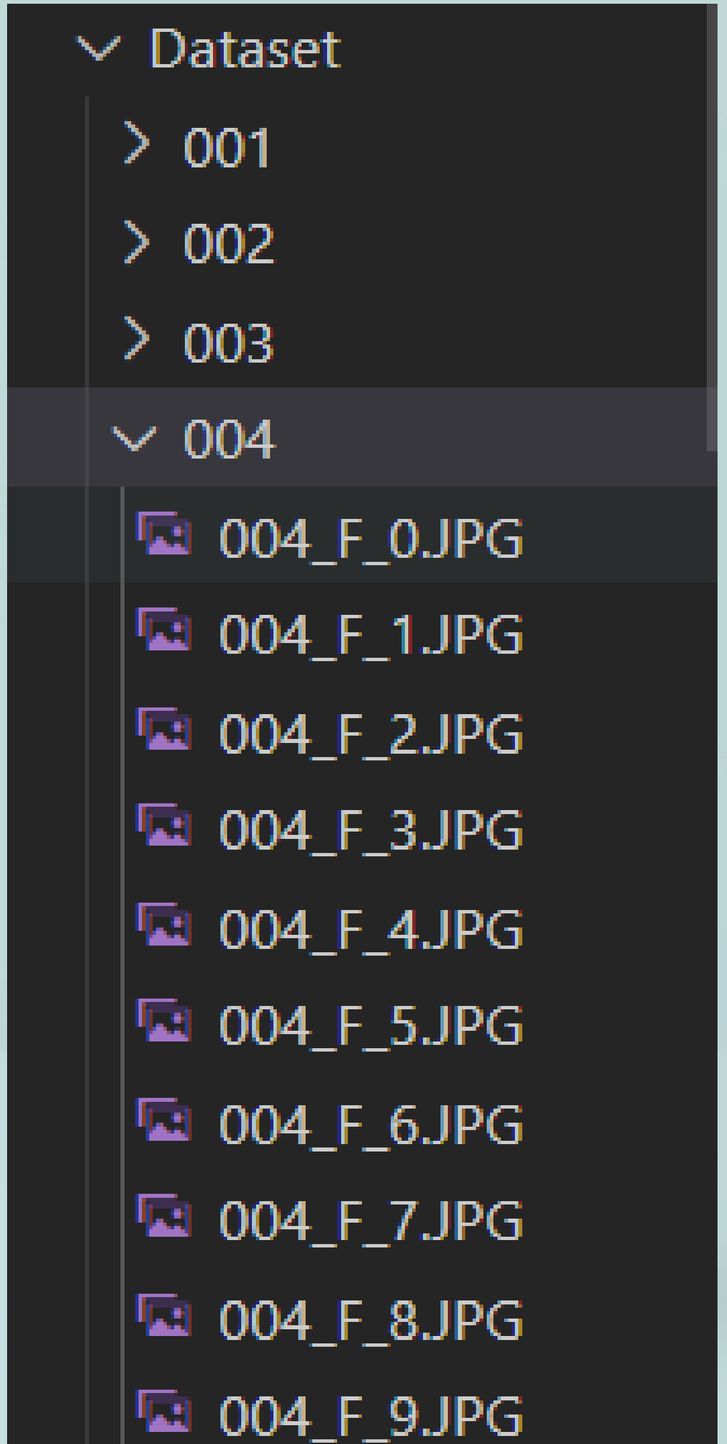
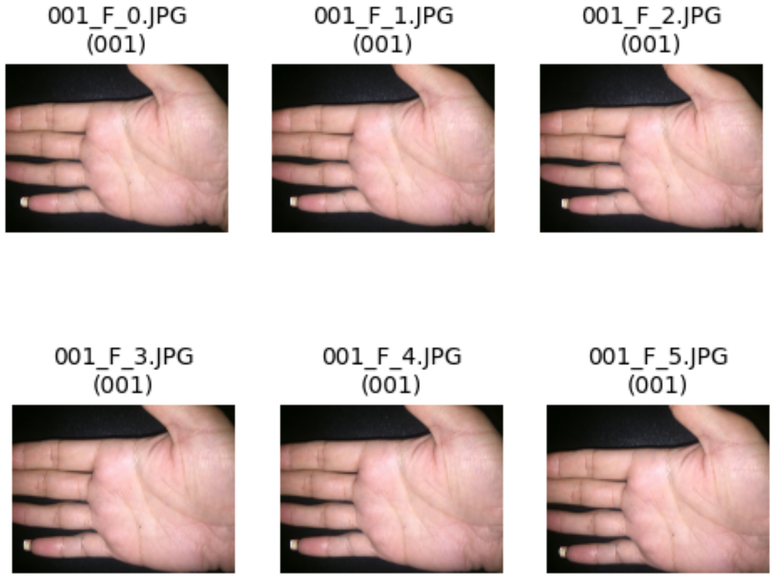
PALM RECOGNITION

MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK



Dataset

Dataset ini dikumpulkan di Universitas Sapienza dan Tor Vergata, Roma, dari telapak tangan kanan 110 mahasiswa internasional. Dataset ini punya 93 folder dan dapat diakses melalui tautan berikut: https://www.kaggle.com/datasets/mahdieizadpanah/sapienza-university-mobile-palmprint-databasesmpd



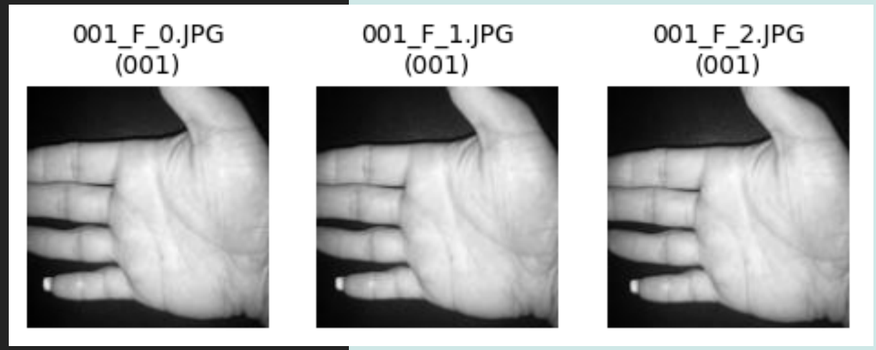
Tahapan utama percobaan:

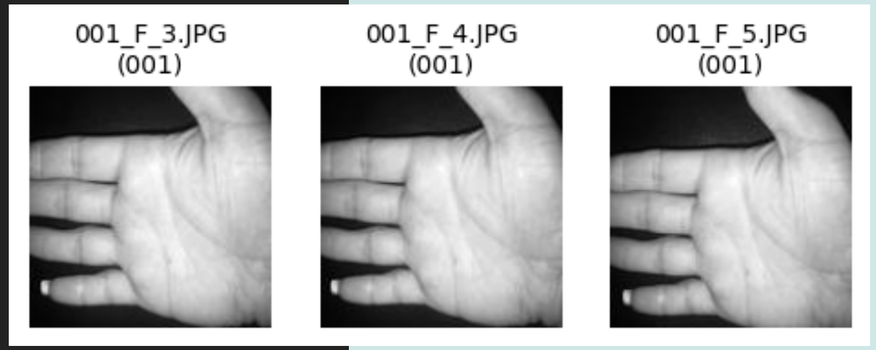
- Persiapan Dataset: Memuat dan mengatur dataset gambar telapak tangan.
- Pra proses Gambar (Preprocessing): Menyiapkan gambar agar optimal untuk pelatihan model, meliputi
  - konversi grayscale,
  - penyesuaian ukuran (resize),
  - peningkatan kontras dengan CLAHE,
  - dan normalisasi ROI.
- Pembuatan dan Pelatihan Model: Merancang, mengompilasi, dan melatih model CNN.
- Evaluasi Model: Menganalisis kinerja model menggunakan metrik seperti akurasi dan loss, serta mengamati hasil prediksi pada data validasi.

Pra Proses Data

Grayscale dan Resize

```python
# Fungsi preprocessing Step 1
def step1_grayscale_resize(img_path):
    img = Image.open(img_path).convert("L")    # Ubah gambar menjadi grayscale (mode "L")
    img = img.resize(target_size)              # Ubah ukuran gambar sesuai nilai target_size (width x height)
    return img                                 # Kembalikan gambar yang sudah diproses

# Proses per batch agar tidak memakai memori terlalu banyak
batch_num = 1
for start in range(0, len(selected_dirs), batch_size):
    batch_folders = selected_dirs[start:start + batch_size]
    print(f"\nStep 1: Memproses batch {batch_num}: {batch_folders}")

    for folder in tqdm(batch_folders, desc="Step1 folders"):
        src_folder = os.path.join(src_path, folder)
        dst_folder = os.path.join(step1_path, folder)
        os.makedirs(dst_folder, exist_ok=True)

        files = []
        for ext in extensions:
            files.extend(glob.glob(os.path.join(src_folder, ext)))

        for f in files:
            try:
                img_proc = step1_grayscale_resize(f)
                fname = os.path.basename(f)
                img_proc.save(os.path.join(dst_folder, fname))
            except Exception as e:
                print("Error Step1:", f, e)

    print(f"Batch {batch_num} selesai.")
    batch_num += 1
```

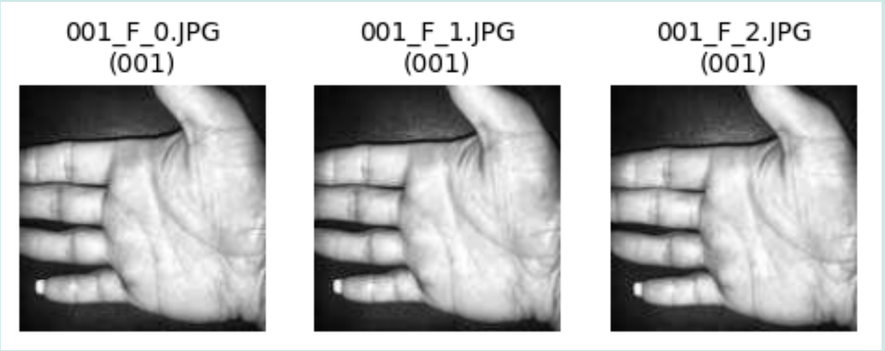


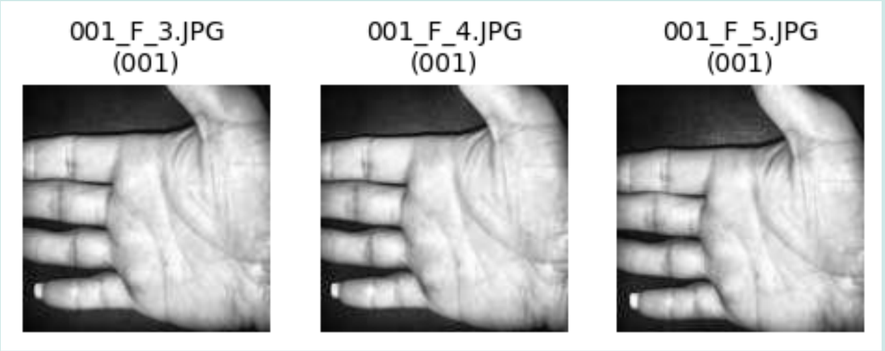


Pra Proses Data

Contrast Limited Adaptive Histogram Equalization (CLAHE)

```python
# Fungsi CLAHE
def step2_clahe(img_path, clipLimit=2.0, tileGridSize=(8,8)):
    # Buka gambar grayscale
    img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)

    # Buat objek CLAHE
    clahe = cv2.createCLAHE(clipLimit=clipLimit, tileGridSize=tileGridSize)
    img_clahe = clahe.apply(img)

    # Kembalikan sebagai PIL Image
    return Image.fromarray(img_clahe)

# Proses setiap folder
for folder in tqdm(selected_dirs, desc="Step2 folders (CLAHE)"):
    src_folder = os.path.join(step1_path, folder)
    dst_folder = os.path.join(step2_path, folder)
    os.makedirs(dst_folder, exist_ok=True)

    files = []
    for ext in extensions:
        files.extend(glob.glob(os.path.join(src_folder, ext)))

    for f in files:
        try:
            img_proc = step2_clahe(f, clipLimit=2.0, tileGridSize=(8,8))
            fname = os.path.basename(f)
            img_proc.save(os.path.join(dst_folder, fname))
        except Exception as e:
```

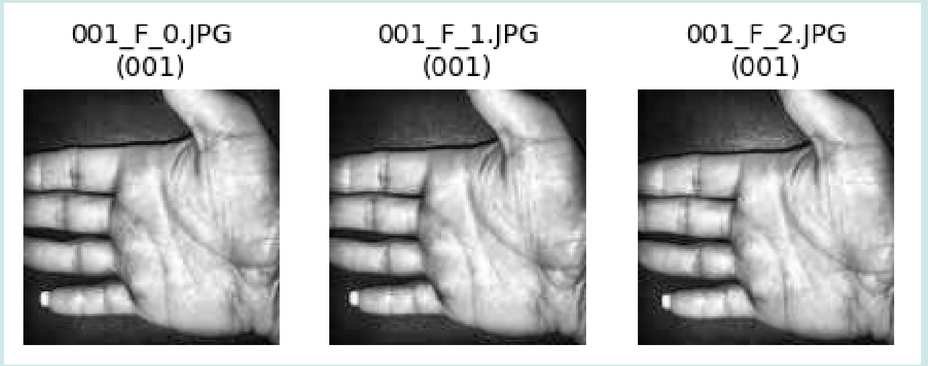


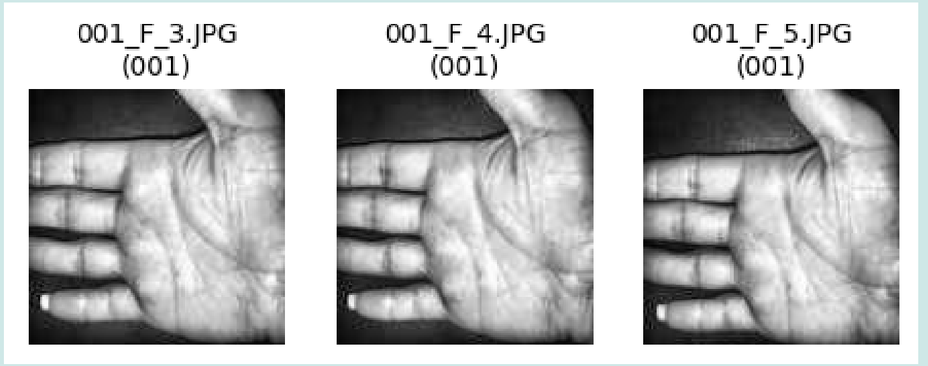


Pra Proses Data

Contrast Limited Adaptive Histogram Equalization (CLAHE) part 2

```python
# Fungsi CLAHE
def step2_clahe(img_path, clipLimit=3.0, tileGridSize=(8,8)):
    # Buka gambar grayscale
    img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)

    # Buat objek CLAHE
    clahe = cv2.createCLAHE(clipLimit=clipLimit, tileGridSize=tileGridSize)
    img_clahe = clahe.apply(img)

    # Kembalikan sebagai PIL Image
    return Image.fromarray(img_clahe)

# Proses setiap folder
for folder in tqdm(selected_dirs, desc="Step2 folders (CLAHE)"):
    src_folder = os.path.join(step1_path, folder)
    dst_folder = os.path.join(step2_path, folder)
    os.makedirs(dst_folder, exist_ok=True)

    files = []
    for ext in extensions:
        files.extend(glob.glob(os.path.join(src_folder, ext)))

    for f in files:
        try:
            img_proc = step2_clahe(f, clipLimit=3.0, tileGridSize=(8,8))
            fname = os.path.basename(f)
            img_proc.save(os.path.join(dst_folder, fname))
        except Exception as e:
```

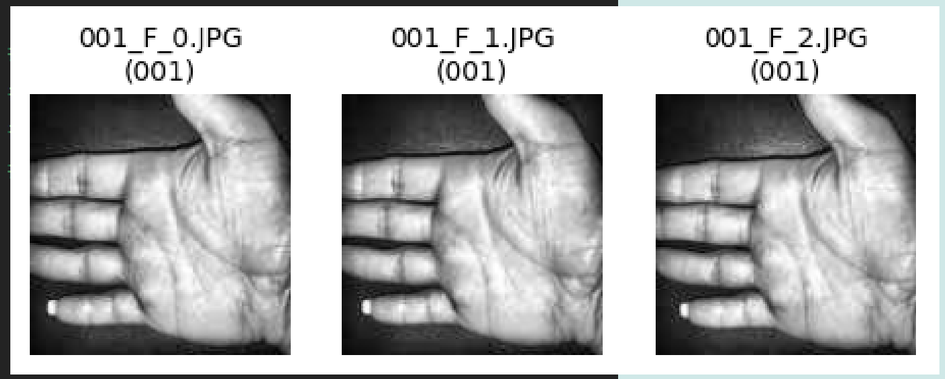


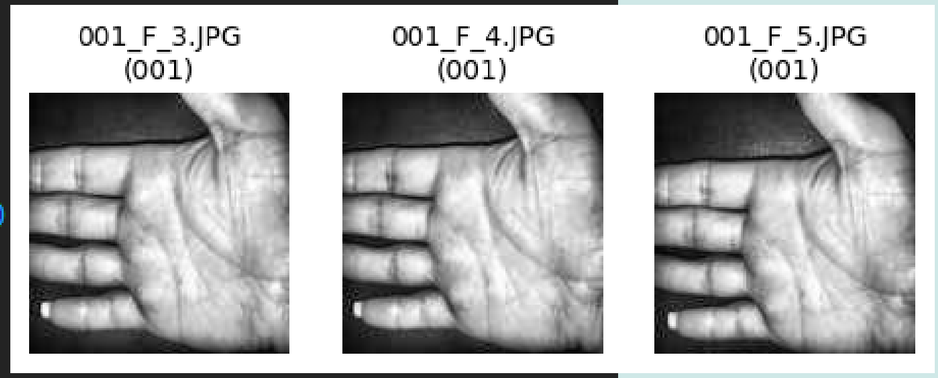


Pra Proses Data

ROI Normalization

```python
# ROI Normalization
def step3_roi_normalize(img_path):
    img = Image.open(img_path)
    img_array = np.array(img, dtype=np.float32) / 255.0
    img_norm = Image.fromarray(np.uint8(img_array * 255))
    return img_norm

print("\nStep 3: ROI Normalization")

# Iterasi setiap folder kelas yang dipilih
for folder in tqdm(selected_dirs, desc="Step3 folders"):
    src_folder = os.path.join(step2_path, folder)
    dst_folder = os.path.join(dst_path, folder)
    os.makedirs(dst_folder, exist_ok=True)

    files = []
    for ext in extensions:
        files.extend(glob.glob(os.path.join(src_folder, ext)))

    for f in files:
        try:
            img_proc = step3_roi_normalize(f)
            fname = os.path.basename(f)
            img_proc.save(os.path.join(dst_folder, fname))
        except Exception as e:
```





Memuat data dan membagi dataset

Load gambar menjadi numpy kemudian dibagi menjadi data train 80% dan data test 20%, untuk model pertama ini memakai gambar yang sudah di grayscale, resize dan kontrasnya sudah ditingkatkan dengan CLAHE

```python
# Path dataset
data_dir = "../Dataset_processed"

# Parameter
img_size = (128, 128)

# Load semua gambar jadi numpy
X = []
y = []
classes = sorted(os.listdir(data_dir))
print("Kelas yang terdeteksi:", classes)

# Loop untuk membaca setiap gambar dari dataset
for label, cls in enumerate(classes):
    cls_path = os.path.join(data_dir, cls)
    for file in os.listdir(cls_path):
        file_path = os.path.join(cls_path, file)
        img = load_img(file_path, target_size=img_size)
        img_array = img_to_array(img) / 255.0   # normalisasi 0-1
        X.append(img_array)
        y.append(label)

# Ubah list menjadi array NumPy dengan tipe float32 (agar efisien dan cocok untuk model)
X = np.array(X, dtype="float32")

# Ubah label (0,1,2,...) menjadi one-hot encoding sesuai jumlah kelas
y = tf.keras.utils.to_categorical(y, num_classes=len(classes))

print("Shape X:", X.shape)  # Tampilkan dimensi data gambar (jumlah, tinggi, lebar, channel)
print("Shape y:", y.shape)  # Tampilkan dimensi label setelah one-hot

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

Model 1

```
# Model CNN sederhana
model = models.Sequential([
    layers.Conv2D(32, (3,3), activation="relu", input_shape=(128,128,3)),  # Conv layer pertama
    layers.MaxPooling2D(2,2),                                              # Pooling untuk reduksi dimensi

    layers.Conv2D(64, (3,3), activation="relu"),                           # Conv layer kedua
    layers.MaxPooling2D(2,2),

    layers.Conv2D(128, (3,3), activation="relu"),                          # Conv layer ketiga
    layers.MaxPooling2D(2,2),

    layers.Flatten(),                                                      # Rata-kan output menjadi vektor
    layers.Dense(256, activation="relu"),
    layers.Dropout(0.5),                                                   # Dropout untuk mengurangi overfitting
    layers.Dense(len(classes), activation="softmax")
])

# Kompilasi model dengan optimizer Adam, loss categorical_crossentropy, dan metrik akurasi
model.compile(
    optimizer="adam",
    loss="categorical_crossentropy",
    metrics=["accuracy"]
)
```

Evaluasi dan Hasil

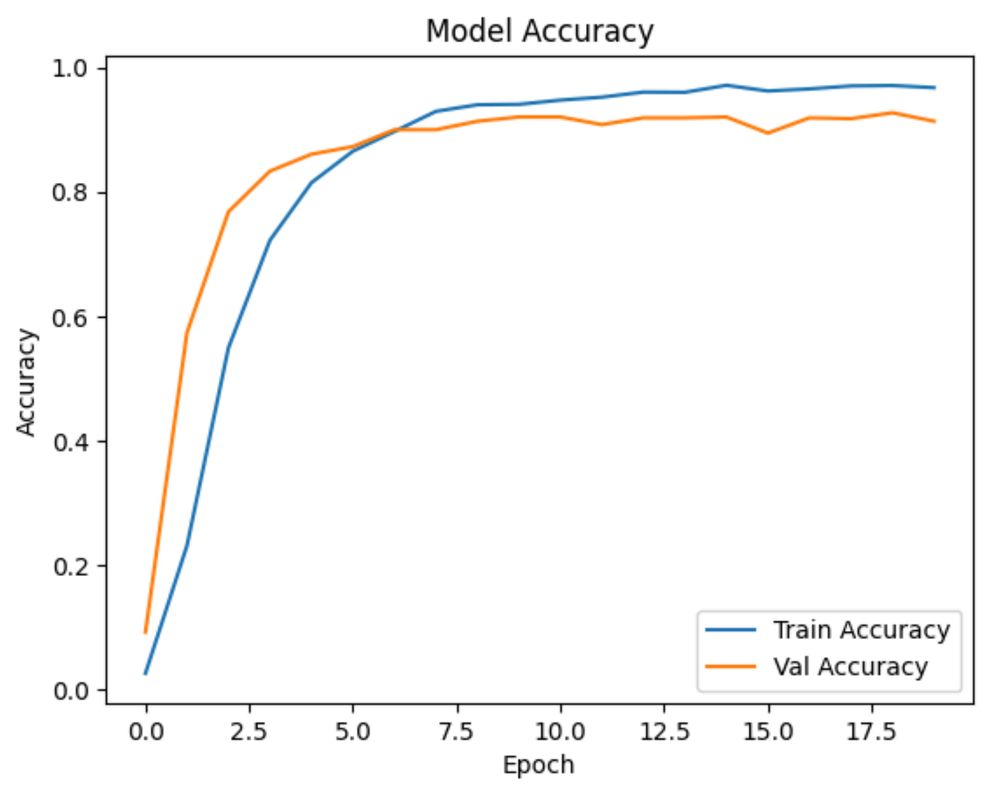
Akurasi dan Loss

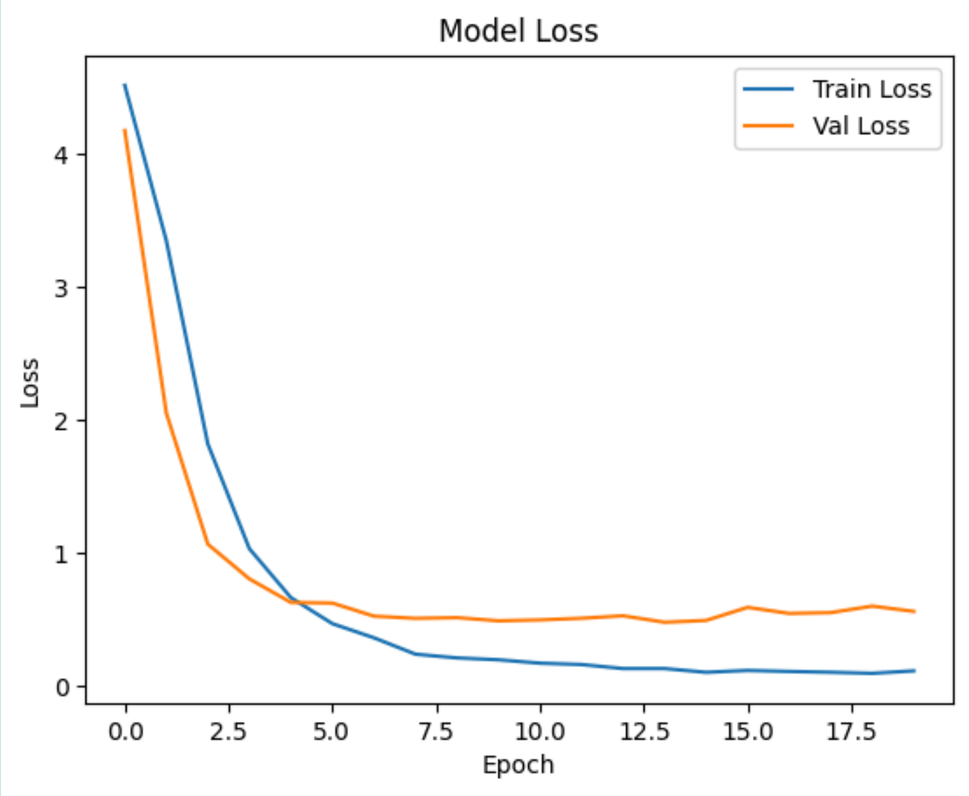
```
Validation Loss: 0.5614
Validation Accuracy: 0.9130
```

Hasil akurasi validasi model pertama adalah 91,30%

Evaluasi dan Hasil

Grafik Training





Evaluasi dan Hasil

Laporan Klasifikasi

	precision	recall	f1-score	support
001	1.00	1.00	1.00	8
002	1.00	1.00	1.00	8
003	1.00	0.88	0.93	8
004	0.88	0.88	0.88	8
005	0.88	0.88	0.88	8
006	1.00	1.00	1.00	8
007	1.00	0.38	0.55	8
008	0.88	0.88	0.88	8
009	1.00	0.88	0.93	8
010	0.88	0.88	0.88	8
011	0.88	0.88	0.88	8
012	0.67	1.00	0.80	8
013	0.88	0.88	0.88	8
014	0.73	1.00	0.84	8
015	1.00	1.00	1.00	8
016	1.00	0.62	0.77	8
017	0.89	1.00	0.94	8
018	1.00	0.88	0.93	8
019	0.80	1.00	0.89	8
020	1.00	0.88	0.93	8
021	0.73	1.00	0.84	8
022	1.00	1.00	1.00	8
...				
accuracy			0.91	736
macro avg	0.92	0.91	0.91	736
weighted avg	0.92	0.91	0.91	736

Evaluasi dan Hasil

Tes Sampling Acak

Percobaan Grayscale

Tanpa Contrast dan ROI

```
Validation Loss: 0.6464
Validation Accuracy: 0.9022
```

hanya grayscale

```
Validation Loss: 0.5614
Validation Accuracy: 0.9130
```

ditambahkan contrast
dan ROI

Mencoba membuat model dengan dataset yang hanya menggunakan gambar grayscale, tanpa penyesuaian kontras. Hasil akurasinya sekitar 90,22%

Kesimpulan:

Model CNN yang dikembangkan berhasil mencapai akurasi validasi yang sangat baik, yaitu 91.30%, dalam tugas palm recognition pada 92 kelas yang berbeda. Tahapan praproses, terutama penggunaan CLAHE untuk meningkatkan kontras, terbukti efektif dalam menonjolkan fitur-fitur penting pada gambar.

Arsitektur CNN yang relatif sederhana mampu mempelajari representasi fitur yang diskriminatif dari data. Hasil ini menunjukkan bahwa pendekatan deep learning dengan CNN dapat digunakan untuk aplikasi biometrik berbasis telapak tangan.

Thank you very much!