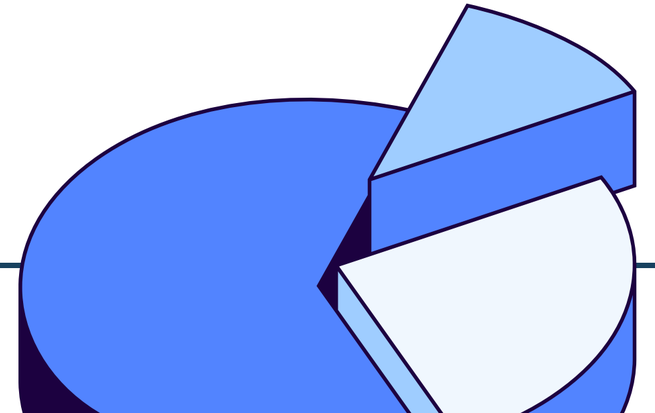
Palm Recognition Prototype

FINANCIAL TECHNOLOGY

LYAN NAZHABIL D. (2308428)
MOCHAMAD ZIDAN R. (2305464)



About Dataset

Sapienza University Mobile Palmprint Database (SMPD) yang prosedur akuisisi imagenya dilakukan di Sapienza dan Tor Vergata universities of Rome dari tangan kanan 92 mahasiswa internasional. Dalam dataset ini, jarak antara telapak tangan dan kamera disesuaikan agar kualitas image terbaik dapat diperoleh. Para mahasiswa diminta untuk mengambil 10 image dalam empat sudut pandang:

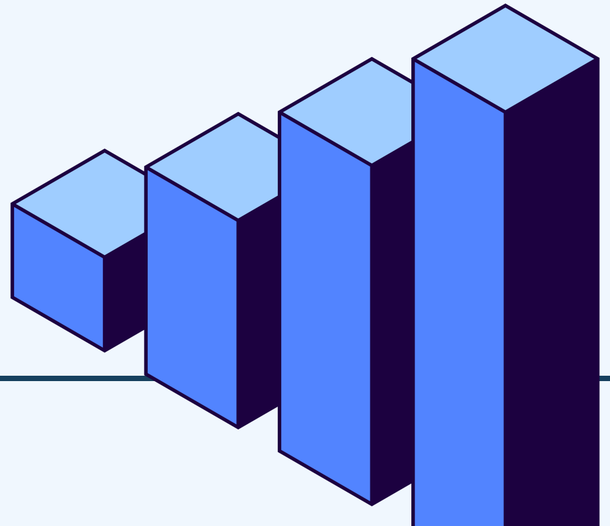
a) Tampilan depan (Frontal view),

b) Tampilan depan yang diputar (Rotated Frontal view),

c) Tampilan perspektif (Perspective view), dan

d) Tampilan perspektif yang diputar (Rotated Perspective view).

Sumber dataset: https://www.kaggle.com/datasets/mahdieizadpanah/sapienza-university-mobile-palmprint-databasesmpd/data

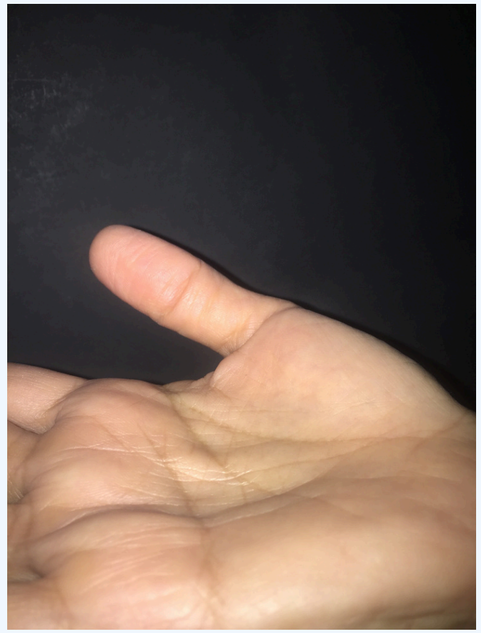


Contoh Images

Frontal view

Rotated Frontal view

Perspective view



Rotated Perspective view

Library yang digunakan

```python
import os
import cv2
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, Model
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

cv2 untuk menyiapkan dan membersihkan data gambar, lalu TensorFlow/Keras menggunakan data tersebut untuk membangun dan melatih model agar bisa mengenali telapak tangan.

Eksplorasi Dataset

```
image_count = len(image_paths)
print(f"Total images found: {image_count}")

# Get class names (individual IDs)
class_names_list = sorted([item.name for item in data_dir.glob('*') if item.is_dir()])
num_classes = len(class_names_list)
print(f"Total classes (individuals): {num_classes}")

print("\nSample of class names (IDs):", class_names_list[:10])

# Show image count per class (sample of first 10 classes)
print("\nImage count per class (sample):")
for class_name in class_names_list[:10]:
    class_dir = data_dir / class_name
    count = 0
    for ext in image_extensions:
        count += len(list(class_dir.glob(ext)))
    print(f"- ID {class_name}: {count} images")
```

Hasilnya:

Total images found: 3757

Total classes (individuals): 94

Sample of class names (IDs): ['001', '002', '003', '004', '005', '006', '007', '008', '009', '010']

Image count per class (sample):

- ID 001: 40 images
- ID 002: 40 images
- ID 003: 40 images

Region of Interest (ROI)

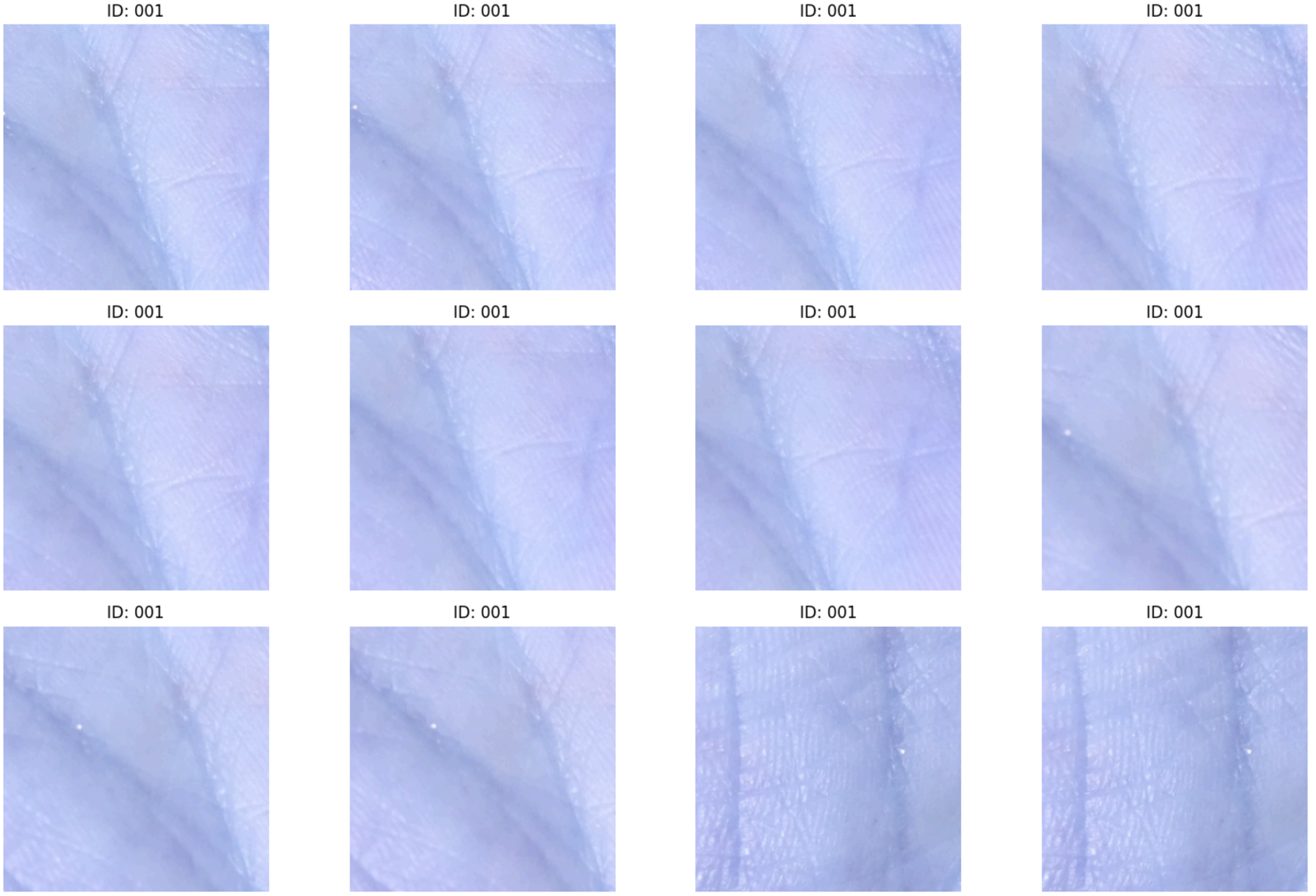
```python
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
blurred = cv2.GaussianBlur(gray, (7, 7), 0)
_, thresh = cv2.threshold(blurred, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

if not contours:
    return None

hand_contour = max(contours, key=cv2.contourArea)
```

```python

M = cv2.moments(hand_contour)
if M["m00"] == 0:
    return None
cx = int(M["m10"] / M["m00"])
cy = int(M["m01"] / M["m00"])

# Crop a larger fixed-size square to provide more detail for the 256x256 resize
CROP_SIZE = 400
half_crop = CROP_SIZE // 2

```

Cara Kerjanya:

1. Isolasi Tangan: Gambar diubah menjadi hitam-putih untuk memisahkan bentuk tangan dari background.
2. Temukan Pusat: Menemukan kontur tangan terbesar dan menghitung titik tengahnya (pusat telapak tangan).
3. Potong (Crop): Mengambil area persegi tepat di tengah telapak tangan yang sudah ditemukan.
4. Standarisasi Ukuran: Mengubah ukuran hasil potongan menjadi 256x256 piksel agar semua data seragam.

Sample Palm ROI Images from Dataset



Siapkan Data Train & Validation

```python
# Train-Validation Split
X_train, X_val, y_train, y_val = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)
```

Dataset split:

- Training samples: 3005
- Validation samples: 752
- Training shape: (3005, 256, 256, 3)
- Validation shape: (752, 256, 256, 3)

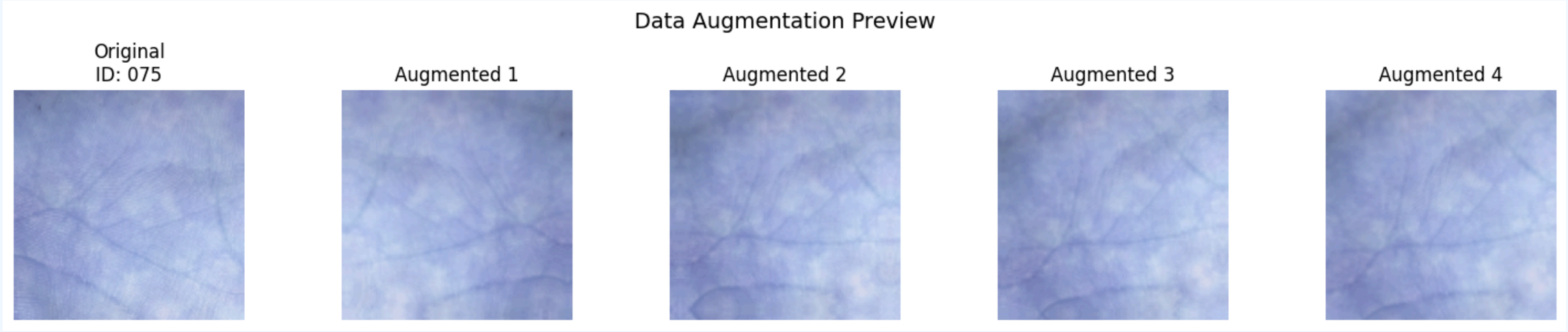
```python
train_ds = tf.data.Dataset.from_tensor_slices((X_train, y_train))
train_ds = train_ds.shuffle(1000).batch(BATCH_SIZE).cache().prefetch(tf.data.AUTOTUNE)

val_ds = tf.data.Dataset.from_tensor_slices((X_val, y_val))
val_ds = val_ds.batch(BATCH_SIZE).cache().prefetch(tf.data.AUTOTUNE)
```

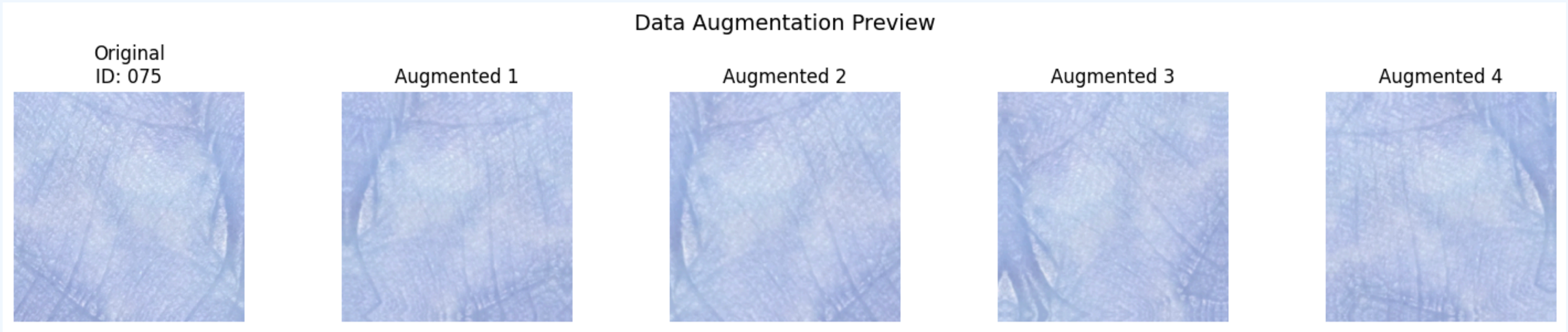
Data Augmentation

```python
data_augmentation = tf.keras.Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1),
], name='augmentation')
```

Data augmentation adalah teknik untuk menciptakan data latihan baru secara artifisial dari data yang sudah ada. Tujuannya adalah agar model menjadi lebih pintar dan tidak overfitting (terlalu menghafal data asli).

- Membalik gambar secara horizontal (efek cermin).
- Memutar gambar sedikit secara acak.
- Melakukan sedikit zoom in atau zoom out secara acak.


Data Augmentation Preview
Original
ID: 075
Augmented 1
Augmented 2
Augmented 3
Augmented 4


Data Augmentation Preview
Original
ID: 075
Augmented 1
Augmented 2
Augmented 3
Augmented 4

Build Base Model

```
base_model = tf.keras.applications.MobileNetV2(
    input_shape=IMG_SHAPE,
    include_top=False,
    weights='imagenet'
)
base_model._name = 'mobilenetv2_base'
base_model.trainable = False  # Freeze base model initially
```

Menggunakan teknik Transfer Learning dengan meminjam arsitektur MobileNetV2, sebuah model canggih yang sudah dilatih oleh Google pada jutaan gambar dari dataset ImageNet.

Build Base Model

```python
# Build complete model architecture
inputs = tf.keras.Input(shape=IMG_SHAPE)
x = layers.Rescaling(1./255)(inputs)
x = data_augmentation(x)
x = base_model(x, training=False)
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dropout(0.3)(x)
outputs = layers.Dense(len(class_names), activation='softmax')(x)

model = tf.keras.Model(inputs, outputs)
```

- Gambar masuk ke model, nilainya dinormalisasi (piksel 0-255 diubah menjadi 0-1), dan diperbanyak dengan augmentasi.
- Gambar kemudian dimasukkan ke "otak" MobileNetV2 untuk diekstrak fitur-fitur pentingnya (pola, tekstur, garis).
- Fitur-fitur yang kompleks diringkas menjadi sebuah vektor yang lebih sederhana.
- Lapisan Dropout lalu mematikan beberapa koneksi secara acak untuk mencegah model overfitting.

Compile Model

```python
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

model.summary()
```

Menggunakan Adam Optimizer, Loss Function menggunakan Sparse Categorical Crossentropy karena klasifikasi multi-kelas

Model: "functional_1"

Layer (type)	Output Shape	Param #
input_layer_2 (InputLayer)	(None, 128, 128, 3)	0
rescaling (Rescaling)	(None, 128, 128, 3)	0
augmentation (Sequential)	(None, 128, 128, 3)	0
mobilenetv2_1.00_128 (Functional)	(None, 4, 4, 1280)	2,257,984
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1280)	0
dropout (Dropout)	(None, 1280)	0
dense (Dense)	(None, 92)	117,852

Total params: 2,375,836 (9.06 MB)
Trainable params: 117,852 (460.36 KB)
Non-trainable params: 2,257,984 (8.61 MB)

Training Model

```
history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=EPOCHS_FEATURE_EXTRACTION,
    callbacks=callbacks,
    verbose=1
)
```

- Belajar dari train_ds: Model akan melihat gambar dan label di data latihan untuk mengenali polanya.
- Diuji dengan validation_data: Setelah setiap siklus belajar, performa model akan diuji pada data validasi untuk memastikan ia tidak hanya "menghafal".
- Ulangi sebanyak epochs

Hasilnya:
--- Phase 1 Training Complete ---
Completed 50 epochs
Final training accuracy: 0.8196
Final validation accuracy: 0.6875

Fine Tuning

```python
# Unfreeze base model for fine-tuning
base_model = model.get_layer('mobilenetv2_base')
base_model.trainable = True

# Freeze early layers, only fine-tune top layers
fine_tune_at = 100
for layer in base_model.layers[:fine_tune_at]:
    layer.trainable = False

print(f"Fine-tuning from layer {fine_tune_at} onwards...")
print(f"Total layers in base model: {len(base_model.layers)}")
print(f"Trainable layers: {len([layer for layer in base_model.layers if layer.trainable])}")

# Recompile with lower learning rate for fine-tuning3
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=LEARNING_RATE_FINE_TUNE),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

Hasilnya:

--- Phase 2 Fine-Tuning Complete ---
Completed 18 additional epochs
Final training accuracy: 0.9201
Final validation accuracy: 0.7460

Training layer-layer akhir (dari lapisan ke-100 dan seterusnya). Lapisan awal yang mengenali fitur universal (seperti garis dan sudut) tetap dibekukan agar pengetahuannya yang berharga tidak rusak.

Model dikompilasi ulang dengan learning_rate yang sangat kecil. Tujuannya adalah untuk membuat penyesuaian yang sangat halus pada bobot yang sudah terlatih, bukan mengubahnya secara drastis.

Evaluasi Model

```python
# Load best model
best_model = tf.keras.models.load_model('best_palm_model.keras')

# Evaluate on validation set
loss, accuracy = best_model.evaluate(val_ds, verbose=1)
print(f"\nFinal Model Performance:")
print(f"- Validation Loss: {loss:.4f}")
print(f"- Validation Accuracy: {accuracy * 100:.2f}%")
print(f"- Total classes: {len(class_names)}")
print(f"- Training samples: {len(X_train)}")
print(f"- Validation samples: {len(X_val)}")
```

Hasilnya:

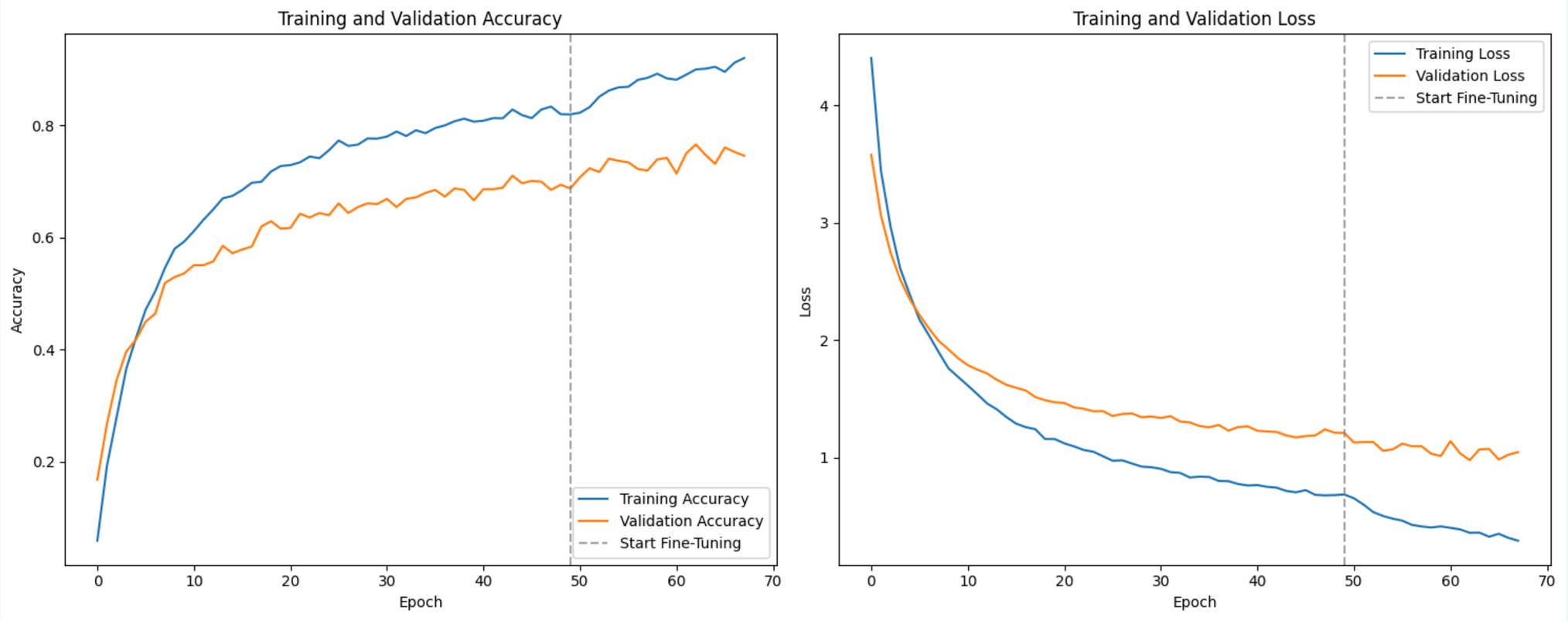
--- Final Model Evaluation ---

24/24 [==============================] - 2s 51ms/step - loss: 0.9776 - accuracy: 0.7660

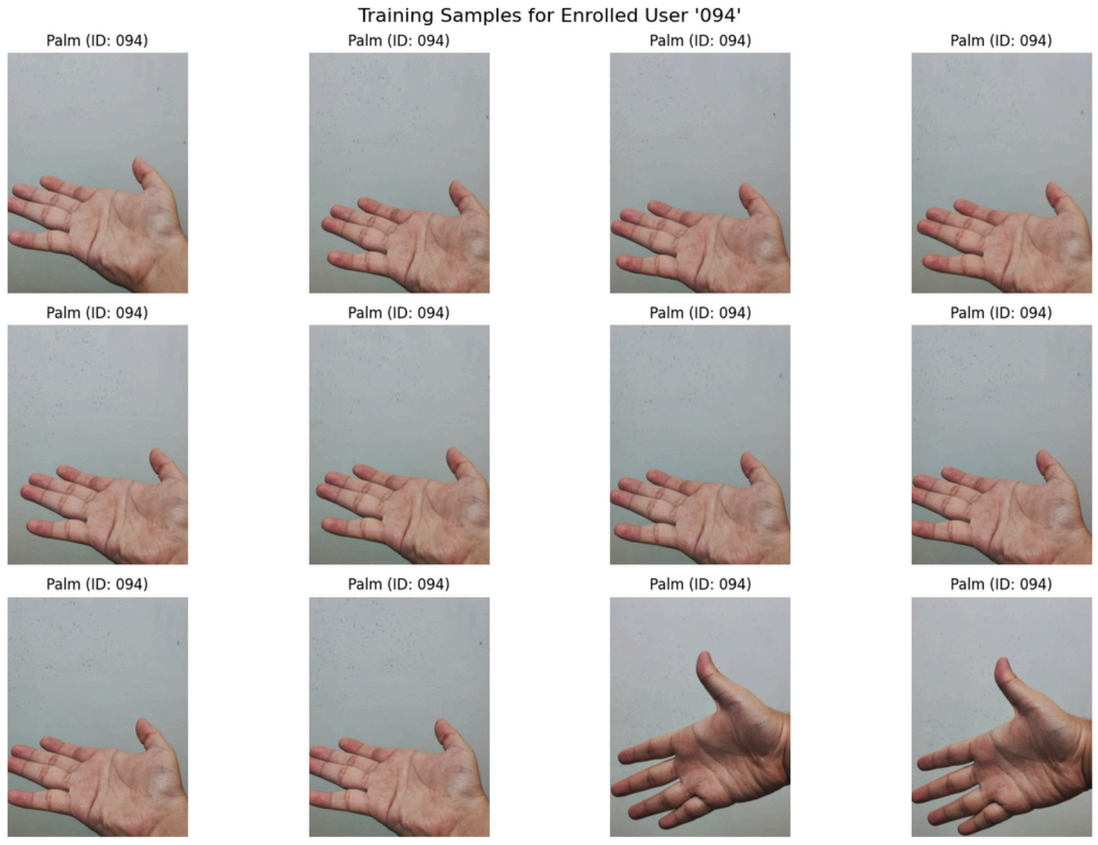
Final Model Performance:

- Validation Loss: 0.9776
- Validation Accuracy: 76.60%
- Total classes: 94
- Training samples: 3005 - Validation samples: 752

Hasil Training



Demo Test

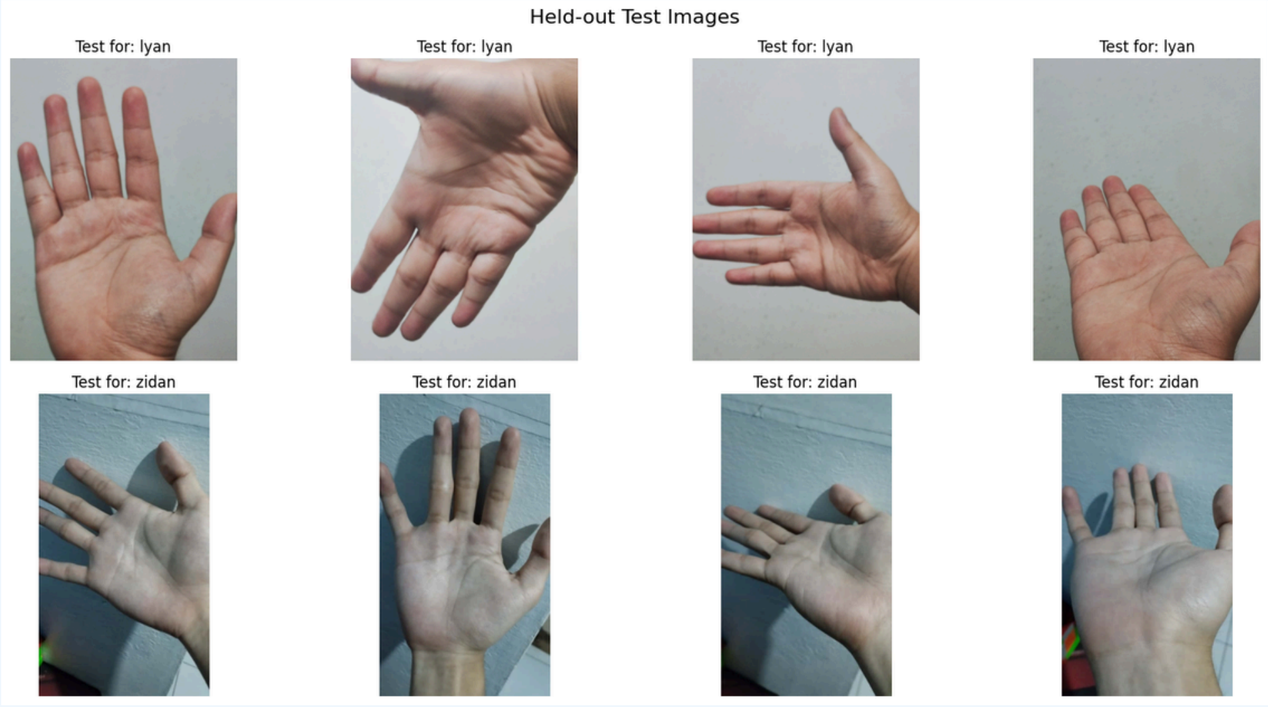

Training Samples for Enrolled User '094'
Palm (ID: 094)
Palm (ID: 094)
Palm (ID: 094)
Palm (ID: 094)
Palm (ID: 094)
Palm (ID: 094)
Palm (ID: 094)
Palm (ID: 094)
Palm (ID: 094)
Palm (ID: 094)
Palm (ID: 094)
Palm (ID: 094)


Training Samples for Enrolled User '095'
Palm (ID: 095)
Palm (ID: 095)
Palm (ID: 095)
Palm (ID: 095)
Palm (ID: 095)
Palm (ID: 095)
Palm (ID: 095)
Palm (ID: 095)
Palm (ID: 095)
Palm (ID: 095)
Palm (ID: 095)
Palm (ID: 095)

Demo Test

```
final_model = tf.keras.models.load_model('best_palm_model.keras')
print(f"Model is ready. It recognizes {len(class_names)} users.")
```

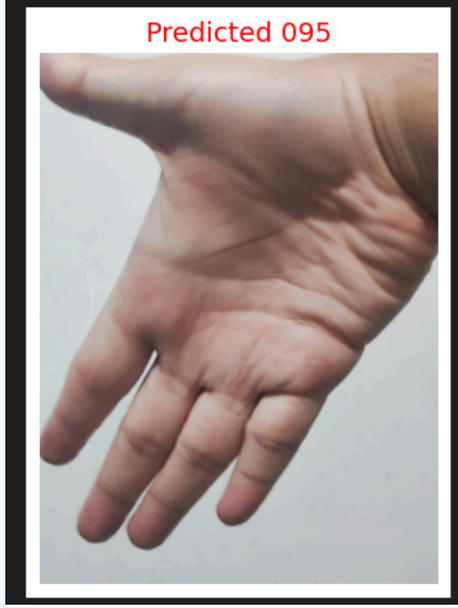
```
img_array = np.expand_dims(roi, axis=0)
predictions = model.predict(img_array, verbose=0)
score = tf.nn.softmax(predictions[0])
predicted_index = np.argmax(score)
predicted_user = class_names_list[predicted_index]

print(f"  Predicted User: '{predicted_user}'")

if predicted_user == expected_user_id:
    print("  Verification Status: CORRECT MATCH ✅")
    result_text = f"Verified: {predicted_user}"
    text_color = 'green'
else:
    print("  Verification Status: INCORRECT MATCH ❌")
    result_text = f"Predicted {predicted_user}"
    text_color = 'red'
```



Demo Test

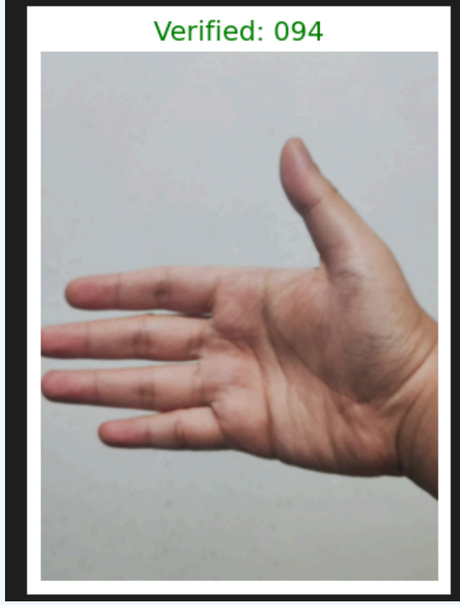
```
\nLoading the best trained model: 'best_palm_model.keras'
Model is ready. It recognizes 94 users.
\n--- Testing image: WhatsApp Image 2025-09-21 at 23.42.43_7dc8bada.jpg (True user: 094) ---
  Predicted User: '095'
  Verification Status: INCORRECT MATCH ❌
```


```
\n--- Testing image: WhatsApp Image 2025-09-21 at 23.42.44_479ed4f7.jpg (True user: 094) ---
  Predicted User: '095'
  Verification Status: INCORRECT MATCH ❌
```



Demo Test

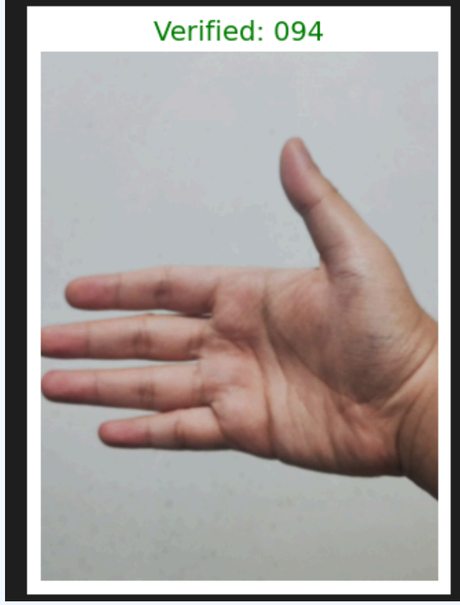
```
\n--- Testing image: WhatsApp Image 2025-09-21 at 23.42.44_9e60cb70.jpg (True user: 094) ---
  Predicted User: '094'
  Verification Status: CORRECT MATCH ✅
```




```
\n--- Testing image: WhatsApp Image 2025-09-21 at 23.42.45_c1af0bd2.jpg (True user: 094)
  Predicted User: '094'
  Verification Status: CORRECT MATCH ✅
```

Demo Test

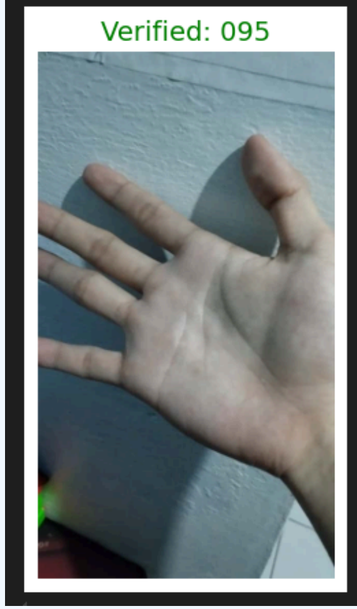
```
\n--- Testing image: WhatsApp Image 2025-09-21 at 23.42.44_9e60cb70.jpg (True user: 094) ---
  Predicted User: '094'
  Verification Status: CORRECT MATCH ✅
```



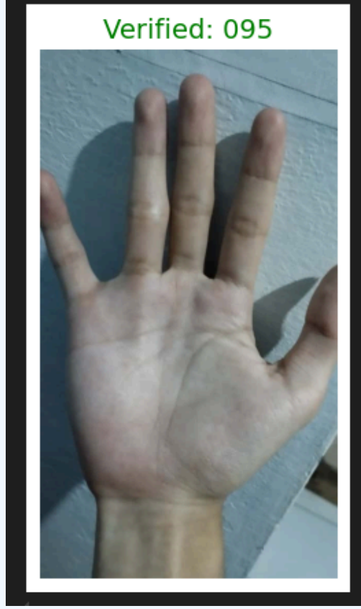

```
\n--- Testing image: WhatsApp Image 2025-09-21 at 23.42.45_c1af0bd2.jpg (True user: 094)
  Predicted User: '094'
  Verification Status: CORRECT MATCH ✅
```

Demo Test

```
\n--- Testing image: WhatsApp Image 2025-09-21 at 20.27.28_5c72fc7b.jpg (True user: 095)
  Predicted User: '095'
  Verification Status: CORRECT MATCH ✅
```

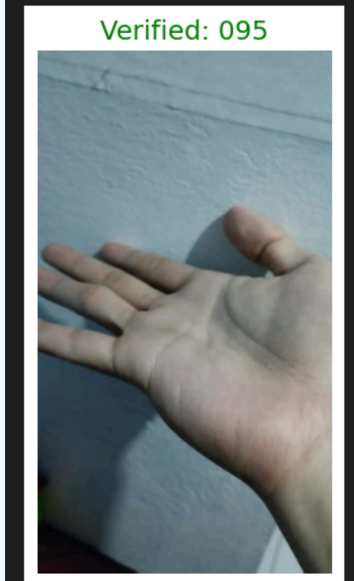



```
\n--- Testing image: WhatsApp Image 2025-09-21 at 20.27.29_5d2c5315.jpg (True user: 095)
  Predicted User: '095'
  Verification Status: CORRECT MATCH ✅
```

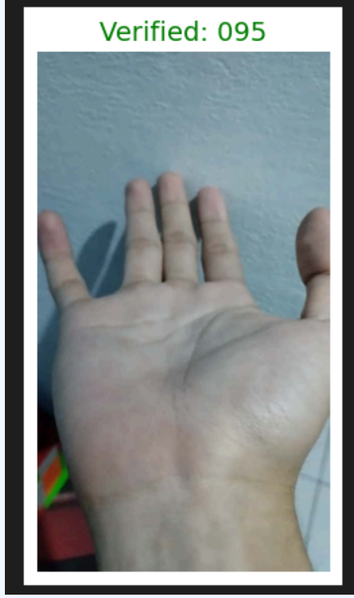


Demo Test

```
\n--- Testing image: WhatsApp Image 2025-09-21 at 20.27.29_5df35ae9.jpg (True user: 095)
  Predicted User: '095'
  Verification Status: CORRECT MATCH ✅
```




```
\n--- Testing image: WhatsApp Image 2025-09-21 at 20.27.29_ebdadfaa.jpg (True user: 095)
  Predicted User: '095'
  Verification Status: CORRECT MATCH ✅
```



Kesimpulan

Kunci keberhasilan implementasi ini terletak pada dua aspek utama. Pertama fungsi ekstraksi Region of Interest (ROI), yang terbukti krusial untuk menyediakan input yang bersih dan konsisten bagi model.
Kedua, hasil eksperimen secara definitif menunjukkan bahwa resolusi image input adalah faktor paling kritis yang memengaruhi performa model. Pada iterasi awal dengan resolusi 128x128, model mengalami information bottleneck dan stagnan pada akurasi di bawah 60%. Namun, setelah resolusi ditingkatkan menjadi 256x256, model berhasil mencapai akurasi validasi akhir sebesar 76.60%, yang berhasil memenuhi target performa yang telah ditetapkan.

Pada tahap demonstrasi, prototipe ini terbukti mampu melakukan tugas identifikasi dengan sukses. Model yang telah dilatih berhasil mengenali pengguna yang telah terdaftar (`lyan` dan `zidan`) dari citra uji baru

Thank You