PRAKTIKUM

MMAI1004 – PEMBELAJARAN MESIN

PERTEMUAN II

DOSEN PENGASUH:

Prof. Dr. Taufik Fuadi Abidin, S.Si., M.Tech

Dr. Rumaisa Kruba, S.Si

Prof. Dr. Ir. Hammam Riza, M.Sc. IPU

Praktisi Komdigi (Dr. Said Mirza Pahlevi)

ASISTEN PRAKTIKUM:

Muhammad Chaidir, S.Kom, M.Kom

Hands on Face Verification using DeepFace

1. Introduction

This session will explore face verification using the Deepface library and the FaceScrub dataset. You will also use MTCNN for detecting faces and extracting the Region of Interest (RoI) before performing verification.

The objectives of this session include:

- ✓ Understanding Deepface for face verification and facial attribute analysis.
- ✓ Using MTCNN for face detection and preprocessing (done in the last meeting).
- ✓ Conducting small experiments with face verification using Deepface.

---

2. Dataset: Facescrub

FaceScrub is a real-world dataset containing 107,818 face images of 530 male and female celebrities (approximately 200 images per person). The dataset can be referenced and downloaded from:

http://vintage.winklerbros.net/facescrub.html

For this tutorial, you need to:

- ✓ Randomly select 6 images as a database and 6 images as validation for each actor.
- ✓ Use a maximum of 200 unique faces.

---

3. Selecting Images for Database and Validation

```
import os
import random


# Define dataset path
dataset_path = "facescrub_dataset_cleaned2"
database_path = "database"
validation_path = "validation"


# Ensure directories exist
os.makedirs(database_path, exist_ok=True)
os.makedirs(validation_path, exist_ok=True)


# Process each actor's directory
```

```
for actor in os.listdir(dataset_path):
    actor_dir = os.path.join(dataset_path, actor)
    if os.path.isdir(actor_dir):
        images = os.listdir(actor_dir)
        random.shuffle(images)

        # Select images
        database_images = images[:6]
        validation_images = images[6:12]

        # Copy to respective folders
        for img in database_images:
            src = os.path.join(actor_dir, img)
            dst = os.path.join(database_path, img)
            os.system(f"cp '{src}' '{dst}'")

        for img in validation_images:
            src = os.path.join(actor_dir, img)
            dst = os.path.join(validation_path, img)
            os.system(f"cp '{src}' '{dst}'")

print("Database and validation images have been successfully selected.")
```

---

4. Face Verification using Deepface

```
import os
import random
import cv2
from deepface import DeepFace
import matplotlib.pyplot as plt

# Define dataset paths
database_path = "database"
validation_path = "validation"



# Ensure directories exist
```

```
if not os.path.exists(database_path) or not
os.path.exists(validation_path):
    print("Error: Database or Validation directory not found!")
    exit()

# Select a random image from database and validation
database_images = os.listdir(database_path)
validation_images = os.listdir(validation_path)

if not database_images or not validation_images:
    print("Error: No images found in database or validation
directories!")
    exit()

img1_path = os.path.join(database_path, random.choice(database_images))
img2_path = os.path.join(validation_path,
random.choice(validation_images))

# 1. Check if file exists
if not os.path.exists(img2_path):
    print("Error: Image file not found!")
else:
    print("File exists:", img2_path)

# 2. Check if image can be read
image = cv2.imread(img2_path)
if image is None:
    print("Error: Image is corrupted or cannot be read!")
else:
    print("Image loaded successfully!")

# 3. Run DeepFace verification
try:
    metric = "cosine"  # Define similarity metric
    result = DeepFace.verify(img1_path, img2_path, model_name="VGG-Face",
distance_metric=metric)
```

```
    # Display the images and result
    fig, axes = plt.subplots(1, 2, figsize=(10, 5))
    axes[0].imshow(cv2.imread(img1_path)[:,:,::-1])
    axes[0].set_title("Database Image")
    axes[1].imshow(cv2.imread(img2_path)[:,:,::-1])
    axes[1].set_title("Validation Image")
    plt.show()

    print("DeepFace Verification Result:", result)
except Exception as e:
    print("DeepFace encountered an error:", str(e))
```

---

5. Conclusion

- ✓ We selected 6 images as databases and 6 images as validation for each actor.
- ✓ We used Deepface to verify face pairs and measure similarity.
- ✓ Understanding these concepts is essential for building robust face verification systems.