DAFTAR PUSTAKA

Al-agha, S. A., Saleh, H. H., & Ghani, R. F. (2017). Geometric-based Feature Extraction and Classification for Emotion Expressions of 3D Video Film. *Journal of Advances in Information Technology, 8(2)*, 74-79. http://dx.doi.org/10.12720/jait.8.2.74-79.

Alamsyah, D., & Pratama, D. (2020). Implementasi Convolutional Neural Networks (Cnn) Untuk Klasifikasi Ekspresi Citra Wajah Pada Fer-2013 Dataset. *Jurnal Teknologi Informasi*, 350-355.

Amda, K. & Fitriyani, R. (2016). *Membaca Ekspresi Wajah.* Depok: Huta Publisher.

Amda, K., & Fitriyani, R. (2018). *Membaca Ekspresi Wajah.* Yogyakarta: Huta Publisher.

Antonius, D. (2019). *Gesture The Secret of Body Language and Facial Expression.* Komunitas Psikologi Digital.

Ashwin, D. V., Kimar, A., & Manikandan, J. (2018). Design of a Real-Time Human Emotion Recognition System. *Ubiquitous Communications and Network Computing* (pp. 177-178). India: Springer, Cham. https://doi.org/10.1007/978-3-319-73423-1_16.

Astuti, D. L. Z., & Samsuryadi. (2018). Kajian Pengenalan Ekspresi Wajah menggunakan Metode PCA dan CNN . *Annual Research Seminar. 4(1)*, 294-297.

Cahyo, D.N., Zahro, H. Z., & Vendyansyah, N. (2023). Pengenalan Ekspresi Mikro Wajah Dengan Ekstraksi Fitur pada Komponen Wajah Menggunakan Metode Local Binary Pattern Histogram. *Jurnal Mahasiswa Teknik Informatika*, 822-829.

Chen, Q., & Sang, L. (2018). Face-Mask Recognition for Fraud Prevention Using Gaussian Mixture Model. *J. Vis. Commun. Image R*, https://doi.org/10.1016/j.jvcir.2018.08.016.

Chollet, F. (2021). *Deep Learning with Python, Second Edition.* Shelter Island: Manning Publications.

Florestiyanto, M. Y., Yuwono, B., & Prasetya, D. B. (2022). *Dasar Pengolahan Citra Digital Edisi 2022.* Yogyakarta: Lembaga Penelitian dan Pengabdian kepada Masyarakat UPN Veteran Yogyakarta .

Ghimire, D., & Lee, J. (2013). Geometric Feature-Based Facial Expression Recognition in Image Sequences Using Multi-Class AdaBoost and Support Vector Machines. *Sensors, 13(6)*, 7714-7734. DOI:10.3390/s130607714.

Gonzalez, R. C., & Woods, R. E. (2018). *Digital Image Processing (4th ed.).* United States: Pearson.

Grattarola, D., & Alippi, C. (2021). Graph Neural Networks in TensorFlow and Keras with Spektral [Application Notes]. *IEEE Computational Intelligence Magazine, 16(1)*, 99-106. https://doi.org/10.1109/MCI.2020.3039072.

Gunawan, R. J., Irawan, B., & Setianingsih, C. (2021). Pengenalan Ekspresi Wajah Berbasis Convolutional Neural Network dengan Model Aristektur VGG16. *e-Proceeding of Engineering, 8(5)*, 6442-6454.

Guntoro, AL. S., Julianto, E. J., & Budiyanto, D. (2022). Pengenalan Ekspresi Wajah Menggunakan Convolutional Neural Network. *Jurnal Informatika Atma Jogja*, 155-160.

Heryadi, Y., & Irwansyah, E. (2020). *Deep Learning: Aplikasinya di Bidang Geospasial.* Jawa Barat: PT. Aritifisia Wahana Informa Teknologi.

Irawan, J. D., Handoko, F., & Adriantatri, E. (2019). Ruang Kuliah Pintar Pemantau Tingkat Efektivitas Pembelajaran yang dapat Mendeteksi Mahasiswa Bosan dan Mengantuk. *Seminar Nasional Inovasi dan Aplikasi Teknologi di Industri*, 250-256.

Kazemi, V., & Sullivan, J. (2014). One Millisecond Face Alignment with an Ensemble of Regression Trees. *IEEE Xplore*.

Kotu, V., & Deshpande, B. (2015). *Predictive Analytics and Data Mining Concepts and Practice with RapidMiner.* Waltham, USA: Morgan Kaufmann.

Kusdiananggalih, P. P., Rachmawati, E., & Risnandar. (2021, 8(2)). Pengenalan Ekspresi Wajah Dari Cross Dataset Menggunakan Convolutional Neural Network (CNN). *Jurnal Tugas Akhir Fakultas Informatika*, 3429-3445.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning. *Nature, 521*, pages436–444. http://dx.doi.org/10.1038/nature14539.

Loffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network. *Proceedings of the 32nd International Conference on Machine Learning (ICML'15)* (pp. 448–456). Lille: JMLR.org.

Manalu, R. E. (2021). Analisis Metode Histogram Equalization Dalam Proses Perbaikan Gambar Closed Circuit Television (CCTV). *TIN: Terapan Informatika Nusantara*, 1-5.

Muttaqiin, A. K., Yuana, H., & Chulkamdi, M. T. (2023). Implementasi Algoritma Convolutional Neural Network Untuk Pengenalan Ekspresi Wajah. *Jurnal Riset Sistem Informasi Dan Teknik Informatika*, 772-792.

Nasution, M. Z. (2019). Penerapan Principal Component Analysis (PCA) dalam Penentuan Faktor Dominan yang Mempengaruhi Prestasi

Belajar Siswa (Studi Kasus : SMK Raksana 2 Medan). *Jurnal Teknologi Informasi, 3(1)*, 41-48.

Nugroho, P. A., Fenriana, I., & Arijanto, R. (2020). Implementasi Deep Learning Menggunakan Convolutional Neural Network (CNN) Pada Ekspresi Manusia. *Jurnal Algor*, 12-21.

Pang, B., Nijkamp, E., & Wu, Y. N. (2019). Deep Learning with TensorFlow: A Review. *Journal of Educational and Behavioral Statistics, 10(10)*, 1-22. https://doi.org/10.3102/1076998619872761.

Putra, D. (2010). *Pengolahan Citra Digital.* Yogyakarta: Andi.

Rahman, T., Khandakar, A., Qiblawey, Y., Tahir, A., Kiranyaz, S., Kashem, S. B. A., Islam, M. T., Maadeed, S. A., Zughaier, S. M., Khan, M. S., & Chowdhury, M. E. H. (2021). Exploring the effect of image enhancement techniques on COVID-19 detection using chest X-ray images . *Computers in Biology and Medicine, 132*, https://doi.org/10.1016/j.compbiomed.2021.104319.

Rosiani, U. D., Asmara, R. S., & Laeily, N. (2019, 6(1)). Penerapan Facial Landmark Point untuk Klasifikasi Jenis Kelamin Berdasarkan Citra Wajah. *JIP (Jurnal Informatika Polinema)*, 55-60. http://dx.doi.org/10.33795/jip.v6i1.328.

Saputri, A. P., Taqwa, A., & Soim, S. (2022). Analisis Deteksi Objek Citra Digital Menggunakan Algoritma Yolo Dan Cnn Dengan Arsitektur Repvgg Pada Sistem Pendeteksian dan Pengenalan Ekspresi Wajah. *Jurnal Ilmiah Indonesia*, 13068-13080.

Seandrio, A. L., Pratomo, A. H., & Florestiyanto, M. Y. . (2021). Implementation of Convolutional Neural Network (CNN) in Facial Expression Recognition. *Telematika: Jurnal Informatika dan Teknologi Informasi*, 211-221.

Setiawan, D., Widodo, S., Ridwan, T., & Ambari, R. (2022). Perancangan Deteksi Emosi Manusia berdasarkan Ekspresi Wajah Menggunakan Algoritma VGG16. *Syntax: Jurnal Informatika, 11(01)*, 1-11. https://doi.org/10.35706/syji.v11i01.6594.

Tinaliah & Elizabeth, T. (2021). Penerapan Convolutional Neural Network untuk Klasifikasi Citra Ekspresi Wajah Manusia Pada MMA Facial Expression Dataset. *Jurnal Teknik Informatika dan Sistem Informasi*, 2051-2059 .

Yudistira, R., Meha, A. I., & Prasetyo, S. Y. J. (2019). Perubahan Konversi Lahan Menggunakan NDVI, EVI, SAVI dan PCA pada Citra Landsat 8 (Studi Kasus : Kota Salatiga). *Indonesian Journal of Computing and Modeling, 2(1)*, 25–30.

Zein, A. (2018). Pendeteksian Kantuk Secara Real Time Menggunakan Pustaka Opencv dan Ddlib Python. *Sainstech, 28(2)*, 22-26. https://doi.org/10.37277/stch.v28i2.238.

LAMPIRAN

Lampiran 1. Contoh Dataset

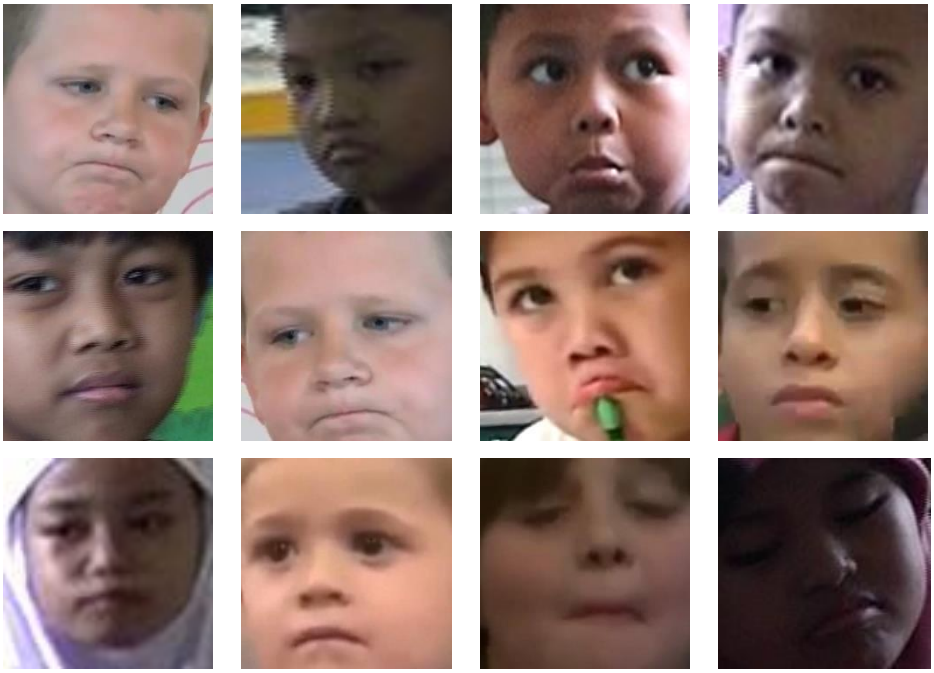
a. Kelas Bahagia

b. Kelas Lelah

c. Kelas Marah

d. Kelas Netral

e. Kelas Sedih



f. Kelas Terkejut

Lampiran 2. Lampiran Source Code Program

a. Deteksi Wajah

```python
import cv2
import dlib
import os
import time

class InvalidImage(Exception):
    pass

# Path to video file
video_path = r'C:\Users\PC\Documents\amiqatun\youtube\Today Classroom - Elko Middle School.mp4'
# Path to dlib's shape predictor model
shape_predictor_path = r'C:\Users\PC\Documents\amiqatun\Master\shape_predictor_68_face_landmarks.dat'
# Output folder for cropped faces
output_folder = r'C:\Users\PC\Documents\amiqatun\youtube\Today Classroom - Elko Middle School'

# Initialize video capture
cam = cv2.VideoCapture(video_path)

# Initialize face detector and landmark predictor
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(shape_predictor_path)

# Ensure output folder exists
if not os.path.exists(output_folder):
    os.makedirs(output_folder)

# Get video frame rate
fps = cam.get(cv2.CAP_PROP_FPS)

# Define how many frames to process per second
frames_per_second = 10
frame_interval = int(fps / frames_per_second)

# Initialize frame counter
frame_count = 0

# Target size for cropped face images
target_size = (512, 512)

# Iterate over frames in the video
while True:
    ret, frame = cam.read()
    if not ret:
        break  # End of video

    try:
        # Process only every frame_interval-th frame
        if frame_count % frame_interval == 0:
            # Convert frame to grayscale
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

            # Detect faces
            faces = detector(gray)

            # Check if any faces are detected
            if faces:
                for idx, face in enumerate(faces):
                    # Get facial landmarks
                    landmarks = predictor(gray, face)

                    # Display landmarks on face (optional)
                    for n in range(68):
                        if n not in range(0, 17):  # Skip jawline points
                            x = landmarks.part(n).x
                            y = landmarks.part(n).y

                    # Crop face from frame
                    x, y, w, h = face.left(), face.top(), face.width(), face.height()

                    # Ensure the crop region is within the frame bounds
                    if x < 0 or y < 0 or x + w > frame.shape[1] or y + h > frame.shape[0]:
                        print(f"Skipping invalid crop region: ({x}, {y}, {w}, {h})")
                        continue

                    face_roi = frame[y:y + h, x:x + w]

                    # Get number of pixels in cropped face
                    jumlah_pixel = face_roi.shape[0] * face_roi.shape[1]
                    print(f"Jumlah pixel hasil crop: {jumlah_pixel}")

                    # Resize cropped face to target size
                    if jumlah_pixel > 0:
                        resized_face = cv2.resize(face_roi, target_size)
                        output_path = os.path.join(output_folder, f'face_{idx}_time_{int(time.time())}.jpg')
                        cv2.imwrite(output_path, resized_face)
                    else:
                        print("Skipping empty cropped image")

        # Increment frame counter
        frame_count += 1

        # Display result (optional)
        cv2.imshow('Deteksi dan Crop', frame)

    except InvalidImage as e:
        print(f"Error: {e}")
        break
    except Exception as e:
        print(f"Error: {e}")
        break

    # Stop program if 'q' key is pressed
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release video capture and close display window
cam.release()
cv2.destroyAllWindows()
```

b. Augmentasi Data

```python
import Augmentor
import os

# Path to the main directory containing the 6 subfolders
main_dir = r"C:\Users\PC\Documents\amiqatun\data mix dua"  # Update this path as needed
output_dir = r"C:\Users\PC\Documents\amiqatun\data mix dua\augmentasi_dua"  # Update this path as needed

# Function to set up and run the augmentation pipeline for each subfolder
def augment_images_in_subfolder(subfolder_path, output_subfolder_path, num_samples=100):
    # Create an output directory for the augmented images
    os.makedirs(output_subfolder_path, exist_ok=True)

    # Ensure there are images in the subfolder
    image_files = [f for f in os.listdir(subfolder_path) if f.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp', '.gif'))]
    if not image_files:
        print(f"No images found in {subfolder_path}. Skipping augmentation.")
        return

    # Set up the pipeline
    p = Augmentor.Pipeline(source_directory=subfolder_path, output_directory=output_subfolder_path)

    # Define augmentation operations
    p.flip_left_right(probability=0.9)
    p.rotate(probability=0.9, max_left_rotation=5, max_right_rotation=5)
    p.zoom_random(probability=0.9, percentage_area=0.9)
    p.random_brightness(probability=0.9, min_factor=0.7, max_factor=1.3)

    # Run the augmentation pipeline
    p.sample(num_samples)

    print(f"Augmented images have been saved in: {output_subfolder_path}")

# Iterate over each subfolder in the main directory and run the augmentation pipeline
for subfolder in os.listdir(main_dir):
    subfolder_path = os.path.join(main_dir, subfolder)
    output_subfolder_path = os.path.join(output_dir, subfolder)
    if os.path.isdir(subfolder_path):
        print(f"Processing subfolder: {subfolder_path}")
        augment_images_in_subfolder(subfolder_path, output_subfolder_path, num_samples=500)  # Adjust num_samples as needed
```

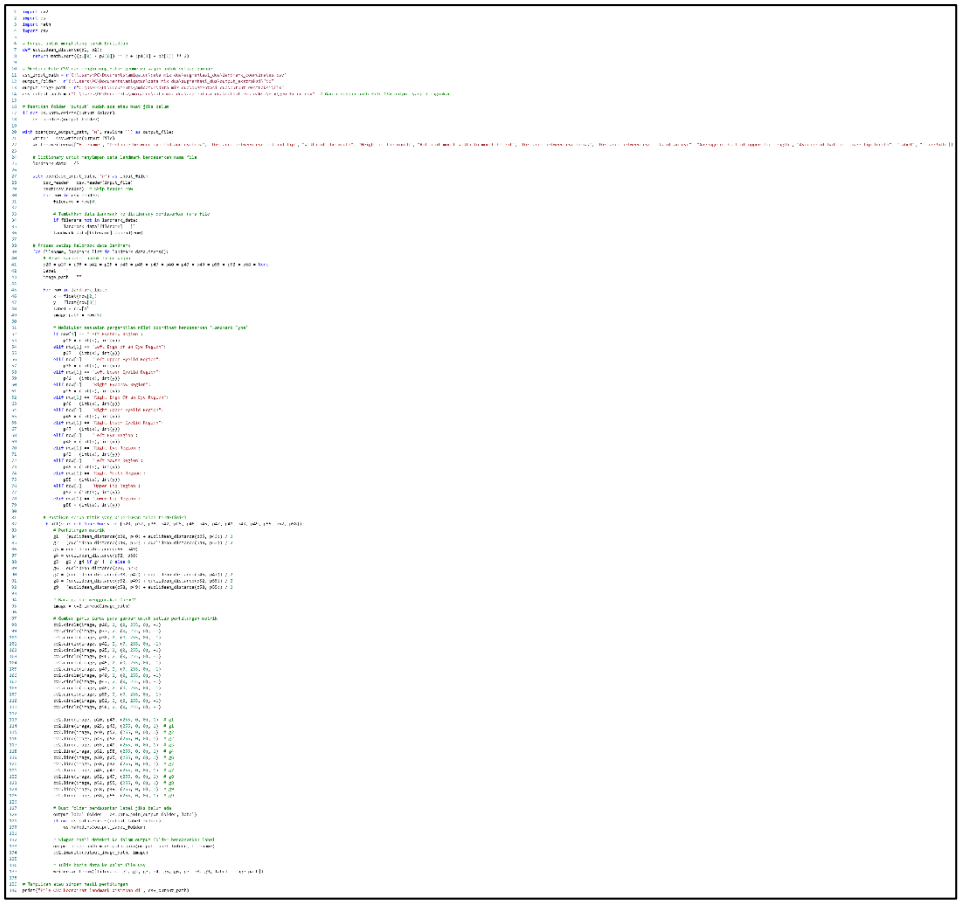
c. *Grayscale*, *Histogram Equalization*, *Resizing*

```python
import os
import cv2

# Define the input and output folders
folder_path = r"C:\Users\PC\Documents\amiqatun\data primer roboflow"
output_folder = r"C:\Users\PC\Documents\amiqatun\data primer roboflow\output_resized"

# Create the output folder if it doesn't exist
if not os.path.exists(output_folder):
    os.makedirs(output_folder)

# Define the target size for resizing
target_size = 96

# Iterate through each expression folder inside the main folder
for expression_folder in os.listdir(folder_path):
    expression_folder_path = os.path.join(folder_path, expression_folder)

    # Ensure it's a directory
    if os.path.isdir(expression_folder_path) and expression_folder != "output_resized":
        # Create a corresponding folder inside the output folder
        output_expression_folder = os.path.join(output_folder, expression_folder)
        if not os.path.exists(output_expression_folder):
            os.makedirs(output_expression_folder)

        # Iterate through each image file in the expression folder
        for filename in os.listdir(expression_folder_path):
            if filename.endswith((".jpg", ".jpeg", ".png")):
                image_path = os.path.join(expression_folder_path, filename)

                # Read the image and perform preprocessing
                image = cv2.imread(image_path)
                grayscale_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
                equalized_grayscale_image = cv2.equalizeHist(grayscale_image)
                resized_image = cv2.resize(equalized_grayscale_image, (target_size, target_size))

                # Save the processed image in the corresponding output folder
                output_file = os.path.join(output_expression_folder, filename)
                cv2.imwrite(output_file, resized_image)

print("Program selesai. Hasil dapat ditemukan di", output_folder)
```

d. Deteksi *Landmark* Wajah

```python
import cv2
import dlib
import os
import csv

# Inisialisasi detector wajah menggunakan dlib
face_detector = dlib.get_frontal_face_detector()
landmark_predictor = dlib.shape_predictor(r'C:\Users\PC\Documents\amiqatun\Master\shape_predictor_68_face_landmarks.dat')

# # Path folder input yang berisi banyak gambar
input_folder = r"C:\Users\PC\Documents\amiqatun\data mix dua\augmentasi_dua\output_resized"

# Output folder untuk menyimpan hasil crop
output_folder = r"C:\Users\PC\Documents\amiqatun\data mix dua\augmentasi_dua\output_ekstraksi"

# Output file CSV untuk menyimpan koordinat landmark
csv_output_path = r"C:\Users\PC\Documents\amiqatun\data mix dua\augmentasi_dua\landmark_coordinates.csv"

# Pastikan folder 'output' sudah ada atau buat jika belum
if not os.path.exists(output_folder):
    os.makedirs(output_folder)

# Counter untuk menghitung jumlah foto yang telah selesai dideteksi
detected_count = 0

# Membuka file CSV untuk menulis koordinat landmark
with open(csv_output_path, 'w', newline='') as csvfile:
    csv_writer = csv.writer(csvfile)
    # Menulis header CSV
    csv_writer.writerow(["Filename", "Landmark Type", "X", "Y", "Label", "Image Path"])

    # Coba membaca setiap gambar dalam folder
    for expression_folder in os.listdir(input_folder):
        expression_folder_path = os.path.join(input_folder, expression_folder)

        # Pastikan itu adalah folder
        if os.path.isdir(expression_folder_path):
            # Buat folder di dalam output_folder
            output_expression_folder = os.path.join(output_folder, expression_folder)
            if not os.path.exists(output_expression_folder):
                os.makedirs(output_expression_folder)

            # Iterasi melalui setiap file gambar di dalam folder ekspresi
            for filename in os.listdir(expression_folder_path):
                if filename.endswith((".jpg", ".jpeg", ".png")):
                    image_path = os.path.join(expression_folder_path, filename)

                    try:
                        # Baca gambar menggunakan OpenCV
                        image = cv2.imread(image_path)

                        # Deteksi wajah menggunakan dlib
                        faces = face_detector(image, 1)  # 1: Upsample the image for more accurate face detection

                        # Iterasi untuk setiap wajah yang terdeteksi
                        for face in faces:
                            landmarks = landmark_predictor(image, face)

                            # Koordinat untuk perhitungan metrik
                            p37 = (landmarks.part(36).x, landmarks.part(36).y)
                            p38 = (landmarks.part(37).x, landmarks.part(37).y)
                            # p39 = (landmarks.part(38).x, landmarks.part(38).y)
                            # p41 = (landmarks.part(40).x, landmarks.part(40).y)
                            p42 = (landmarks.part(41).x, landmarks.part(41).y)
                            # p44 = (landmarks.part(43).x, landmarks.part(43).y)
                            p45 = (landmarks.part(44).x, landmarks.part(44).y)
                            p46 = (landmarks.part(45).x, landmarks.part(45).y)
                            p47 = (landmarks.part(46).x, landmarks.part(46).y)
                            # p48 = (landmarks.part(47).x, landmarks.part(47).y)

                            p20 = (landmarks.part(19).x, landmarks.part(19).y)
                            p25 = (landmarks.part(24).x, landmarks.part(24).y)
                            p40 = (landmarks.part(39).x, landmarks.part(39).y)
                            p43 = (landmarks.part(42).x, landmarks.part(42).y)
                            p55 = (landmarks.part(54).x, landmarks.part(54).y)
                            p49 = (landmarks.part(48).x, landmarks.part(48).y)
                            p52 = (landmarks.part(51).x, landmarks.part(51).y)
                            p58 = (landmarks.part(57).x, landmarks.part(57).y)

                            # Gambar circle untuk setiap landmark dan tulis ke file CSV
                            for i, (landmark_type, landmark_point) in enumerate([
                                ("Left Eyebrow", p20),
                                ("Left Edge of an Eye", p37),
                                ("Left Upper Eyelid", p38),
                                ("Left Lower Eyelid", p42),
                                ("Left Eye", p40),
                                ("Left Mouth", p49),
                                ("Right Eyebrow", p25),
                                ("Right Edge Of an Eye", p46),
                                ("Right Upper Eyelid", p45),
                                ("Right Lower Eyelid", p47),
                                ("Right Eye", p43),
                                ("Right Mouth", p55),
                                ("Upper Lip", p52),
                                ("Lower Lip", p58),
                            ]):
                                x, y = landmark_point
                                cv2.circle(image, (x, y), 1, (0, 255, 0), -1)
                                csv_writer.writerow([filename, f"{landmark_type} Region", x, y, expression_folder, image_path])

                            # Simpan hasil deteksi ke dalam output_folder
                            output_path = os.path.join(output_expression_folder, f'{filename}')
                            cv2.imwrite(output_path, image)

                            # Increment counter
                            detected_count += 1

                    except Exception as e:
                        print(f"Error pada {filename}: {e}")

print("Program selesai. Hasil dapat ditemukan di", output_folder)
print(f"Jumlah foto yang telah selesai dideteksi: {detected_count}")
print("File CSV koordinat landmark disimpan di", csv_output_path)
```

e. Ekstraksi Fitur Geometri Wajah



f. *Training* dan Evaluasi Model

```python
import pandas as pd
import seaborn as sns
import cv2
import os
import matplotlib.pyplot as plt
import cv2
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from sklearn.metrics import confusion_matrix, classification_report
from keras.utils import to_categorical
from keras.models import Model
from keras.layers import Conv2D, MaxPooling2D, Dense, Input, concatenate, Dropout, BatchNormalization, GlobalAveragePooling2D, Flatten, Concatenate, Activation, Add
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint
from sklearn.utils.class_weight import compute_class_weight
from keras.callbacks import *
import random
import time
from sklearn.decomposition import PCA
```

```python
# Load data
data = df

# Count the occurrences of each unique label
label_counts = data["Label"].value_counts()

# Print the count of unique labels with their label names
print("Count of unique labels:")
for label, count in label_counts.items():
    print(f"{label}: {count}")
```

```python
def normalize_image(image):
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    resized_image = cv2.resize(gray_image, (96, 96))
    normalized_image = resized_image / 255.0
    return normalized_image

# Load image data
image_paths = data["ImagePath"]
images = [cv2.imread(image_path) for image_path in image_paths]

# Normalisasi geo feature
scaler = MinMaxScaler()
X_geometry = scaler.fit_transform(data.drop(columns=["Filename", "Label", "ImagePath"]))

# Normalisasi image data
X_images_normalized = np.array([normalize_image(image) for image in images])

# print before PCA
print("Shape before PCA:", X_geometry.shape)

# Apply PCA
pca = PCA(n_components=2)
X_geometry_pca = pca.fit_transform(X_geometry)

# after PCA
print("Shape after PCA:", X_geometry_pca.shape)

X_images_normalized = X_images_normalized[..., np.newaxis]

# convert label to numerical using LabelEncoder
label_encoder = LabelEncoder()
data["Label"] = label_encoder.fit_transform(data["Label"])

# Encoding label
y = to_categorical(data["Label"])

# Split data into training, validation, and test sets
X_train_geo, X_temp_geo, y_train_geo, y_temp_geo = train_test_split(X_geometry_pca, y, test_size=0.2, random_state=42, stratify=data["Label"])
X_train_img, X_temp_img, y_train_img, y_temp_img = train_test_split(X_images_normalized, y, test_size=0.2, random_state=42, stratify=data["Label"])

print("\nTotal data")
print(f"Training set size (geo): {len(X_train_geo)}")
print(f"Training set size (img): {len(X_train_img)}")

# Further split the temporary set into validation and test sets (50% of the remaining 20%)
X_val_geo, X_test_geo, y_val_geo, y_test_geo = train_test_split(X_temp_geo, y_temp_geo, test_size=0.5, random_state=42, stratify=y_temp_geo)
X_val_img, X_test_img, y_val_img, y_test_img = train_test_split(X_temp_img, y_temp_img, test_size=0.5, random_state=42, stratify=y_temp_geo)

print(f"Validation set size (geo): {len(X_val_geo)}")
print(f"Test set size (geo): {len(X_test_geo)}")
print(f"Validation set size (img): {len(X_val_img)}")
print(f"Test set size (img): {len(X_test_img)}")

# Ensure all data splits have the same number of samples
assert len(X_train_geo) == len(X_train_img) == len(y_train_geo) == len(y_train_img)
assert len(X_val_geo) == len(X_val_img) == len(y_val_geo) == len(y_val_img)
assert len(X_test_geo) == len(X_test_img) == len(y_test_geo) == len(y_test_img)

# Convert lists of images to NumPy arrays (already done above for X_images_normalized)
X_train_img_array = np.array(X_train_img)
X_val_img_array = np.array(X_val_img)
X_test_img_array = np.array(X_test_img)

# Ensure dimensions of inputs match the model structure
assert X_train_img_array.shape[0] == X_train_geo.shape[0]
assert X_val_img_array.shape[0] == X_val_geo.shape[0]
assert X_test_img_array.shape[0] == X_test_geo.shape[0]

print("\nTraining set shape (geo):", X_train_geo.shape)
print("Validation set shape (geo):", X_val_geo.shape)
print("Test set shape (geo):", X_test_geo.shape)

print("Training set shape (img):", X_train_img_array.shape)
print("Validation set shape (img):", X_val_img_array.shape)
print("Test set shape (img):", X_test_img_array.shape)
```

```
image_height = 96
image_width = 96
num_classes = len(np.unique(data["Label"]))
num_features = X_geometry_pca.shape[1]

# Step 1: Build the CNN model for face image data
input_image = Input(shape=(image_height, image_width, 1))
model_cnn = Conv2D(16, kernel_size=(3, 3), activation='relu', padding='same')(input_image)
model_cnn = BatchNormalization()(model_cnn)
model_cnn = MaxPooling2D(pool_size=(2, 2))(model_cnn)

model_cnn = Conv2D(32, kernel_size=(3, 3), activation='relu', padding='same')(model_cnn)
model_cnn = BatchNormalization()(model_cnn)
model_cnn = MaxPooling2D(pool_size=(2, 2))(model_cnn)

model_cnn = Conv2D(64, kernel_size=(3, 3), activation='relu', padding='same')(model_cnn)
model_cnn = BatchNormalization()(model_cnn)
model_cnn = MaxPooling2D(pool_size=(2, 2))(model_cnn)

model_cnn = Conv2D(128, kernel_size=(3, 3), activation='relu', padding='same')(model_cnn)
model_cnn = BatchNormalization()(model_cnn)
model_cnn = MaxPooling2D(pool_size=(2, 2))(model_cnn)

model_cnn = Conv2D(256, kernel_size=(3, 3), activation='relu', padding='same')(model_cnn)
model_cnn = BatchNormalization()(model_cnn)
model_cnn = MaxPooling2D(pool_size=(2, 2))(model_cnn)

model_cnn = Conv2D(256, kernel_size=(3, 3), activation='relu', padding='same')(model_cnn)
model_cnn = BatchNormalization()(model_cnn)
model_cnn = MaxPooling2D(pool_size=(2, 2))(model_cnn)

model_cnn = Flatten()(model_cnn)

# Step 2: Build the model for geometric feature data
input_geometry = Input(shape=(num_features,))
model_geometry = Dense(256, activation='relu')(input_geometry)
model_geometry = Dense(64, activation='relu')(model_geometry)
model_geometry = Dense(256, activation='relu')(model_geometry)

# Step 3: Combine both models using Keras Functional API
combined_input = concatenate([model_cnn, model_geometry])
combined_output = Dense(256, activation='relu')(combined_input)
combined_output = Dense(64, activation='relu')(combined_input)
combined_output = Dense(num_classes, activation='softmax')(combined_output)
model_combined = Model(inputs=[input_image, input_geometry], outputs=combined_output)

# Step 4: Compile and train the combined model using training data
model_combined.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])

model_combined.summary()
```

```
checkpoint = ModelCheckpoint('modelR2.keras',
                             monitor='val_accuracy',
                             save_best_only=True,
                             verbose=1,
                             mode='auto')
```

```
# Start timing before training starts
start_time = time.time()

history = model_combined.fit([X_train_img_array, X_train_geo], y_train_img,
                             batch_size=12, epochs=50,
                             validation_data=([X_val_img_array, X_val_geo], y_val_img),
                             callbacks=[checkpoint]
                             )

# Calculate the elapsed time
elapsed_time = time.time() - start_time

# Calculate the elapsed time in minutes and seconds
elapsed_time_minutes = int(elapsed_time // 60)
elapsed_time_seconds = int(elapsed_time % 60)

# Print the formatted elapsed time
print(f"Training time: {elapsed_time_minutes} minutes and {elapsed_time_seconds} seconds")

# # Step 5: Evaluasi model gabungan menggunakan data pengujian
loss, accuracy = model_combined.evaluate([np.array(X_test_img_array), X_test_geo], y_test_img)
```

```python
# evualiation model
print(f"Test Loss: {loss:.4f}")
print(f"Test Accuracy: {accuracy:.4f}")

# get validation loss dan validation acc from history
final_val_loss = history.history['val_loss'][-1]
final_val_accuracy = history.history['val_accuracy'][-1]

print(f"Validation Loss: {final_val_loss:.4f}")
print(f"Validation Accuracy: {final_val_accuracy:.4f}")
```

```python
# Plot training and validation accuracy
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy (Batch Size=12, lr=0,001)')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```python
# Plot training and validation loss
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss (Batch Size=12, lr=0,001)')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```python
# acc & loss from history
val_accuracy = history.history['val_accuracy']
val_loss = history.history['val_loss']

# Plot
epochs = range(1, len(val_accuracy) + 1)

plt.plot(epochs, val_accuracy, 'b', label='Validation Accuracy')
plt.plot(epochs, val_loss, 'r', label='Validation Loss')
plt.title('Validation Accuracy and Loss (Batch Size=12, lr=0,001)')
plt.xlabel('Epochs')
plt.ylabel('Accuracy / Loss')
plt.legend()

plt.show()
```

```python
# Confusion Matrix and Classification Report
y_pred_probs = model_combined.predict([X_test_img_array, X_test_geo])
y_pred = np.argmax(y_pred_probs, axis=1)
y_true = np.argmax(y_test_img, axis=1)

cm = confusion_matrix(y_true, y_pred)
class_labels = label_encoder.classes_

plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_labels, yticklabels=class_labels)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

cr = classification_report(y_true, y_pred, target_names=class_labels, labels=np.arange(len(class_labels)))
print("Classification Report:\n", cr)
```

```python
# Randomly select an index for the test image
random_idx = random.randint(0, len(X_test_img_array) - 1)

# Extract the random test image and corresponding geo data
test_img = X_test_img_array[random_idx]
test_geo = X_test_geo[random_idx]

# Make a prediction
y_pred_prob_single = model_combined.predict([np.expand_dims(test_img, axis=0), np.expand_dims(test_geo, axis=0)])
y_pred_single = np.argmax(y_pred_prob_single, axis=1)[0]

# Get the true label for the random test image
y_true_single = y_true[random_idx]

# Plot the image
plt.imshow(test_img, cmap='gray')
plt.title(f"True Label: {class_labels[y_true_single]} \n Predicted Label: {class_labels[y_pred_single]}")
plt.axis('off')
plt.show()
```

```python
# Number of test samples
num_test_samples = len(X_test_img_array)

# Set up the plot grid
num_cols = 5
num_rows = (num_test_samples + num_cols - 1) // num_cols
fig, axes = plt.subplots(num_rows, num_cols, figsize=(15, num_rows * 3))

for idx in range(num_test_samples):
    # Extract the test image and corresponding geo data
    test_img = X_test_img_array[idx]
    test_geo = X_test_geo[idx]

    # Make a prediction
    y_pred_prob_single = model_combined.predict([np.expand_dims(test_img, axis=0), np.expand_dims(test_geo, axis=0)])
    y_pred_single = np.argmax(y_pred_prob_single, axis=1)[0]

    # Get the true label for the test image
    y_true_single = y_true[idx]

    # Plot the image
    ax = axes[idx // num_cols, idx % num_cols]
    ax.imshow(test_img, cmap='gray')
    ax.set_title(f"True: {class_labels[y_true_single]}\nPred: {class_labels[y_pred_single]}")
    ax.axis('off')

# Hide any empty subplots
for j in range(idx + 1, num_rows * num_cols):
    axes[j // num_cols, j % num_cols].axis('off')

plt.tight_layout()
plt.show()
```

Lampiran 3. Daftar hadir dan berita acara seminar hasil

KEMENTERIAN PENDIDIKAN , KEBUDAYAAN
RISET DAN TEKNOLOGI
UNIVERSITAS HASANUDDIN
FAKULTAS TEKNIK
DEPARTEMEN TEKNIK INFORMATIKA
Kampus FakultasTeknikUnhas, Jl. Poros Malino, Gowa
http://eng.unhas.ac.id/informatika, Email : informatika@unhas.ac.id

DAFTAR HADIR SEMINAR HASIL

Nama/Stambuk : Amiqatun Nasyati Yusri D121201080

Judul Skipsi/T.A : " **Klasifikasi Ekspresi Wajah Peserta Didik dalam Proses Pembelajaran di Kelas**"

Hari/Tanggal : Jumat , 21 Juni 2024
Jam : 09.00 Wita – Selesai
Tempat : Ruang Lab. UBICON Departemen Teknik Informatika Gowa

No.	Jabatan	Nama Dosen	Tanda Tangan
	Pembimbing	1. Dr. Amil Ahmad Ilham,ST.,M.IT	1.....
II.	Anggota Penguji	3.Dr.Ir. Ingrid Nurtanio,M.T	2 .....
		4.Prof. Dr. Eng.Intan Sari Areni,ST.,M.T	3 .....

PANITIA UJIAN

Ketua,

Dr. Amil Ahmad Ilham,ST.,M.IT

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET, DAN TEKNOLOGI
UNIVERSITAS HASANUDDIN
FAKULTAS TEKNIK
DEPARTEMEN TEKNIK INFORMATIKA
Kampus Fakultas Teknik Unhas, Jl. Poros Malino, Gowa
http://eng.unhas.ac.id/informatika, Email : informatika@unhas.ac.id

Nomor : 802/UN4.7.7/TD.06/2024
Lamp : -
Hal : Penerbitan Surat Penugasan Panitia/Penguji Seminar Hasil Strata Satu (S1)

Kepada Yth :

Wakil Dekan Bidang Akademik dan Kemahasiswaan
Fakultas Teknik Universitas Hasanuddin

Di-

Gowa

Dengan hormat,
Berdasarkan Persetujuan Pembimbing Mahasiswa, Bersama ini diusulkan susunan Panitia/Penguji Seminar Hasil Strata Satu (S1) bagi mahasiswa Departemen Teknik Informatika Fakultas Teknik tersebut di bawah ini :

Nama / Stambuk : Amiqatun Nasyati Yusri D121 20 1080
Judul TA : Klasifikasi Ekspresi Wajah Peserta Didik dalam Proses Pembelajaran di Kelas

Dengan ini kami sampaikan Susunan Panitia Seminar Hasil Program Strata Satu (S1) Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin dengan susunan sebagai berikut :

Pembimbing I/ Ketua : 1. Dr. Amil Ahmad Ilham, ST., M.IT
Anggota : 2. Dr. Ir. Ingrid Nurtanio, M.T
: 3. Prof. Dr.Eng. Intan Sari Areni, ST., M.T.

Untuk dapat diterbitkan surat penugasannya

Demikian penyampaian kami, atas perhatian dan kerjasamanya diucapkan terima kasih.

Gowa, 19 Juni 2024
Ketua Departemen Tek.Informatika,

Prof. Dr. Ir. Indrabayu, ST, MT., M.Bus.Sys., IPM, ASEAN.Eng
Nip.19750716 200212 1 004

Tembusan :
1. Arsip

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET, DAN TEKNOLOGI
UNIVERSITAS HASANUDDIN
FAKULTAS TEKNIK
Jalan Poros Malino Km. 6 Bontomarannu, Gowa, 92171, Sulawesi Selatan
☎ +62811 4420 909, E-mail: teknik@unhas.ac.id , https://eng.unhas.ac.id

SURAT PENUGASAN
No. 14134/UN4.7.1/TD.06/2024

Dari : Dekan Fakultas Teknik Universitas Hasanuddin

Kepada : Mereka yang tercantum namanya dibawah ini

Isi :

1. Bahwa merujuk kepada Peraturan Rektor Universitas Hasanuddin **Nomor : 29/UN4.1/2023 tentang Penyelenggaraan Program Sarjana Universitas Hasanuddin** , dengan ini menugaskan Saudara sebagai **PENGUJI/PANITIA SEMINAR HASIL** Program Strata Satu (S1) Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin dengan susunan sebagai berikut :

   Pembimbing I/ Ketua : 1. Dr. Amil Ahmad Ilham, ST., M.T
   Anggota : 2. Dr. Ir. Ingrid Nurtanio, M.T
   : 3. Prof. Dr.Eng. Intan Sari Areni, ST., M.T.

   Untuk menguji bagi mahasiswa tersebut dibawah ini :

   Nama/NIM : Amiqatun Nasyati Yusri D121 20 1080
   Program Studi : Teknik Informatika
   Judul thesis/Skrips : Klasifikasi Ekspresi Wajah Peserta Didik dalam Proses Pembelajaran di Kelas

2. Waktu seminar ditetapkan oleh Panitia Seminar Hasil Program Strata Satu (S1)
3. Agar Surat Penugasan ini dilaksanakan sebaik-baiknya dengan penuh rasa tanggung jawab.
4. Surat penugasa ini berlaku sejak tanggal ditetapkan sampai dengan berakhirnya seminar tersebut dengan ketentuan bahwa segala sesuatunya akan ditinjau dan diperbaiki sebagaimana mestinya apabila dikemudia hari terdapat kekeliruan dalam keputusan ini.

Ditetapkan di Gowa
Pada tanggal 19 Juni 2024
a.n. Dekan,
Wakil Dekan Bidang Akademik dan Kemahasiswaan
Fakultas Teknik Unhas

Dr. Amil Ahmad Ilham, ST., M.IT
NIP. 197310101998021001

Tembusan :
1. Dekan Fak. Teknik Unhas
2. Ketua Departemen Teknik Informatika FT-UH
3. Mahasiswa yang bersangkutan

Lampiran 4. Daftar hadir dan berita acara ujian skripsi

KEMENTERIAN PENDIDIKAN , KEBUDAYAAN
RISET DAN TEKNOLOGI
UNIVERSITAS HASANUDDIN
FAKULTAS TEKNIK
DEPARTEMEN TEKNIK INFORMATIKA
Kampus FakultasTeknikUnhas, Jl. PorosMalino, Gowa
http://eng.unhas.ac.id/informatika, Email : informatika@unhas.ac.id

DAFTAR HADIR UJIAN SKRIPSI MAHASISWA
FAKULTAS TEKNIK UNHAS

Nama/Stambuk : 1. Amiqatun Nasyati Yusri D121201080

Judul Skipsi/T.A : **" Klasifikasi Ekspresi Wajah Peserta Didik dalam Proses Pembelajaran di Kelas"**

Hari/Tanggal : Jumat, 19 Juli 2024
Jam : 09.00 Wita – Selesai
Tempat : Ruang Lab. UBICON Departemen Teknik Informatika Gowa

No.	Jabatan	Nama Dosen	Tanda Tangan
L.	Pembimbing I	1. Dr. Ir.Amil Ahmad Ilham,ST.,M.IT	1. .....
II.	Anggota Penguji	2. Dr.Ir. Ingrid Nurtanio,M.T	2 .....
		3. Prof. Dr. Eng.Intan Sari Areni,ST.,M.T	3 .....

PANITIA UJIAN

Ketua,

Dr. Ir.Amil Ahmad Ilham,ST.,M.IT

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET, DAN TEKNOLOGI
UNIVERSITAS HASANUDDIN
FAKULTAS TEKNIK
DEPARTEMEN TEKNIK INFORMATIKA
Kampus Fakultas Teknik Unhas, Jl. Poros Malino, Gowa
http://eng.unhas.ac.id/informatika, Email : informatika@unhas.ac.id

Gowa, 16 Juli 2024

Nomor : 962/UN4.7.7.1/TD.06/2024
Lamp : -
Hal : Usulan Susunan Panitia/Penguji Ujian Sarjana

Yth. : Bapak Wakil Dekan Bidang Akademik dan Kemahasiswaan
Fakultas Teknik Unhas
Di
Gowa

Dalam rangka penyelesaian studi pada Departemen Teknik Informatika Fakultas Teknik Unhas, bersama ini kami usulkan susunan Panitia/Penguji Ujian Sarjana Program Strata Satu (S1) bagi mahasiswa Departemen Teknik Informatika Fakultas Teknik Uniersitas Hasanuddin atas nama :

Pembimbing / Ketua : 1. Dr. Amil Ahmad Ilham
Penguji / Anggota : 2. Dr. Ir. Ingrid Nurtanio, M.T
3. Prof. Dr.Eng. Intan Sari Areni, ST., M.T

Untuk Bertugas sebagai Penguji/ Penanggap Ujian Sarjana bagi Mahasiswa :

Nama : Amiqatun Nasyati Yusri
Stambuk : D121 20 1080

Dengan Judul Skripsi :

" Klasifikasi Ekspresi Wajah Peserta Didik dalam Proses Pembelajaran di Kelas "

Pada :
Hari/Tanggal : Jumar, 19 Juli 2024
Jam : 09.00 Wita - Selesai
Tempat : Ruang Sidang Lab. Ubicom

Demikian penyampaian kami, atas perhatiannya diucapkan terimah kasih.

Ketua Departemen Tek.Informatika,

Prof. Dr. Ir. Indrabayu.,ST, MT, M.Bus.Sys., IPM, ASEAN.Eng
Nip.197507016 200212 1 004

Tembusan :
1. Arsip

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET, DAN TEKNOLOGI
UNIVERSITAS HASANUDDIN
FAKULTAS TEKNIK
Jalan Poros Malino Km. 6 Bontomarannu, Gowa, 92171, Sulawesi Selatan
☎ +62811 4420 909, E-mail: teknik@unhas.ac.id , https://eng.unhas.ac.id

SURAT PENUGASAN

No. 16733/*UN4.7.1/TD.06/2024*

Dari : Dekan Fakultas Teknik Universitas Hasanuddin.

Kepada. : Mereka yang tercantum namanya di bawah ini.

I s i : 1. Bahwa merujuk kepada Peraturan Rektor Universitas Hasanuddin **Nomor : 29/UN4.1/2023 tentang Penyelenggaraan Program Sarjana Universitas Hasanuddin**, dengan ini menugaskan Saudara sebagai **PENGUJI/PANITIA UJIAN SARJANA** Program Strata Satu (S1) Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin dengan susunan sebagai berikut :

Pembimbing / Ketua : 1. Dr. Amil Ahmad Ilham
Penguji / Anggota : 2. Dr. Ir. Ingrid Nurtanio, M.T
3. Prof. Dr.Eng. Intan Sari Areni, ST., M.T.

untuk menguji bagi mahasiswa tersebut di bawah ini :

Nama/NIM	: Amiqatun Nasyati Yusri	D121201080
Program Studi	: Teknik Informatika	
Judul Thesis/Skripsi	: Klasifikasi Ekspresi Wajah Peserta Didik dalam Proses Pembelajaran di Kelas	

2. Waktu Ujian ditetapkan oleh Panitia Ujian Sarjana Program Strata Satu (S1).
3. Agar Surat penugasan ini dilaksanakan sebaik-baiknya dengan penuh rasa tanggung jawab.
4. Surat penugasan ini berlaku sejak tanggal ditetapkan sampai dengan berakhirnya Ujian Sarjana tersebut, dengan ketentuan bahwa segala sesuatunya akan ditinjau dan diperbaiki sebagaimana mestinya apabila dikemudian hari ternyata terdapat kekeliruan dalam keputusan ini.

Ditetapkan di Gowa,
Pada tanggal 16 Juli 2024
a.n. Dekan
Wakil Dekan Bidang Akademik dan Kemahasiswaan
Fakultas Teknik Unhas

Dr. Amil Ahmad Ilham, ST., M.IT
NIP.197310101998021001

Tembusan :
1. Dekan Fak. Teknik Unhas
2. Ketua Departemen Teknik Informatika FT-UH
3. Kasubag. Umum dan Perlengkapan FT-UH

Lampiran 5. Lembar perbaikan skripsi

LEMBAR PERBAIKAN SKRIPSI

"KLASIFIKASI EKSPRESI WAJAH PESERTA DIDIK DALAM PROSES PEMBELAJARAN DI KELAS"

OLEH:

AMIQATUN NASYATI YUSRI
D121201080

Skripsi ini telah dipertahankan pada Ujian Akhir Sarjana tanggal 19 Juli 2024.

Telah dilakukan perbaikan penulisan dan isi skripsi berdasarkan usulan dari penguji dan pembimbing skripsi.

Persetujuan perbaikan oleh tim penguji:

	Nama	Tanda Tangan
Ketua	Dr. Amil Ahmad Ilham, S.T., M.IT	
Anggota	Dr. Ir. Ingrid Nurtanio, M.T.	
	Prof. Dr. Eng. Intan Sari Areni, S.T., M.T.	

Persetujuan Perbaikan oleh pembimbing:

Pembimbing	Nama	Tanda Tangan
I	Dr. Amil Ahmad Ilham, S.T., M.IT	