

CITRA DIGITAL
APLIKASI SIGN LANGUAGE DETECTOR
BERBASIS ANDROID MENGGUNAKAN MOBILENETV2

disusun oleh :

- | | | | |
|--------------------------------|------------------|----------------|---------------|
| 1. Nurthariqa Octaviani | 201351105 | Ketua | Pagi A |
| 2. Octavia Saramitha P | 201351106 | Anggota | Pagi A |



PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TINGGI TEKNOLOGI WASTUKANCANA
PURWAKARTA

2023

DAFTAR ISI

DAFTAR ISI	i
DAFTAR TABEL	ii
DAFTAR GAMBAR	iii
I. Business Understanding.....	1
A. Latar Belakang Masalah	1
B. Tujuan	1
C. Solusi.....	1
D. Pengguna	1
II. Data Understanding	1
A. Sumber Data.....	1
B. Deskripsi Data.....	4
C. Explore Data	4
D. Kualitas Data	5
III. Data Preparation.....	6
A. Pemilihan Data.....	6
B. Data Preprocessing.....	6
C. Data Issue.....	7
IV. Modeling.....	8
A. Model	8
B. Proses Pembagian Dataset	8
C. Proses Pelatihan Model.....	9
D. Validasi	10
V. Evaluasi	11
VI. Deployment.....	12
A. Save Model	12
B. Android	12

DAFTAR TABEL

Tabel 4. 1 Proses Pelatihan Model.....	10
Tabel 4. 2 Validasi Model	11

DAFTAR GAMBAR

Gambar 2. 1 Sumber Data <i>Source Code</i>	2
Gambar 2. 2 Kode Pembuatan Dataset	2
Gambar 2. 3 <i>American Sign Language Dataset</i>	3
Gambar 2. 4 Dataset <i>Shape</i>	4
Gambar 2. 5 Format Dataset	4
Gambar 2. 6 Hasil Gambar Visualisai Data	5
Gambar 2. 7 Visualisai Jumlah Gambar Dari Setiap Kelas	5
Gambar 2. 8 Kualitas Data Gambar ‘A’	6
Gambar 2. 9 Kode ImageDataGenerator	6
Gambar 4. 1 Kode Model MobileNetV2.....	8
Gambar 4. 2 Kode train_test_split	8
Gambar 4. 3 Hasil Pembagian Data.....	9
Gambar 4. 4 Kode Proses Pelatihan Model Dan Validasi.....	10
Gambar 5. 1 Evaluasi.....	11
Gambar 5. 2 Evaluasi Precision, Recall, Dan F1-Score.	12
Gambar 6. 1 Save Model.....	12
Gambar 6. 2 Framework Flutter.....	13
Gambar 6. 3 Proses Deployment	13
Gambar 6. 4 Sign Language Detector App	13

I. Business Understanding

A. Latar Belakang Masalah

Komunikasi merupakan sebuah penghubung antara dua hal bertujuan untuk dapat mengerti satu sama lain. Komunikasi sangat erat hubungannya dengan makhluk hidup seperti manusia yang selalu bersosial dimana dapat sedikit diartikan sebagai manusia yang ingin menyampaikan suatu hal pada manusia lain. Salah satu komunikasi yang digunakan adalah non verbal dengan cara menyampaikan informasi dalam kata-kata yang disampaikan dalam bahasa tubuh atau menggunakan gerakan pola pada *gesture* tangan manusia yang memiliki arti disetiap pola bentuknya.

Berkomunikasi menggunakan bahasa isyarat merupakan hal yang asing bagi kebanyakan orang dan sangat sedikit orang yang paham bagaimana berkomunikasi menggunakan bahasa isyarat karena bukan bahasa yang wajib dipelajari. Perkara ini menjadi masalah bagi orang-orang berkebutuhan khusus untuk berinteraksi dan berkomunikasi menggunakan bahasa isyarat.

Berdasarkan permasalahan yang diuraikan diatas, maka dibutuhkan sistem pengenalan bahasa isyarat abjad untuk mengklasifikasikan peragaan bahasa isyarat berdasarkan kelas abjad secara *real-time*. Sehingga penyandang tuna rungu mamou memperagakan bahasa isyarat secara langsung melalui webcam.

B. Tujuan

Pengolahan objek secara *real-time* diperlukan untuk dapat mengembangkan fungsi dari teknologi ini agar digunakan untuk tujuan komunikasi dan menyampaikan pesan. Tujuannya sebagai penelitian dasar untuk membangun aplikasi penerjemah bahasa isyarat ke dalam bentuk teks yang dapat dikembangkan lebih lanjut untuk membantu memecahkan masalah komunikasi penyandang tuna rungu.

C. Solusi

Untuk membantu orang-orang yang berkebutuhan khusus tanpa memerlukan cara yang memerlukan biaya yang sangat mahal yaitu menggunakan penerjemah manusia, maka dengan adanya sistem deteksi bahasa isyarat secara *real time* ini orang-orang yang tidak tahu bahasa isyarat ke dalam bentuk *text* sehingga orang-orang akan mengerti apa yang ingin disampaikan oleh orang yang memiliki kebutuhan khusus.

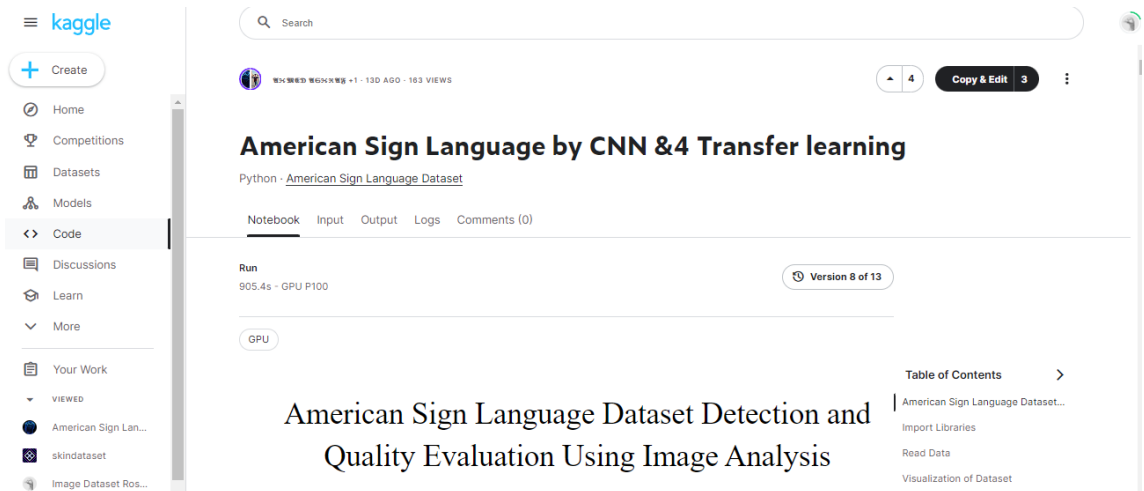
D. Pengguna

Sistem ini tidak hanya dapat digunakan oleh pengguna berkebutuhan khusus saja melainkan semua orang yang ingin belajar bahasa isyarat agar dapat berkomunikasi dengan baik juga dapat menggunakannya.

II. Data Understanding

A. Sumber Data

Pada proses data understanding, visualisasi, data preparation, dan evaluasi menggunakan sumber data dari kaggle dengan link <https://www.kaggle.com/code/ahmedashrafahmed/american-sign-language-by-cnn-4-transfer-learning?scriptVersionId=155176661>



Gambar 2. 1 Sumber Data *Source Code*

Dataset pada penelitian kali ini membuat langsung dengan menggunakan library OpenCV (cv2) untuk membuat dataset gambar dari kamera webcam.

```
import cv2
import os

def create_dataset(output_path, label, num_samples):
    # Membuka kamera
    cap = cv2.VideoCapture(0)

    # Membuat folder jika belum ada
    output_folder = os.path.join(output_path, label)
    os.makedirs(output_folder, exist_ok=True)

    # Mengambil gambar dari webcam
    for i in range(num_samples):
        ret, frame = cap.read()
        cv2.imshow('Capturing Images (Press "q" to stop)', frame)

        # Simpan gambar
        image_path = os.path.join(output_folder, f"{label}_{i}.jpg")
        cv2.imwrite(image_path, frame)

        # Tunggu 100 milidetik (0.1 detik)
        cv2.waitKey(100)

    # Menutup kamera dan jendela
    cap.release()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    # Atur parameter sesuai kebutuhan Anda
    output_path = "C:/Users/TODAY/VSCODE/Deteksi/dataset" # Folder tempat dataset akan disimpan
    label = "A" # Label untuk gambar yang diambil
    num_samples = 100 # Jumlah gambar yang akan diambil

    # Membuat dataset
    create_dataset(output_path, label, num_samples)
```

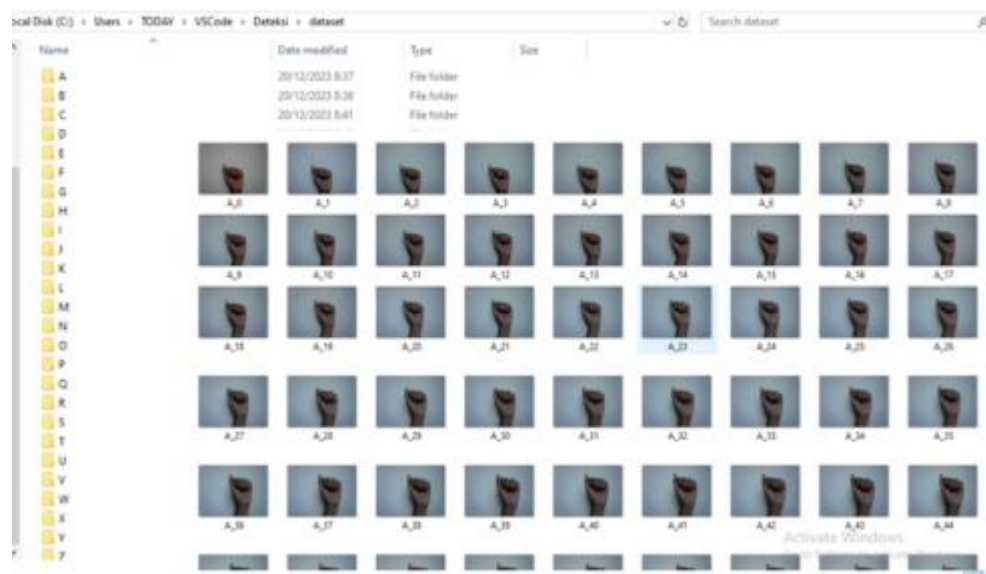
Gambar 2. 2 Kode Pembuatan Dataset

Fungsi kode

- `cap = cv2.VideoCapture(0)`, membuka kamera dengan nomor 0, yang biasanya merupakan kamera utama.

- `output_folder = os.path.join(output_path, label)` `os.makedirs(output_folder, exist_ok=True)`
untuk membuat folder output (tempat untuk menyimpan dataset) berdasarkan label.
- `for i in range(num_samples):` `ret, frame = cap.read()``cv2.imshow('Capturing Images (Press "q" to stop)', frame)`
Mengambil gambar dari webcam sebanyak `num_samples` kali.
Menampilkan setiap gambar dalam jendela dengan judul "Capturing Images (Press 'q' to stop)".
- `image_path = os.path.join(output_folder, f'{label}_{i}.jpg')` `cv2.imwrite(image_path, frame)`
Menyimpan setiap gambar ke dalam folder output dengan format nama file "label_nomor.jpg".
- `cv2.waitKey(100)`
Menunggu 100 milidetik (0.1 detik) sebelum mengambil gambar berikutnya.
- `cap.release()`
`cv2.destroyAllWindows()`
Menutup kamera dan jendela setelah selesai mengambil gambar.
- `if __name__ == "__main__":`
 `output_path = "C:/Users/TODAY/VSCode/Deteksi/dataset"`
 `label = "3"`
 `num_samples = 100`
 `create_dataset(output_path, label, num_samples)`
 kode ini menjalankan fungsi `create_dataset` dengan parameter yang telah diatur sesuai kebutuhan.
 - **output_path**: Path folder tempat dataset akan disimpan.
 - **label**: Label untuk gambar yang diambil.
 - **num_samples**: Jumlah gambar yang akan diambil.

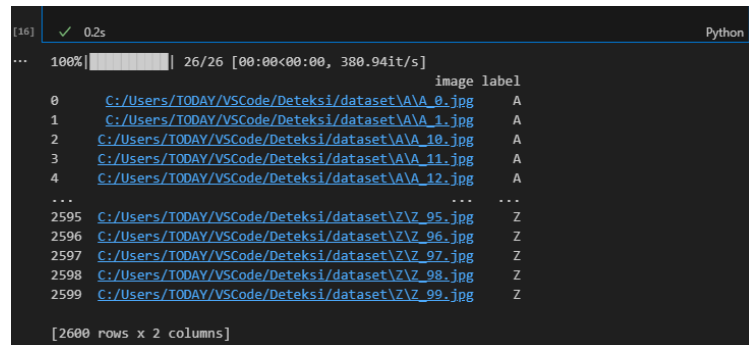
Dataset dibuat dengan *american sign language alfabet A-Z*, hasilnya sebagai berikut



Gambar 2. 3 *American Sign Language Dataset*

B. Deskripsi Data

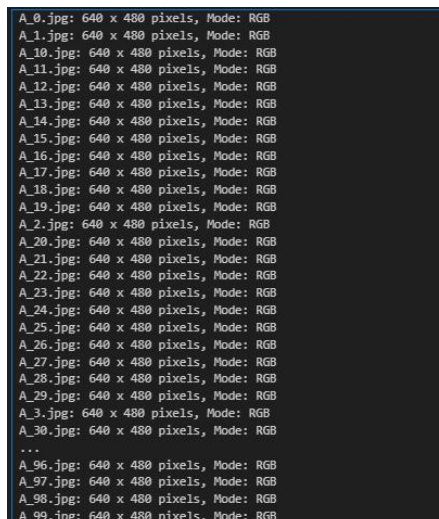
Dataset yang dipakai adalah berupa data kumpulan gambar alfabet dari Bahasa isyarat amerika, dipisahkan dalam 26 folder yang mewakili berbagai kelas. Masing-masing citra dari setiap folder ada 100 citra. Kumpulan data berisi 2600 citra berukuran 640x480 piksel. Ada 26 kelas, berisi huruf A-Z. Format citra digital yang digunakan adalah RGB. Dataset ini memiliki 2600 baris dan 2 kolom.



```
[16] ✓ 0.2s Python
... 100%|██████████| 26/26 [00:00<00:00, 380.94it/s]
      image label
0      C:/Users/TODAY/VSCode/Deteksi/dataset\A\A_0.jpg  A
1      C:/Users/TODAY/VSCode/Deteksi/dataset\A\A_1.jpg  A
2      C:/Users/TODAY/VSCode/Deteksi/dataset\A\A_10.jpg  A
3      C:/Users/TODAY/VSCode/Deteksi/dataset\A\A_11.jpg  A
4      C:/Users/TODAY/VSCode/Deteksi/dataset\A\A_12.jpg  A
...
2595   C:/Users/TODAY/VSCode/Deteksi/dataset\Z\Z_95.jpg  Z
2596   C:/Users/TODAY/VSCode/Deteksi/dataset\Z\Z_96.jpg  Z
2597   C:/Users/TODAY/VSCode/Deteksi/dataset\Z\Z_97.jpg  Z
2598   C:/Users/TODAY/VSCode/Deteksi/dataset\Z\Z_98.jpg  Z
2599   C:/Users/TODAY/VSCode/Deteksi/dataset\Z\Z_99.jpg  Z

[2600 rows x 2 columns]
```

Gambar 2. 4 Dataset *Shape*

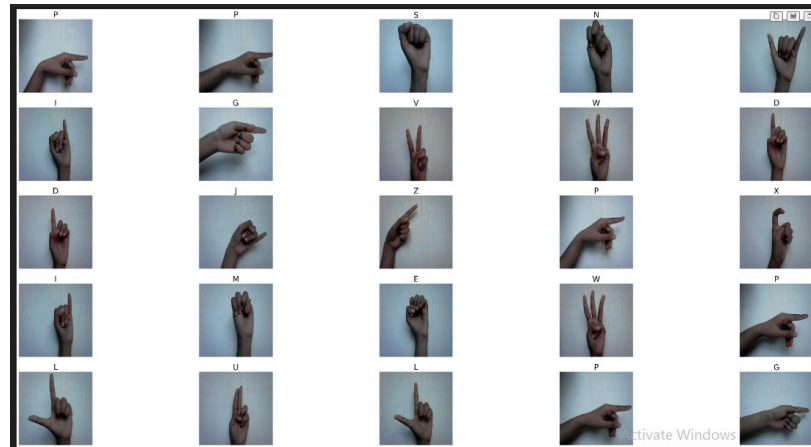


```
A_0.jpg: 640 x 480 pixels, Mode: RGB
A_1.jpg: 640 x 480 pixels, Mode: RGB
A_10.jpg: 640 x 480 pixels, Mode: RGB
A_11.jpg: 640 x 480 pixels, Mode: RGB
A_12.jpg: 640 x 480 pixels, Mode: RGB
A_13.jpg: 640 x 480 pixels, Mode: RGB
A_14.jpg: 640 x 480 pixels, Mode: RGB
A_15.jpg: 640 x 480 pixels, Mode: RGB
A_16.jpg: 640 x 480 pixels, Mode: RGB
A_17.jpg: 640 x 480 pixels, Mode: RGB
A_18.jpg: 640 x 480 pixels, Mode: RGB
A_19.jpg: 640 x 480 pixels, Mode: RGB
A_2.jpg: 640 x 480 pixels, Mode: RGB
A_20.jpg: 640 x 480 pixels, Mode: RGB
A_21.jpg: 640 x 480 pixels, Mode: RGB
A_22.jpg: 640 x 480 pixels, Mode: RGB
A_23.jpg: 640 x 480 pixels, Mode: RGB
A_24.jpg: 640 x 480 pixels, Mode: RGB
A_25.jpg: 640 x 480 pixels, Mode: RGB
A_26.jpg: 640 x 480 pixels, Mode: RGB
A_27.jpg: 640 x 480 pixels, Mode: RGB
A_28.jpg: 640 x 480 pixels, Mode: RGB
A_29.jpg: 640 x 480 pixels, Mode: RGB
A_3.jpg: 640 x 480 pixels, Mode: RGB
A_30.jpg: 640 x 480 pixels, Mode: RGB
...
A_96.jpg: 640 x 480 pixels, Mode: RGB
A_97.jpg: 640 x 480 pixels, Mode: RGB
A_98.jpg: 640 x 480 pixels, Mode: RGB
A_99.jpg: 640 x 480 pixels, Mode: RGB
```

Gambar 2. 5 Format Dataset

C. Explore Data

Visualisasi data dilakukan oleh penulis agar orang lebih mudah memahami sesuatu secara visual. Berikut hasil visualisasi data gambar :



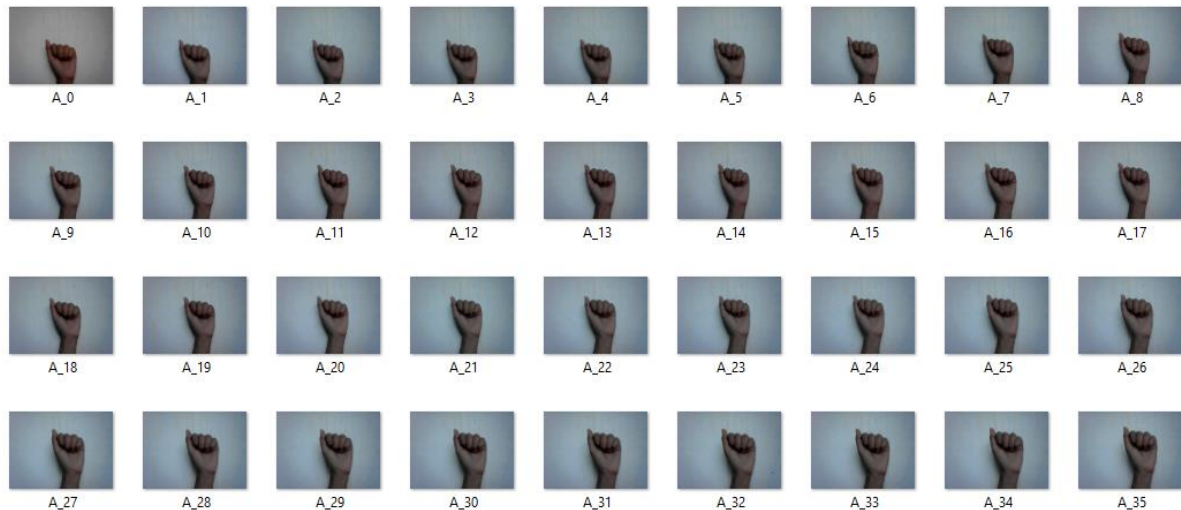
Gambar 2. 6 Hasil Gambar Visualisai Data



Gambar 2. 7 Visualisai Jumlah Gambar Dari Setiap Kelas

D. Kualitas Data

Kualitas data yang digunakan sangat bagus dan banyak.



Gambar 2. 8 Kualitas Data Gambar ‘A’

III. Data Preparation

A. Pemilihan Data

Pengerjaan proyek ini menggunakan dataset yang telah dibuat yang merupakan basisdata citra dengan format JPG dalam jumlah besar dengan komponen warna RGB. Citra yang sudah dilakukan cropping dan resizeing citra agar mudah diimplementasikan yaitu dengan ukuran per citra adalah sekitar 640x480 pixel.

B. Data Preprocessing

Kita membuat generator data menggunakan **ImageDataGenerator** dari Keras untuk melatih, menguji, dan memvalidasi model dengan augmentasi gambar. Berikut kode nya :

```
image_size = (224,224)
batch_size = 32
datagen = ImageDataGenerator(
    rescale=1./255,
    horizontal_flip=True
)
train_generator = datagen.flow_from_dataframe(
    df_train,
    x_col='image',
    y_col='label',
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=True
)
test_generator = datagen.flow_from_dataframe(
    df_test,
    x_col='image',
    y_col='label',
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)
val_generator = datagen.flow_from_dataframe(
    df_val,
    x_col='image',
    y_col='label',
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=True
)
```

Gambar 2. 9 Kode ImageDataGenerator

Pertama mengatur **image_size**, ukuran gambar yang diinginkan setelah diubah ukurannya dan **batch_size**, jumlah sampel gambar yang diberikan ke model pada setiap iterasi pelatihan. Lalu menggunakan ImageDataGenerator, **rescale=1./255**, mereskalasi nilai piksel gambar menjadi rentang 0-1 untuk meningkatkan konvergensi. **horizontal_flip=True**,

melakukan flip horizontal secara acak pada gambar. Ini adalah salah satu bentuk augmentasi yang umum digunakan.

Selanjutnya pembuatan generator untuk pelatihan, pengujian, dan validasi.

- `flow_from_dataframe`: Membuat generator aliran data dari DataFrame.
- `df_train`: DataFrame yang berisi informasi gambar dan label untuk pelatihan.
- `x_col='image'`: Nama kolom dalam DataFrame yang berisi path/gambar.
- `y_col='label'`: Nama kolom dalam DataFrame yang berisi label kelas.
- `target_size`: Ukuran gambar yang diharapkan oleh model.
- `batch_size`: Jumlah sampel yang dihasilkan oleh generator pada setiap iterasi.
- `class_mode='categorical'`: Mode kelas untuk penggunaan label kategori yang bersifat one-hot encoded.
- `shuffle=True`: Mengacak urutan data pada setiap epoch.

Proses yang sama juga diterapkan pada generator pengujian (`test_generator`) dan generator validasi (`val_generator`). Perbedaannya adalah pengujian tidak perlu diacak (`shuffle=False`) agar dapat diukur dengan baik, dan validasi diacak (`shuffle=True`).

C. Data Issue

Beberapa tambahan informasi dari pembuatan dataset yang dibuat sendiri, kemungkinan masalah yang mungkin timbul :

1. Keterbatasan Variasi Posisi Tangan
Gambar hanya diambil dari sudut depan jauh dan dekat, model mungkin tidak mampu mengenali bahasa isyarat dari sudut atau posisi lainnya. Variasi dalam posisi tangan (seperti sudut samping, sudut atas, dll.) sangat penting untuk mengajarkan model mengenali isyarat dari berbagai sudut.
2. Ketidakmampuan Model untuk Mengatasi Perubahan Skala
Hanya mengambil gambar dari sudut jauh dan dekat mungkin membuat model kurang mampu mengenali isyarat dengan benar saat tangan berada pada jarak yang berbeda.
3. Dataset terlalu terfokus pada kondisi tertentu
Hanya gambar tangan dekat atau jauh, model yang dibangun mungkin menjadi overfitting pada kondisi tersebut dan tidak dapat mengenali isyarat dalam situasi yang berbeda.

Untuk mengatasi masalah ini, penelitian selanjutnya dapat melakukan beberapa hal:

1. Mengumpulkan Data dari Berbagai Sudut dan Posisi
Pastikan dataset mencakup variasi sudut dan posisi tangan yang mencerminkan keberagaman pengguna yang mungkin menggunakannya.
2. Augmentasi Data
Gunakan teknik augmentasi data untuk memperkaya variasi dalam dataset. Ini bisa mencakup rotasi, pemangkasan, dan transformasi lainnya.
3. Pertimbangkan Penggunaan Kamera Multisudut
Jika memungkinkan, pertimbangkan menggunakan lebih dari satu kamera untuk mendapatkan gambar dari sudut yang berbeda secara bersamaan.
4. Perhatikan Variabilitas Pencahayaan
Pastikan bahwa dataset mencakup variasi pencahayaan yang mungkin terjadi pada penggunaan sehari-hari.

Memastikan keberagaman dan representasi yang baik dalam dataset adalah kunci untuk membuat model yang kuat dan dapat diandalkan untuk penelitian mengenali bahasa isyarat dari berbagai kondisi.

IV. Modeling

A. Model

Model yang dipakai adalah MobileNetV2. MobileNetV2 adalah arsitektur model neural network yang dirancang khusus untuk keperluan pengolahan citra di perangkat mobile atau berkinerja rendah. Model ini dikenal efisien dan ringan sehingga sangat cocok untuk aplikasi di perangkat mobile. Berikut kode programnya :

```
model_MobileNet = tf.keras.applications.MobileNetV2(
    include_top=False,
    input_shape=(224, 224, 3),
    weights='imagenet'
)
model_MobileNet.trainable = False

# Membuat model dengan input dari gambar
input_layer = tf.keras.layers.Input(shape=(224, 224, 3))
x = model_MobileNet(input_layer, training=False)
x = tf.keras.layers.Flatten()(x)
x = tf.keras.layers.Dense(256, activation='relu')(x)
x = tf.keras.layers.Dropout(0.3)(x)
output = tf.keras.layers.Dense(36, activation='softmax')(x)

model_MobileNet = tf.keras.Model(inputs=[input_layer], outputs=[output])

# Menampilkan ringkasan model
model_MobileNet.summary()
```

Gambar 4. 1 Kode Model MobileNetV2

Model ini adalah model klasifikasi gambar menggunakan MobileNetV2 sebagai feature extractor dan menambahkan lapisan Dense kustom untuk klasifikasi kelas. Model ini digunakan untuk klasifikasi gambar menjadi salah satu dari 36 kelas yang dihasilkan oleh lapisan output softmax. Model ini telah diinisialisasi dengan bobot yang telah dilatih pada dataset ImageNet, dan lapisan MobileNetV2 tersebut tidak akan dilatih ulang (trainable=False).

B. Proses Pembagian Dataset

Pembagian dataset menjadi subset pelatihan, validasi, dan pengujian (train-validation-test split) langkah awal proses pra-pemrosesan data. Selain itu, kita menulis beberapa kode dalam fungsi **train_test_split** yang merupakan preprocessing data:

```
X_train, X_test1, y_train, y_test1 = train_test_split(df['image'], df['label'], test_size=0.3, random_state=42, shuffle=True, stratify=df['label'])
X_val, X_test, y_val, y_test = train_test_split(X_test1, y_test1, test_size=0.5, random_state=42, shuffle=True, stratify=y_test1)
df_train = pd.DataFrame({'image': X_train, 'label': y_train})
df_test = pd.DataFrame({'image': X_test, 'label': y_test})
df_val = pd.DataFrame({'image': X_val, 'label': y_val})
```

Gambar 4. 2 Kode train_test_split

- Pembagian dataset utama
X_train, X_test1, y_train, y_test1 = train_test_split(df['image'], df['label'], test_size=0.3, random_state=42, shuffle=True, stratify=df['label'])
- Memisahkan dataset utama (df) menjadi subset pelatihan (X_train, y_train) dan subset pengujian (X_test1, y_test1).

- `test_size=0.3`: Menunjukkan bahwa 30% dari data akan digunakan sebagai subset pengujian.
- `random_state=42`: Menetapkan seed untuk memastikan reproduktibilitas (hasil yang sama setiap kali kode dijalankan).
- `shuffle=True`
- Kode ini mengacak urutan data sebelum pembagian. Pengacakan dapat membantu mencegah model mengingat urutan spesifik dari data dan dapat meningkatkan kegeneralisasian model.
- `stratify=df['label']`
Kode ini memastikan bahwa pembagian dataset mempertahankan distribusi kelas yang sama di antara subset pelatihan dan pengujian. Kode ini penting terutama jika dataset memiliki kelas yang tidak seimbang. Stratifikasi dapat membantu meningkatkan performa model pada kelas-kelas yang kurang banyak representasinya.

Dengan melakukan pembagian ini, Anda dapat menggunakan `df_train` untuk melatih model, `df_val` untuk mengevaluasi dan menyesuaikan model selama pelatihan, dan `df_test` untuk menguji performa model pada data yang belum pernah dilihatnya sebelumnya.

Dari Gambar 4. 2 terdapat kode yang dijelaskan `ImageDataGenerator` menghasilkan data 1820 gambar yang dijadikan pelatihan, 390 gambar yang dijadikan pengujian, dan 390 gambar yang dijadikan validasi. Generator data telah berhasil menemukan dan memvalidasi jumlah tertentu dari gambar-gambar yang ada didalam dataset untuk digunakan dalam proses pelatihan, pengujian, dan validasi model.

```
Found 1820 validated image filenames belonging to 26 classes.
Found 390 validated image filenames belonging to 26 classes.
Found 390 validated image filenames belonging to 26 classes.
```

Gambar 4. 3 Hasil Pembagian Data

C. Proses Pelatihan Model

Pada proses pelatihan model melibatkan memasukkan data pelatihan ke dalam model, menghitung nilai kehilangan (`loss`), melakukan optimisasi untuk mengupdate bobot model, dan mengulangi proses ini sejumlah epoch (iterasi melalui seluruh dataset).

```

# Menentukan parameter pelatihan
batch_size = 32
num_epochs = 10

# Menentukan optimizer, fungsi loss, dan metrik evaluasi
model_MobileNet.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

# Melatih model
history = model_MobileNet.fit(
    train_generator,
    epochs=num_epochs,
    validation_data=val_generator,
    batch_size=batch_size,
    callbacks=[early_stopping_cb, checkpoint_cb]
)

```

Gambar 4. 4 Kode Proses Pelatihan Model Dan Validasi

Pelatihan model ada pada kode

Tabel 4. 1 Proses Pelatihan Model

Kode	Penjelasan Kode
<pre> model_MobileNet.fit(train_generator, epochs=num_epochs, batch_size=batch_size, callbacks=[early_stopping_cb, checkpoint_cb]) </pre>	<ul style="list-style-type: none"> • <code>train_generator</code>: Generator data pelatihan yang digunakan untuk memberikan data pelatihan ke model. • <code>epochs=num_epochs</code>: Jumlah epoch atau iterasi melalui seluruh dataset pelatihan. • <code>batch_size=batch_size</code>: Jumlah sampel yang digunakan dalam satu iterasi pelatihan. • <code>callbacks=[early_stopping_cb, checkpoint_cb]</code>: Callbacks yang digunakan selama pelatihan, seperti Early Stopping dan Model Checkpoint.

D. Validasi

Proses ini dilakukan untuk mengukur seberapa baik model bekerja pada data yang tidak digunakan selama pelatihan. Ini memberikan gambaran tentang kemampuan model untuk menggeneralisasi dari data pelatihan ke data yang belum pernah dilihat sebelumnya. Validasi membantu mendeteksi tanda-tanda overfitting. Overfitting terjadi ketika model mempelajari detail yang sangat spesifik pada data pelatihan dan tidak dapat menggeneralisasi dengan baik pada data baru. Jika performa pada data validasi lebih rendah daripada data pelatihan, ini bisa menjadi indikasi overfitting. Validasi memungkinkan kita untuk memilih model terbaik dari serangkaian model yang dilatih. Model terbaik adalah model yang memberikan kinerja optimal pada data validasi. Dari gambar 2.13 berikut penjelasan kode pada tahap validasi :

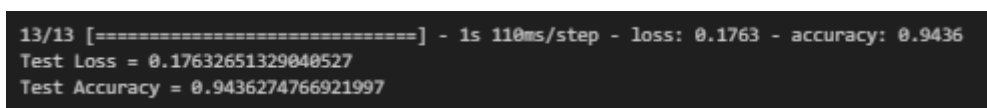
Tabel 4. 2 Validasi Model

Kode	Penjelasan
<pre>history = model_MobileNet.fit(train_generator, epochs=num_epochs, validation_data=val_generator, batch_size=batch_size, callbacks=[early_stopping_cb, checkpoint_cb])</pre>	<ul style="list-style-type: none"> Validasi model dilakukan pada setiap epoch menggunakan dataset validasi (val_generator). Hasil validasi, termasuk metrik seperti akurasi, akan dicatat dalam history.

Dengan menggunakan generator data (train_generator dan val_generator), model dilatih pada setiap epoch dan diukur performanya pada dataset validasi.

V. Evaluasi

Berikut adalah hasil evaluasi



```
13/13 [=====] - 1s 110ms/step - loss: 0.1763 - accuracy: 0.9436
Test Loss = 0.17632651329040527
Test Accuracy = 0.9436274766921997
```

Gambar 5. 1 Evaluasi

Hasil Loss (Kehilangan) sebesar 0.1763, loss adalah nilai yang dihitung oleh fungsi kehilangan selama evaluasi pada dataset pengujian. Semakin rendah nilai loss, semakin baik modelnya. Dalam penelitian ini, nilai loss adalah 0.1763, yang dapat dianggap relatif rendah.

Hasil Accuracy (Akurasi) sebesar 94.36% (0.9436), akurasi adalah proporsi prediksi yang benar dari keseluruhan prediksi. Pada model ini, akurasi sebesar 94.36% menunjukkan bahwa model berhasil mengklasifikasikan sebagian besar sampel dengan benar pada dataset pengujian.

Selain akurasi dan loss, perlu juga memperhatikan metrik lain seperti precision, recall, dan F1-score, terutama jika dataset memiliki kelas yang tidak seimbang. Berikut informasi lebih lanjut mengenai precision, recall, dan F1-score.

Classification Report is :				precision	recall	f1-score	support
0	0.83	0.83	0.83	0.83	12		
1	0.85	0.92	0.88	0.88	12		
2	0.69	0.92	0.79	0.79	12		
3	0.92	0.92	0.92	0.92	12		
4	1.00	0.92	0.96	0.96	12		
5	0.92	1.00	0.96	0.96	12		
6	1.00	0.83	0.91	0.91	12		
7	0.92	0.92	0.92	0.92	12		
8	1.00	1.00	1.00	1.00	12		
9	1.00	1.00	1.00	1.00	12		
10	1.00	0.92	0.96	0.96	12		
11	1.00	1.00	1.00	1.00	12		
12	1.00	1.00	1.00	1.00	12		
13	0.86	1.00	0.92	0.92	12		
14	0.92	1.00	0.96	0.96	12		
15	1.00	1.00	1.00	1.00	12		
16	1.00	1.00	1.00	1.00	12		
17	1.00	1.00	1.00	1.00	12		
18	1.00	1.00	1.00	1.00	12		
19	0.92	1.00	0.96	0.96	12		
20	1.00	1.00	1.00	1.00	11		
21	1.00	1.00	1.00	1.00	11		
22	1.00	0.80	0.89	0.89	10		
...							
accuracy				0.94	408		
macro avg	0.94	0.94	0.94	0.94	408		
weighted avg	0.94	0.94	0.94	0.94	408		

Gambar 5. 2 Evaluasi Precision, Recall, Dan F1-Score.

VI. Deployment

A. Save Model

Sebelum di deploy, model perlu dikonversi ke dalam format TensorFlow Lite (TFLite). Karena format ini memiliki ukuran yang lebih kecil, dan dioptimalkan untuk perangkat mobile dan perangkat Internet of Things (IoT).

```
import tensorflow as tf

# Load model
model = tf.keras.models.load_model('ASL_model.h5')

# Convert the model
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Save the TensorFlow Lite model to a file
with open('ASL_model.tflite', 'wb') as f:
    f.write(tflite_model)
```

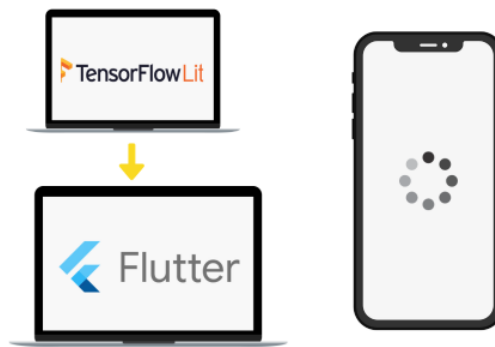
Gambar 6. 1 Save Model

TensorFlow Lite (TFLite) adalah versi ringan dari TensorFlow yang dirancang khusus untuk ponsel, perangkat IoT, dan lingkungan yang membutuhkan ukuran model yang lebih kecil dan efisiensi inferensi yang lebih tinggi.

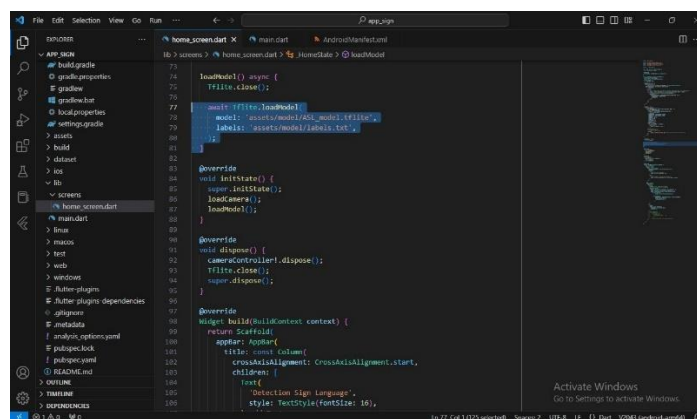
B. Android

Flutter adalah framework pengembangan aplikasi UI yang dapat digunakan untuk membuat aplikasi lintas platform (iOS dan Android) dengan menggunakan bahasa

pemrograman Dart. Dengan menggunakan paket TensorFlow Lite untuk Flutter dapat mengintegrasikan model TFLite ke dalam aplikasi Flutter.

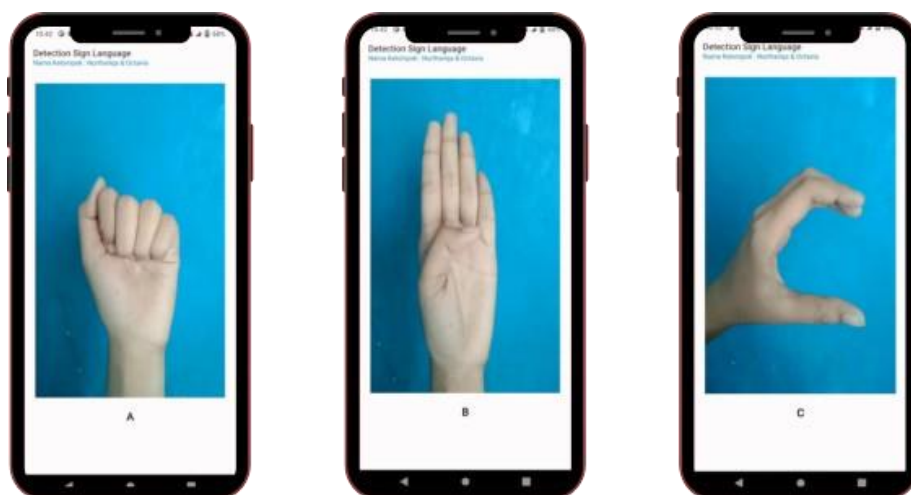


Gambar 6. 2 Framework Flutter



Gambar 6. 3 Proses Deployment

Lalu setelah model tflite berhasil dibuat, di implementasikan di Flutter untuk bisa di implementasikan ke mobile. Model disimpan di assets dan di simpan, library **tflite_flutter** untuk mengintegrasikan TensorFlow Lite ke dalam proyek Flutter di file pubspec.yaml, dan jangan lupa untuk mengatur pengaturan di file build.gradle untuk versi android, tflite, dan kebutuhan lainnya. Lalu setelah selesai, melakukan build apk, berikut hasilnya ketika diimplementasikan di android.



Gambar 6. 4 Sign Language Detector App

Link Kode Program Dan Apk

https://drive.google.com/drive/folders/1_IPFEaA5R5CY-EfxDQepWV5UtBWP5y0k?usp=sharing