INNOVATIVE: Journal Of Social Science Research
Volume 4 Nomor 5 Tahun 2024 Page 3608-3628
E-ISSN 2807-4238 and P-ISSN 2807-4246
Website: https://j-innovative.org/index.php/Innovative

Penerapan Mediapipe dalam Pengenalan Bisindo Berbasis *Deep Learning* dan *Computer Vision*

(Studi Kasus: SLB-C B Yayasan Pendidikan Luar Biasa (YPLB) Blitar)

Didik Kholidil Ahwani[1✉], Saiful Nur Budiman[2], Mohammad Faried Rahmat[3]
Universitas Islam Balitar
Email: didikkholidil@gmail.com[1✉]

Abstrak

Di Indonesia, komunikasi antara komunitas Tuli dengan masyarakat umum sering kali terhambat oleh perbedaan bahasa, khususnya Bahasa Isyarat Indonesia (BISINDO). Penelitian ini memanfaatkan teknologi kecerdasan buatan, khususnya Convolutional Neural Networks (CNN) dalam platform MediaPipe, untuk meningkatkan pengenalan huruf-huruf dalam BISINDO melalui analisis gerakan tangan. Pendekatan ini bertujuan untuk mengurangi jurang komunikasi dengan memungkinkan interpretasi yang lebih baik terhadap gerakan kompleks dalam BISINDO. Metode penelitian ini melibatkan pengembangan dan evaluasi model CNN menggunakan MediaPipe untuk mengenali gerakan isyarat tangan. Pengukuran kinerja model dilakukan menggunakan Confusion matrix, dengan hasil akurasi mencapai 94% selama pelatihan dan 78,46%pada pengujian real-time. Hasil ini menunjukkan bahwa model berhasil mengklasifikasikan gerakan tangan dengan baik dalam kondisi ideal maupun di lingkungan dunia nyata. Penelitian ini memberikan kontribusi dalam mengembangkan solusi teknologi untuk mendukung komunikasi inklusif bagi komunitas Tuli, dengan potensi untuk diterapkan dalam aplikasi pengenalan bahasa isyarat dan teknologi asistensi lainnya.

Kata Kunci: *Bahasa Isyarat Indonesia (BISINDO), MediaPipe, Convolutional Neural Networks (CNN), Pengenalan Gerakan Tangan, Kecerdasan Buatan*

Abstract

In Indonesia, communication between the Deaf community and the general population is often hindered by language differences, particularly with Indonesian Sign Language (BISINDO). This research leverages artificial intelligence technology, specifically Convolutional Neural Networks (CNN) within the MediaPipe platform, to enhance the recognition of BISINDO letters through hand gesture analysis. The approach aims to bridge communication gaps by enabling better interpretation of complex gestures in BISINDO. The research methodology involves the development and evaluation of a CNN model using MediaPipe to recognize hand gestures. Model performance measurement is conducted using a Confusion matrix, achieving accuracies of 94% during Training and 78,46%in real-time testing. These results indicate successful classification of hand gestures in both ideal conditions and real-world environments. This study contributes to the development of technological solutions to support inclusive communication for the Deaf community, with potential applications in sign language recognition and other assistive technologies.

Keywords: *Indonesian Sign Language (BISINDO), MediaPipe, Convolutional Neural Networks (CNN), Hand Gesture Recognition, Artificial Intelligence*

PENDAHULUAN

Di jantung pulau-pulau Indonesia, komunitas Tuli berkomunikasi dengan gerakan dan ekspresi, menggunakan Bahasa Isyarat Indonesia (BISINDO) sebagai bahasa mereka. Namun, dalam komunikasi yang dido*min*asi oleh kata-kata dan suara, BISINDO sering kali menjadi sebuah bahasa yang terpisah dan tidak dipahami oleh masyarakat umum. Ini menciptakan sebuah jurang komunikasi yang signifikan antara komunitas Tuli dan pendengar.

Di era teknologi yang pesat, solusi untuk tantangan ini mungkin terletak pada kecanggihan kecerdasan buatan (AI). Dalam penelitian ini, kami mengarahkan sorotan pada MediaPipe, sebuah platform yang dirancang oleh Google untuk memfasilitasi pengenalan gerakan yang kompleks (Halder, 2021; Indriani dkk., 2021). Inti dari inovasi ini adalah algoritma *Convolutional Neural Networks* (CNN).

MediaPipe, sebuah platform yang dikembangkan oleh Google, untuk memfasilitasi pengenalan gerakan yang kompleks. MediaPipe dirancang dengan tujuan untuk mendukung analisis visual dan pengolahan data dari input video secara real-time. Salah satu keunggulan utama MediaPipe adalah kemampuannya dalam mengenali dan melacak objek serta gerakan dalam video dengan akurasi tinggi.

CNN, dikenal karena keefektifannya dalam mengolah data visual, menjadi pilihan untuk menginterpretasikan isyarat tangan yang kompleks (Budiman dkk., 2023b). Melalui

lapisan-lapisan konvolusinya, CNN mampu menangkap detail dan pola dalam gerakan isyarat tangan, mengubah setiap gerakan menjadi informasi yang bisa diproses oleh komputer. Kekuatan utama dari CNN terletak pada kemampuannya untuk secara efektif mengenali pola visual dalam data gambar dan video.

Selain itu, penggunaan OpenCV (Open Source Computer Vision Library) memperkuat penelitian ini dengan menyediakan alat-alat pemrosesan gambar dan video yang efisien. OpenCV mendukung berbagai operasi, dari membaca dan menulis gambar hingga deteksi objek dan pengenalan pola (Budiman dkk., 2023a), serta dapat diintegrasikan dengan MediaPipe dan CNN untuk meningkatkan akurasi pengenalan gerakan isyarat tangan.

Untuk mengukur akurasi model dalam mengenali isyarat tangan menggunakan algoritma CNN dalam MediaPipe, dapat digunakan metrik seperti *Confusion matrix*. *Confusion matrix* adalah tabel yang menggambarkan performa model klasifikasi dengan membandingkan nilai prediksi dengan nilai sebenarnya pada sebuah set data uji yang telah dilabeli dengan benar. Dengan menggunakan *Confusion matrix*, dapat dievaluasi seberapa baik model CNN dalam mengklasifikasikan isyarat tangan dengan benar, termasuk dalam hal deteksi dan pengenalan gerakan.

Penelitian ini didasarkan pada penelitian sebelumnya yang dilakukan oleh Kazuhito Takahashi pada tahun 2020 yang memakai algoritma MLP (*Multi-Layer Perceptron*) dan LSTM (Takahashi, 2020). Alasan penulis memilih projek ini sebagai dasar penelitian karena banyak hal yang bisa ditingkatkan dari projek ini seperti meningkatkan akurasi dengan algoritma CNN.

Untuk mengevaluasi kinerja model dalam mengenali isyarat tangan menggunakan algoritma CNN dalam MediaPipe, dapat menggunakan metrik seperti *Confusion matrix*. *Confusion matrix* adalah tabel yang digunakan untuk menggambarkan kinerja model klasifikasi pada sebuah set data uji yang telah dilabeli dengan benar.

Dengan begitu, penelitian ini menawarkan solusi untuk menggunakan kecerdasan buatan, khususnya melalui teknologi CNN di platform MediaPipe, untuk membantu komunikasi antara komunitas Tuli dan non-Tuli. Di tengah banyaknya bahasa isyarat, Bahasa Isyarat Indonesia (BISINDO) sering kali tersisihkan dan kurang dipahami oleh masyarakat umum. Dengan CNN, kita bisa lebih baik mengerti dan menafsirkan gerakan isyarat tangan yang kompleks dalam BISINDO, yang diharapkan bisa memperkuat hubungan antara keduanya. Dengan menggunakan metrik seperti *Confusion matrix*, penelitian ini bisa memberikan gambaran yang lebih jelas tentang seberapa baik model ini

bekerja dalam mengenali isyarat tangan dengan benar. Dengan terus mengembangkan teknologi ini, kita bisa mengurangi kesenjangan komunikasi dan memperjuangkan inklusi bagi semua orang dalam masyarakat. Berdasarkan latar belakang diatas maka penulis berinisiatif mengangkat skripsi dengan judul penerapan mediapipe dalam pengenalan bisindo berbasis *deep learning* dan *computer vision.*

METODE PENELITIAN

Jenis Penelitian

Penelitian ini termasuk Penelitian kuantitatif, berfokus pada pengukuran variabel-variabel dan penggunaan statistik. Dalam pengembangan teknologi MediaPipe untuk pengenalan huruf A-Z dalam Bahasa Isyarat Indonesia (BISINDO), penelitian ini menggunakan pendekatan kuantitatif untuk mengukur tingkat keberhasilan dan akurasi pengenalan huruf, serta untuk menganalisis pola-pola gerakan dan ekspresi yang terkait dengan setiap huruf.

Penelitian ini dapat memberikan wawasan yang berharga tentang area-area yang perlu diperbaiki atau dioptimalkan. Analisis statistik terhadap pola-pola gerakan dan ekspresi yang terkait dengan setiap huruf juga dapat memberikan pemahaman yang lebih mendalam tentang karakteristik unik dari masing-masing isyarat tangan.

Pengumpulan Data

1. Teknik Pengumpulan Data
    a. Observasi

Gambar 1. Proses Pengambilan Data

Dalam penelitian ini, pendekatan observasi digunakan untuk mengumpulkan data terkait penggunaan Bahasa Isyarat Indonesia (BISINDO) di Sekolah Luar Biasa (SLB). Melalui penggunaan kamera, peneliti secara langsung merekam gerakan tangan guru SLB saat mereka menggunakan BISINDO. Observasi ini memberikan pemahaman yang mendalam tentang cara komunikasi guru SLB dalam menggunakan huruf-huruf A-Z BISINDO. Data yang terkumpul dari observasi ini akan menjadi dasar penting dalam pengembangan model pengenalan huruf BISINDO menggunakan teknologi seperti MediaPipe.

b. Studi literatur

Penelitian ini juga menggunakan bantuan dari studi literatur dari buku dan artikel tentang penggunaan Bahasa Isyarat Indonesia (BISINDO), mediapipe, dan *deep learning* CNN. Ini membantu memahami masalah dan solusi yang telah ditemukan oleh para ahli sebelumnya.

Temuan dari buku dan artikel ini ditulis dalam daftar rujukan di bagian belakang penelitian. Dengan informasi dari studi literatur dan hasil dari observasi, peneliti berharap dapat membuat saran yang berguna untuk pengembangan teknologi yang mendukung penggunaan BISINDO di SLB.

2. Jenis Data

Dalam rangka memahami penggunaan Bahasa Isyarat Indonesia (BISINDO), penelitian ini mengadopsi pendekatan pengumpulan data primer melalui partisipasi langsung guru SLB sebagai sumber informasi. Guru diminta untuk memberikan informasi dan wawasan tentang penggunaan huruf A-Z BISINDO.

Data primer yang terkumpul dari partisipasi para guru ini menjadi dasar yang kuat dalam merancang teknologi pendukung, seperti pengembangan model pengenalan huruf BISINDO menggunakan MediaPipe, untuk meningkatkan pemahaman dan efektivitas penggunaan bahasa isyarat secara umum.

3. Instrumen Pengumpulan Data

Instrumen penelitian adalah alat atau metode yang digunakan untuk mengumpulkan data dalam penelitian. Ini bisa berupa alat pengukuran, tes, atau teknik pengamatan yang dirancang untuk mencapai tujuan penelitian. Instrumen ini harus relevan, dapat diandalkan, dan valid untuk memastikan data yang dikumpulkan berguna bagi penelitian. Dengan demikian, pemilihan dan penggunaan

instrumen yang tepat sangat penting untuk mendapatkan data yang akurat dan bermakna dalam penelitian.

HASIL DAN PEMBAHASAN

Pengumpulan dataset

1. Pengambilan data

```
# Mendapatkan citra tangan dari kamera
ret, image = cap.read()
# Melakukan deteksi tangan menggunakan model MediaPipe
results = hands.process(image)
# Mendapatkan landmark tangan dari hasil deteksi
landmark_list = calc_landmark_list(debug_image, hand_landmarks)
# Menyimpan data landmark ke dalam file CSV
logging_csv(number, mode, landmark_list, pre_processed_point_history_list)
```

Program ini mengumpulkan data melalui penggunaan kamera yang terhubung ke perangkat. Data yang dikumpulkan adalah citra tangan yang dideteksi oleh model deteksi tangan dari MediaPipe. Citra tangan kemudian diproses untuk mengidentifikasi landmark (titik-titik penting) di tangan.

Data landmark yang diperoleh dari setiap citra tangan kemudian disimpan dalam file CSV sebagai bagian dari pengumpulan data.

2. Normalisasi

```
# Pre-processing landmark: mengubah koordinat menjadi relatif dan dinormalisasi
pre_processed_landmark_list = pre_process_landmark(landmark_list)
```

Setelah data landmark diperoleh dari citra tangan, langkah normalisasi diterapkan pada data tersebut. Normalisasi dilakukan untuk memastikan bahwa data memiliki skala 0-1 yang seragam dan dapat dibandingkan dengan data lainnya dengan mudah.

Proses normalisasi melibatkan dua langkah utama:

a. Pertama, data landmark dikonversi menjadi koordinat relatif terhadap titik acuan tertentu, yaitu titik tangan pertama yang terdeteksi.
b. Kedua, koordinat relatif tersebut dinormalisasi dengan membagi nilai-nilai koordinat dengan nilai maksimum yang ditemukan di antara semua nilai koordinat. Hal ini dilakukan agar data memiliki rentang nilai antara 0 dan 1.

3. Pelabelan data

```
# Melakukan klasifikasi tindakan tangan berdasarkan landmark

hand_sign_id = keypoint_classifier(pre_processed_landmark_list)

# Melakukan klasifikasi gerakan jari berdasarkan riwayat titik
```

a. Data landmark yang telah dinormalisasi akan digunakan untuk melakukan klasifikasi tindakan tangan. Sebelumnya, model klasifikasi telah dilatih dengan menggunakan data yang telah dilabeli sebelumnya.

b. Label-label tersebut diperoleh dari file CSV yang berisi label untuk klasifikasi titik kunci dan riwayat titik. Setelah data landmark dinormalisasi, model klasifikasi digunakan untuk memprediksi label tindakan tangan berdasarkan data landmark tersebut.

Setelah data landmark disimpan dalam file CSV, data ini mencakup koordinat MediaPipe tangan yang digunakan untuk proses pelabelan lebih lanjut.

Dataset

Class 0: 200 rows, 43 float columns	A
Class 1: 200 rows, 43 float columns	B
Class 2: 200 rows, 43 float columns	C
Class 3: 200 rows, 43 float columns	D
Class 4: 200 rows, 43 float columns	E
Class 5: 200 rows, 43 float columns	F
Class 6: 200 rows, 43 float columns	G
Class 7: 200 rows, 43 float columns	H
Class 8: 200 rows, 43 float columns	I
Class 9: 200 rows, 43 float columns	J
Class 10: 200 rows, 43 float columns	K
Class 11: 200 rows, 43 float columns	L
Class 12: 200 rows, 43 float columns	M
Class 13: 200 rows, 43 float columns	N
Class 14: 200 rows, 43 float columns	O
Class 15: 200 rows, 43 float columns	P
Class 16: 200 rows, 43 float columns	Q
Class 17: 200 rows, 43 float columns	R
Class 18: 200 rows, 43 float columns	S
Class 19: 200 rows, 43 float columns	T
Class 20: 200 rows, 43 float columns	U
Class 21: 200 rows, 43 float columns	V
Class 22: 200 rows, 43 float columns	W
Class 23: 200 rows, 43 float columns	X
Class 24: 200 rows, 43 float columns	Y
Class 25: 200 rows, 43 float columns	Z

Dataset ini disimpan dalam file 'keypoint.csv' dan terdiri dari 26 kelas yang mewakili huruf A-Z. Setiap kelas memiliki 200 baris data yang mewakili koordinat dari titik-titik kunci pada gambar. Dalam total, setiap kelas memiliki 43 kolom yang berisi nilai float yang masing-masing mewakili koordinat X dan Y dari titik-titik tersebut.

Misalnya, kelas 0 mewakili huruf A, dan setiap baris dalam kelas ini berisi koordinat dari titik-titik kunci yang membentuk huruf A pada gambar. Hal yang sama berlaku untuk kelas lainnya, dengan kelas 1 mewakili huruf B, kelas 2 mewakili huruf C, dan seterusnya hingga kelas 25 yang mewakili huruf Z.

Pelatihan model

1. Model CNN

Dalam penelitian ini, digunakan model jaringan saraf konvolusional (CNN) untuk mengklasifikasikan gerakan tangan berdasarkan data koordinat titik kunci (landmarks) yang dihasilkan oleh Mediapipe. Model ini menerima input berupa data dengan ukuran 21x2 piksel, di mana 21 adalah jumlah titik kunci pada tangan dan 2 adalah koordinat x dan y dari setiap titik kunci.

Struktur Model

```
# Model building
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 2), activation='relu',
padding='same', input_shape=(21, 2, 1)),
    tf.keras.layers.MaxPooling2D((2, 1)),
    tf.keras.layers.Dropout(0.25),
    tf.keras.layers.Conv2D(64, (3, 2), activation='relu',
padding='same'),
    tf.keras.layers.MaxPooling2D((2, 1)),
    tf.keras.layers.Dropout(0.25),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(NUM_CLASSES,
activation='softmax')
])
```

a. Lapisan Konvolusional Pertama

- Conv2D: Lapisan ini memiliki 32 filter dengan ukuran 3x2. Fungsi aktivasi yang digunakan adalah ReLU dipilih karena sederhana dalam komputasi dan mencegah masalah vanishing gradient. Fungsi aktivasi ini memperkenalkan non-linearitas ke dalam model, memungkinkan pembelajaran pola yang lebih kompleks dalam data. Padding 'same' digunakan agar ukuran output sama

dengan input. Input yang diterima berukuran (21, 2, 1).

- MaxPooling2D: Lapisan pooling dengan ukuran pool 2x1 digunakan untuk mengurangi ukuran fitur.
- Dropout: Dropout sebesar 25% digunakan untuk mencegah *overfitting* dengan menonaktifkan beberapa neuron secara acak selama pelatihan.

b. Lapisan Konvolusional Kedua

- Conv2D: Lapisan ini memiliki 64 filter dengan ukuran 3x2. Fungsi aktivasi yang digunakan adalah ReLU dan padding 'same'.
- MaxPooling2D: Lapisan pooling dengan ukuran pool 2x1 digunakan lagi untuk mengurangi ukuran fitur lebih lanjut.
- Dropout: Dropout sebesar 25% digunakan kembali untuk mencegah *overfitting*.

c. Lapisan Flatten

- Flatten: Lapisan ini meratakan output dari lapisan sebelumnya menjadi satu vektor panjang. Ini diperlukan untuk menghubungkan lapisan konvolusional dengan lapisan berikutnya.

d. Lapisan Dense

- Dense: Lapisan ini memiliki 128 neuron dengan fungsi aktivasi ReLU. Lapisan ini menggabungkan semua fitur yang telah diekstraksi.
- Dropout: Dropout sebesar 50% digunakan untuk mencegah *overfitting*

e. Lapisan Output

- Dense: Lapisan ini memiliki 26 neuron sesuai dengan jumlah kelas yang ingin diprediksi (NUM_CLASSES = 26). Fungsi aktivasi yang digunakan adalah Softmax menghasilkan distribusi probabilitas dari output, yang sangat berguna dalam klasifikasi multikelas karena memungkinkan model untuk mengeluarkan prediksi kelas dalam bentuk probabilitas.

2. Pelatihan Model

```
model.fit(
    X_train,
    y_train,
    epochs=100,
    batch_size=128,
    validation_data=(X_test, y_test),

callbacks=[tf.keras.callbacks.ModelCheckpoint(model
_save_path_keras, verbose=1,
save_weights_only=False),

tf.keras.callbacks.EarlyStopping(patience=20,
verbose=1)]
)
```

Proses pelatihan dimulai dengan menggunakan fungsi model.fit(). Data pelatihan yang terdiri dari X_*Train* dan y_*Train* diberikan ke model untuk dipelajari selama 100 *epoch*, dimana setiap *epoch* berarti model akan melalui seluruh dataset sebanyak 100 kali. Data pelatihan ini dibagi menjadi batch kecil dengan ukuran 128 untuk mempercepat proses pelatihan.

3. Hasil pelatihan (*Confusion matrix*)

Fungsi confusion matrix digunakan untuk mengevaluasi seberapa baik model bekerja dalam mengenali huruf-huruf dari gambar.

Classification Report

	precision	recall	f1-score	support
0	0.98	0.78	0.87	63
1	0.74	0.98	0.84	57
2	0.88	1.00	0.94	51
3	0.96	0.47	0.64	57
4	1.00	1.00	1.00	64
5	1.00	1.00	1.00	46
6	1.00	1.00	1.00	54
7	0.98	1.00	0.99	46
8	1.00	1.00	1.00	43
9	1.00	1.00	1.00	58
10	0.93	1.00	0.96	52
11	0.98	1.00	0.99	50
12	1.00	0.44	0.61	52
13	0.50	1.00	0.67	29
14	1.00	1.00	1.00	52
15	0.90	1.00	0.95	52
16	0.98	1.00	0.99	47
17	0.98	1.00	0.99	51
18	0.90	0.90	0.90	52
19	1.00	1.00	1.00	48
20	1.00	1.00	1.00	52
21	1.00	1.00	1.00	42
22	0.94	1.00	0.97	50
23	0.98	0.98	0.98	44
24	1.00	0.98	0.99	47
25	1.00	1.00	1.00	41
accuracy			0.94	1300
macro avg	0.95	0.94	0.93	1300
weighted avg	0.95	0.94	0.93	1300

Kesimpulan: Model pengenalan isyarat tangan berbasis CNN yang digunakan dalam penelitian ini menunjukkan kinerja yang sangat baik dengan akurasi keseluruhan 0.94. Meskipun ada beberapa kelas yang memerlukan perbaikan lebih lanjut (seperti kelas 0 (huruf A), 1 (huruf B), 3 (huruf D), dan 12 (huruf M)), sebagian besar kelas memiliki *precision, recall,* dan *f1-score* yang tinggi, menunjukkan kemampuan model untuk mengidentifikasi isyarat tangan dengan tepat dan andal.

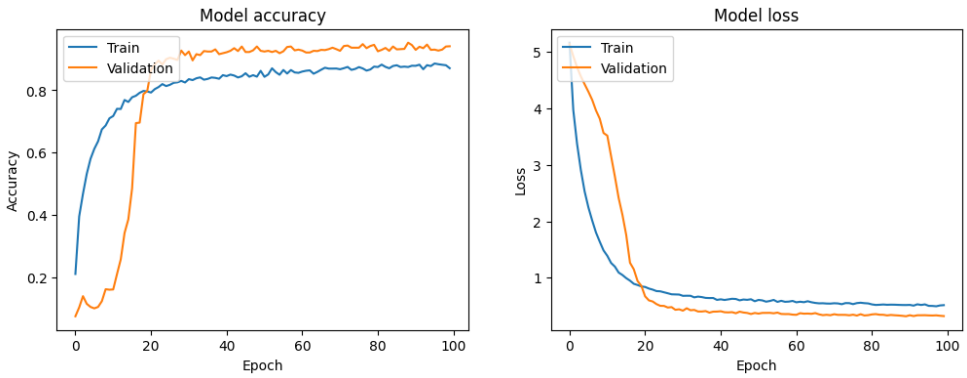
4. Performa model

```
# Plot training & validation accuracy values
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper
left')

# Plot training & validation loss values
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper
left')
plt.show()
```

Kode ini membuat dua grafik untuk memvisualisasikan performa model selama pelatihan:

a. Plot Akurasi:
   - *Train*ing: Grafik akurasi data pelatihan.
   - *Validation*: Grafik akurasi data validasi.
   - Menunjukkan seberapa baik model memprediksi data selama pelatihan dan validasi.

b. Plot Loss:
   - *Train*ing: Grafik loss data pelatihan.
   - *Validation*: Grafik loss data validasi.
   - Menunjukkan seberapa besar kesalahan model selama pelatihan dan validasi.



Gambar 2. visualisasi performa model

Performa model dalam konteks machine learning dan data science merujuk pada seberapa baik model dapat membuat prediksi atau keputusan berdasarkan data yang diberikan. Evaluasi performa model penting untuk memastikan bahwa model

tidak hanya bekerja dengan baik pada data yang digunakan untuk melatihnya (*Train*ing data) tetapi juga dapat menggeneralisasi dengan baik pada data yang belum pernah dilihat sebelumnya (*Validation*/test data).

5. Simpan model

```
# Save as a model dedicated to inference
model.save(model_save_path,
include_optimizer=False)
```

Baris ini menyimpan model Keras yang sudah dilatih ke dalam file dengan format HDF5. Parameter include_optimizer=False berarti informasi tentang optimizer (seperti state dan hyperparameters) tidak akan disimpan, yang membuat file lebih kecil dan lebih cocok untuk inferensi.

Dengan menyimpan model tanpa informasi optimizer, file hasil penyimpanan akan lebih ringkas dan fokus pada bobot model serta arsitektur jaringan. Ini sangat bermanfaat ketika model hanya akan digunakan untuk prediksi (inferensi) di masa mendatang, karena informasi optimizer hanya diperlukan saat melanjutkan pelatihan model.

Pengujian secara real-time

1. Inisialisasi dan Membaca Model:

```
from model import KeyPointClassifier
import mediapipe as mp

# Inisialisasi classifier
keypoint_classifier = KeyPointClassifier()

# Baca label
with
open('model/keypoint_classifier/keypoint_classifier_l
abel.csv', encoding='utf-8-sig') as f:
    keypoint_classifier_labels = csv.reader(f)
    keypoint_classifier_labels = [row[0] for row in
keypoint_classifier_labels]
```

2. Mengimpor Modul:

a. from model import KeyPointClassifier: Mengimpor kelas KeyPointClassifier dari modul model.

b. import mediapipe as mp: Mengimpor pustaka MediaPipe yang digunakan untuk pengolahan citra dan deteksi pose tangan.

3. Inisialisasi Classifier:

keypoint_classifier = KeyPointClassifier(): Membuat instance dari KeyPointClassifier, yang akan digunakan untuk klasifikasi titik kunci tangan.

4. Membaca Label:

a. Membuka file CSV yang berisi label untuk classifier dengan encoding utf-8-sig untuk memastikan karakter khusus dibaca dengan benar.

b. Membaca isi file CSV menggunakan csv.reader dan menyimpan label-label tersebut ke dalam list keypoint_classifier_labels.

5. Proses Deteksi dan Klasifikasi:

```
# Inisialisasi MediaPipe
mp_hands = mp.solutions.hands
hands = mp_hands.Hands(
    static_image_mode=False,
    max_num_hands=2,
    min_detection_confidence=0.5,
    min_tracking_confidence=0.5
)

# Mulai capture kamera
cap = cv.VideoCapture(0)

while cap.isOpened():
    ret, image = cap.read()
    if not ret:
        break

    image = cv.flip(image, 1)  # Mirror display
    debug_image = copy.deepcopy(image)

    # Konversi ke RGB untuk MediaPipe
    image = cv.cvtColor(image, cv.COLOR_BGR2RGB)
    image.flags.writeable = False
    results = hands.process(image)
    image.flags.writeable = True
```

a. Gambar dari kamera dikonversi ke RGB karena MediaPipe menggunakan format ini.

b. Gambar diolah untuk mendeteksi titik-titik kunci tangan.

c. Landmark tangan diklasifikasikan menggunakan model keypoint_classifier.

d. Hasil klasifikasi ditampilkan pada layar dengan teks dan garis-garis tangan.

6. Kontrol Kamera dan Tampilan

```
    if results.multi_hand_landmarks:
        for hand_landmarks in
results.multi_hand_landmarks:
            # Lakukan klasifikasi berdasarkan
landmark
            landmarks = hand_landmarks.landmark
            keypoints = [(lm.x, lm.y, lm.z) for lm in
landmarks]
            prediction =
keypoint_classifier(keypoints)
```

```
    if results.multi_hand_landmarks:
        for hand_landmarks in
results.multi_hand_landmarks:
            # Lakukan klasifikasi berdasarkan
landmark
            landmarks = hand_landmarks.landmark
            keypoints = [(lm.x, lm.y, lm.z) for lm in
landmarks]
            prediction =
keypoint_classifier(keypoints)
            predicted_label =
keypoint_classifier_labels[prediction]

            # Tampilkan hasil klasifikasi
            cv.putText(debug_image, predicted_label,
(10, 30), cv.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2,
cv.LINE_AA)

mp.solutions.drawing_utils.draw_landmarks(
                debug_image, hand_landmarks,
mp_hands.HAND_CONNECTIONS
            )
```

Berikut adalah penjelasan yang lebih sederhana dan ringkas:

a. Pengecekan Keberadaan Landmark Tangan:

   Memeriksa apakah ada landmark tangan yang terdeteksi dalam hasil (`results.multi_hand_landmarks`).

b. Iterasi Melalui Landmark Tangan yang Terdeteksi:

   Jika ada landmark tangan yang terdeteksi, iterasi dilakukan untuk setiap set landmark tangan yang ditemukan.

c. Mengambil dan Memproses Landmark:

   Mengambil daftar landmark dari setiap tangan yang terdeteksi dan mengubahnya menjadi daftar koordinat (x, y, z) untuk setiap titik kunci.

d. Klasifikasi Berdasarkan Landmark:

   Menggunakan classifier untuk memprediksi isyarat tangan berdasarkan titik kunci yang diambil.

   Mengambil label yang sesuai dengan prediksi dari daftar label.

e. Menampilkan Hasil Klasifikasi:

   Menampilkan teks hasil klasifikasi pada gambar.

   Menggambar landmark tangan yang terdeteksi beserta koneksinya pada gambar.

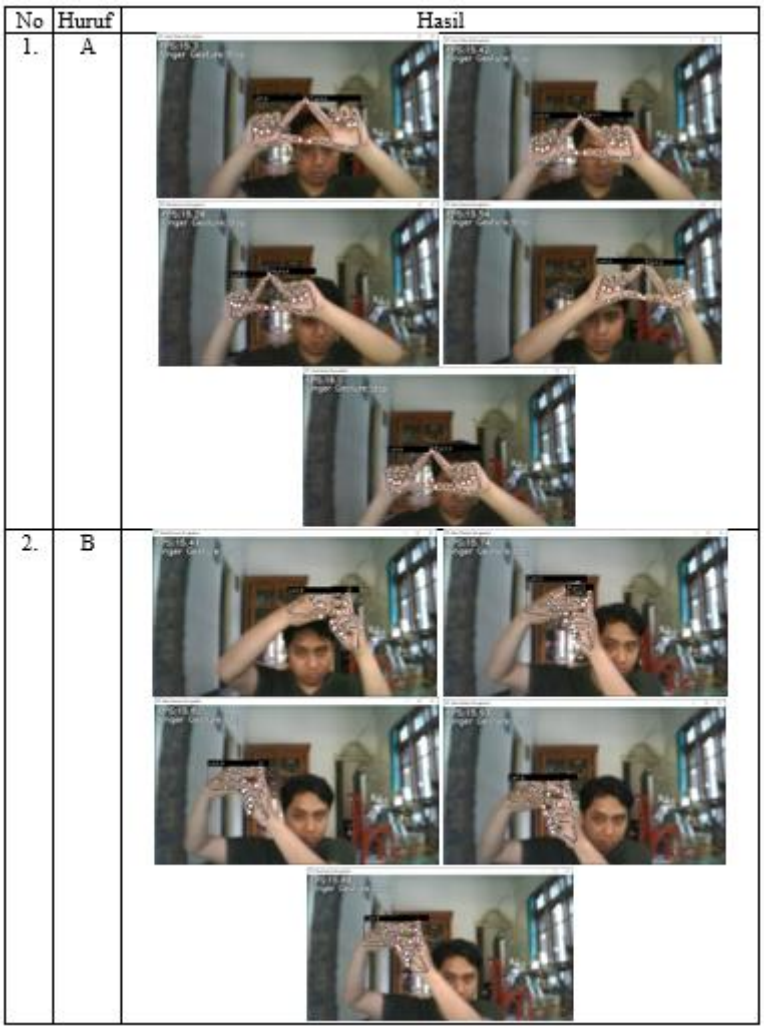
f. Loop untuk Menangkap Gambar secara Real-Time:

   Loop ini berjalan terus menerus untuk menangkap gambar dari kamera secara real-time dan memproses setiap frame.

g. Pembaruan Tampilan pada Setiap Frame:

   Tampilan diperbarui dengan hasil klasifikasi pada setiap frame, menunjukkan label

prediksi dan visualisasi landmark tangan pada gambar yang dihasilkan dari kamera. Dengan langkah-langkah ini, kode ini memungkinkan deteksi dan klasifikasi isyarat tangan secara real-time, dengan hasil klasifikasi yang diperbarui dan ditampilkan pada setiap frame yang ditangkap dari kamera.



Gambar 3. Hasil pengujian

Diatas adalah salah satu sampel pengujian,

1. Langkah pertama, jumlahkan semua persentase:
   (A)100 + (B)0 + (C)100 + (D)100 + (E)100 + (F)100 + (G)100 + (H)20 + (I)100 + (J)20 + (K)60 + (L)100 + (M)100 + (N)60 + (O)100 + (P)40 + (Q)60 + (R)100 + (S)40 + (T)100 + (U)100 + (V)100 + (W)100 + (X)40 + (Y)100 + (Z)100 = 2040
2. Langkah kedua, bagi hasil penjumlahan dengan jumlah data (26):
   $$2040/26 = 78{,}4615$$
3. Langkah ketiga, kalikan dengan 100:
   $$78{,}4615 \times 100 = 7846{,}15$$

Jadi, rata-rata persentase dari hasil penelitian adalah sekitar 78,46%. walaupun persetase termasuk tinggi tetapi masih ada beberapa huruf yang memiliki akurasi rendah seperti huruf B, H, K, S, X. akurase rendah bisa dikarenakan beberapa faktor: kualitas kamera, pencahayaan, dan *background*.

Ada 2 poin yang bisa diambil dari hasil pelatihan (*Confusion matrix*) dan hasil pengujian secara *real-time* :

1. Akurasi dari *Confusion matrix*: 94% - Ini adalah akurasi yang diperoleh selama fase pelatihan model. *Confusion matrix* memberikan informasi rinci tentang seberapa baik model mengklasifikasikan data pelatihan ke dalam kategori yang benar. Akurasi 94% menunjukkan bahwa model memiliki kinerja yang sangat baik saat diuji pada data pelatihan.
2. Akurasi pengujian real-time: 78,46% - Ini menunjukkan bahwa model memiliki akurasi 78,46% ketika diuji dalam kondisi real-time. Pengujian real-time melibatkan tantangan tambahan seperti kondisi pencahayaan, sudut pandang, dan kualitas kamera.

   Meskipun terdapat sedikit penurunan akurasi dari 94% menjadi 78,46% saat beralih dari pengujian standar ke pengujian rea*l-ti*me, hasil ini tetap menunjukkan bahwa model berfungsi dengan baik di lingkungan dunia nyata.

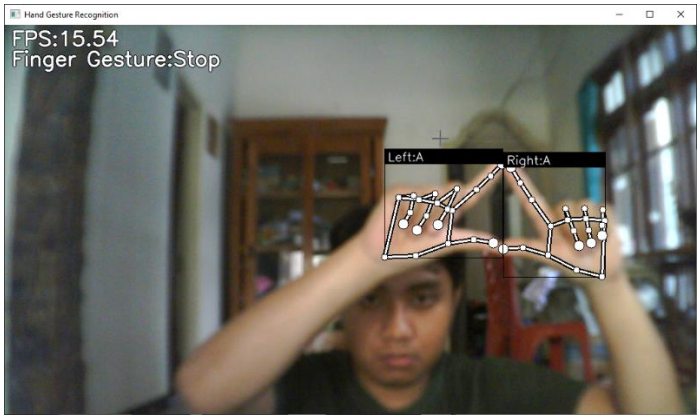
7. Evaluasi algoritma:

```
        if hand_sign_id_to_display is not None:
            if hand_sign_id_to_display in
[keypoint_classifier_labels.index('C'),
keypoint_classifier_labels.index('D')]:
                if left_hand_sign_id is not None and
right_hand_sign_id is not None:
                    hand_sign_id_to_display =
keypoint_classifier_labels.index('D')
                else:
                    hand_sign_id_to_display =
keypoint_classifier_labels.index('C')
            if hand_sign_id_to_display ==
keypoint_classifier_labels.index('S'):
                if not is_letter_s(landmark_list):
                    hand_sign_id_to_display = None  # Jika
tidak cocok dengan bentuk S, set ke None


        if hand_sign_id_to_display is not None:
            if hand_sign_id_to_display ==
keypoint_classifier_labels.index('L'):
                # Cek apakah hanya satu tangan yang terdeteksi

```

Kode ini digunakan untuk memastikan bahwa huruf yang ditampilkan adalah hasil

dari deteksi tangan yang benar dan sesuai dengan aturan yang diinginkan, seperti memastikan huruf C dan D bisa dibedakan berdasarkan jumlah tangan yang terdeteksi, kemudian S memiliki bentuk yang benar dan huruf L hanya terdeteksi jika hanya satu tangan yang terdeteksi.

Alasan digunakannya algoritma khusus untuk beberapa huruf dikarenakan ada beberapa huruf yang mirip dengan satu dan lainnya, dan juga permasalahan dimana ada bebrapa huruf yang saat pengujian tidak terdeteksi selain dikarenakan kualitas kamera kualitas pencahayaan dan background dari pengujian juga penting, untuk mengatasi masalah tersebut diperlukan kamera yang berkualitas bagus tetapi kamera bagus juga tetap memiliki kekurangan yaitu fps yang rendah.



Gambar 4. Performa fps kamera

Untuk solusi fps rendah bisa diatasi dengan penggunaan kamera yang khusus digunakan untuk mendeteksi tangan, dikarenakan kamera ini khusus untuk harga dari kamera ini mahal.

SIMPULAN

Berdasarkan penelitian yang dilakukan, maka peneliti mengambil beberapa kesimpulan antara lain:

Penerapan MediaPipe dalam pengenalan BISINDO dimulai dengan pengumpulan data isyarat BISINDO dalam bentuk koordinat, yang kemudian dinormalisasi dan dilabeli untuk memastikan konsistensi dan keakuratan dalam pelatihan model. Dataset yang telah diproses digunakan untuk melatih model Convolutional Neural Network (CNN) dengan arsitektur yang sesuai dan dilatih menggunakan data yang telah dilabeli, di mana hasil pelatihan disimpan dalam format file hdf5. Model ini kemudian diuji secara real-time dengan menggunakan MediaPipe untuk menangkap dan memproses input

video langsung, memungkinkan evaluasi performa model dalam kondisi dunia nyata.

Hasil pelatihan model menggunakan confusion matrix menunjukkan akurasi sebesar 94%, yang menunjukkan bahwa model memiliki kinerja yang sangat baik dalam mengenali isyarat BISINDO dari dataset pelatihan. Namun, pengujian secara real-time menunjukkan bahwa model mencapai akurasi sebesar 78,46%. Meskipun terdapat penurunan akurasi dibandingkan dengan fase pelatihan, model tetap berfungsi dengan baik di lingkungan dunia nyata. Hasil ini menunjukkan potensi besar untuk aplikasi praktis dalam pengenalan bahasa isyarat, dengan model yang mampu beradaptasi dengan berbagai kondisi seperti pencahayaan dan kualitas kamera.

Untuk mengatasi masalah akurasi rendah pada huruf-huruf tertentu seperti B, H, K, S, dan X, disarankan untuk menggunakan kamera khusus untuk mendeteksi tangan dan memastikan pencahayaan yang baik selama proses pengumpulan data.

DAFTAR PUSTAKA

Ardiansyah, A. R., Nur'azizan, A. H., & Fernandis, R. (2024). Implementasi Deteksi Bahasa Isyarat Tangan Menggunakan OpenCV dan MediaPipe. *STAINS (SEMINAR NASIONAL TEKNOLOGI & SAINS)*, *3*(1), Article 1.

Arifah, I. I., Fajri, F. N., & Pratamasunu, G. Q. O. (2022). Deteksi Tangan Otomatis Pada Video Percakapan Bahasa Isyarat Indonesia Menggunakan Metode YOLO Dan CNN. *Journal of Applied Informatics and Computing*, *6*(2), 171–176. https://doi.org/10.30871/jaic.v6i2.4694

Budiman, S. N., Lestanti, S., & Yuana, H. (2023a). *Klasifikasi Alfabet Sistem Isyarat Bahasa Indonesia (SIBI) Menggunakan Computer Vision dan Deep Learning*. Penerbit NEM.

Budiman, S. N., Lestanti, S., & Yuana, H. (2023b). *SIBI (Sistem Bahasa Isyarat Indonesia) berbasis Machine Learning dan Computer Vision untuk Membantu Komunikasi Tuna Rungu dan Tuna Wicara*.

Dharmadi, R. (2018, April 23). Mengenal Convolutional Layer Dan Pooling Layer. *Nodeflux*. https://medium.com/nodeflux/mengenal-convolutional-layer-dan-pooling-layer-3c6f5c393ab2

Fadillah, R. Z., Irawan, A., & Susanty, M. (2021). Data Augmentasi Untuk Mengatasi Keterbatasan Data Pada Model Penerjemah Bahasa Isyarat Indonesia (BISINDO). *JURNAL INFORMATIKA*, *8*(2).

Halder, A. (2021). *Real-time Vernacular Sign Language Recognition using MediaPipe and Machine Learning*.

Han, J.-S., Lee, C.-I., Youn, Y.-H., & Kim, S.-J. (2022). A Study on Real-time Hand Gesture Recognition Technology by Machine Learning-based MediaPipe. *Journal of System and Management Sciences*. https://doi.org/10.33168/JSMS.2022.0225

Hand landmarks detection guide | MediaPipe | Google for Developers. (t.t.). Diambil 21 Januari 2024, dari https://developers.google.com/mediapipe/ solutions/vision/ hand_landmarker

Handhika, T., Zen, R. I. M., Murni, Lestari, D. P., & Sari, I. (2018). Gesture recognition for Indonesian Sign Language (BISINDO). *Journal of Physics: Conference Series*, *1028*, 012173. https://doi.org/10.1088/1742-6596/1028/1/012173

Indriani, Harris, Moh., & Agoes, A. S. (2021). *Applying Hand Gesture Recognition for User Guide Application Using MediaPipe:* 2nd International Seminar of Science and Applied Technology (ISSAT 2021), Bandung, Indonesia. https://doi.org/ 10.2991/aer.k.211106.017

Klobility. (2019, September 23). *Klobility - BISINDO dan SIBI: Apa Bedanya?* Klobility. https://www.klobility.id/post/perbedaan-bisindo-dan-sibi

KM, K., & NR, S. (2022). RECOGNIZATION OF HAND GESTURES USING MEDIAPIPE HANDS. *International Research Journal of Modernization in Engineering Technology and Science*, *04*(06), 8.

Nur'azizan, A. H., Ardiansyah, A. R., & Fernandis, R. (2024). *Implementasi Deteksi Bahasa Isyarat Tangan Menggunakan OpenCV dan MediaPipe*. *3*.

Putri, H. M., Fadlisyah, F., & Fuadi, W. (2022). PENDETEKSIAN BAHASA ISYARAT INDONESIA SECARA REAL-TIME MENGGUNAKAN LONG SHORT-TERM MEMORY (LSTM). *Jurnal Teknologi Terapan and Sains 4.0*, *3*(1), Article 1. https://doi.org/10.29103/tts.v3i1.6853

Saha, S. (2022, November 16). *A Comprehensive Guide to Convolutional Neural Networks—The ELI5 way*. Medium. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Takahashi, K. (2020). *Hand-gesture-recognition-using-mediapipe* [Jupyter Notebook, Python]. https://github.com/Kazuhito00/hand-gesture-recognition-using-mediapipe.git

Ur Rehman, M., Ahmed, F., Attique Khan, M., Tariq, U., Abdulaziz Alfouzan, F., M. Alzahrani, N., & Ahmad, J. (2022). Dynamic Hand Gesture Recognition Using 3D-CNN and LSTM Networks. *Computers, Materials & Continua*, *70*(3), 4675–4690. https://doi.org/10.32604/cmc.2022.019586.

Yolanda, D., Gunadi, K., & Setyati, E. (2020). Pengenalan Alfabet Bahasa Isyarat Tangan Secara Real-Time dengan Menggunakan Metode Convolutional Neural Network dan Recurrent Neural Network. *Jurnal Infra*, *8*(1), Article 1.