-f32bf68945a4. [Diakses: 30-Juli-2023].

[7] Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network," Physica D: Nonlinear Phenomena, vol. 404, hal. 37, 2021