DAFTAR PUSTAKA

Aldo, D. *et al.* (2022) 'Pengembangan Sistem Informasi Terpadu Industri Pariwisata Kota Batam Menerapkan User Centered Design Berbasis Website', *Jurnal Media Informatika Budidarma*, 6(2), p. 898. Available at: https://doi.org/10.30865/mib.v6i2.3849.

Ansari, I.I. *et al.* (2025) 'The effect of Sleep Deprivation on Cognitive Performance', *Bulletin of Business and Economics (BBE)*, 14(1), pp. 39–43. Available at: https://doi.org/10.61506/01.00577.

Anwar, K. (2025) '"Sistem Deteksi Wajah Berbasis Deep Learning Menggunakan Convolutional Neural Network (CNN)"', *Journal of Computer Science and Information Technology*, 1(2), pp. 46–52. Available at: https://doi.org/10.70716/jocsit.v1i2.258.

Baareh, A.K.M. (2024) 'Facial Recognition and Discovery Using Convolution Deep Learning Neural Network', *Journal of Computer Science*, 20(11), pp. 1559–1568. Available at: https://doi.org/10.3844/jcssp.2024.1559.1568.

Dagani, J. *et al.* (2024) 'The Interplay of Sleep Quality, Mental Health, and Sociodemographic and Clinical Factors among Italian College Freshmen', *Journal of Clinical Medicine*, 13(9). Available at: https://doi.org/10.3390/jcm13092626.

Deng, Q. and Govindasamy, S. (2022) 'Uplink Spectral Efficiency of Large, Distributed Antenna Systems With MMSE Processing', *IEEE Access*, 10, pp. 23198–23216. Available at: https://doi.org/10.1109/ACCESS.2022.3153045.

Ern, L.J. and Sia, F. (2023) 'A Mobile-Based Application for Detecting Sleep Deprivation Using Deep Learning for University Student', *5th IEEE International Conference on Artificial Intelligence in Engineering and Technology, IICAIET 2023*, pp. 331–335. Available at: https://doi.org/10.1109/IICAIET59451.2023.10291804.

Kao, I.H. and Chan, C.Y. (2022) 'Comparison of Eye and Face Features on Drowsiness Analysis', *Sensors*, 22(17). Available at: https://doi.org/10.3390/s22176529.

Majeed, F. *et al.* (2023) 'Detection of Drowsiness among Drivers Using Novel Deep Convolutional Neural Network Model', *Sensors (Basel, Switzerland)*, 23(21). Available at: https://doi.org/10.3390/s23218741.

Matsubara, A. *et al.* (2023) 'Sleep Deprivation Increases Facial Skin Yellowness', *Journal of Clinical Medicine*, 12(2). Available at: https://doi.org/10.3390/jcm12020615.

Mohialden, Y.M. *et al.* (2024) 'Top Python-Based Deep Learning Packages: A Comprehensive Review', *International Journal Papier Advance and Scientific Review*, 5(1), pp. 1–9. Available at: https://doi.org/10.47667/ijpasr.v5i1.283.

Nayoma, F.S. and Kusnawi, K. (2025) 'Integration of BERT-VAD, MFCC-Delta, and VGG16 in Transformer-Based Fusion Architecture for Multimodal Emotion Classification', *Jurnal Teknik Informatika (Jutif)*, 6(4), pp. 2586–2601. Available at: https://doi.org/10.52436/1.jutif.2025.6.4.4915.

Seandrio, A.L., Pratomo, A.H. and Florestiyanto, M.Y. (2021) 'Implementation of Convolutional Neural Network (CNN) in Facial Expression Recognition', *Telematika*, 18(2), p. 211. Available at: https://doi.org/10.31315/telematika.v18i2.4823.

Sivakumar, M., Parthasarathy, S. and Padmapriya, T. (2024) 'Trade-off between training and testing ratio in machine learning for medical image processing', *PeerJ Computer Science*, 10. Available at: https://doi.org/10.7717/PEERJ-CS.2245.

Suardiaz-Muro, M. *et al.* (2023) 'Sleep quality and sleep deprivation: relationship with academic performance in university students during examination period', *Sleep and Biological Rhythms*, 21(3), pp. 377–383. Available at: https://doi.org/10.1007/s41105-023-00457-1.

Sujon, K.M. *et al.* (2025) 'Accuracy, precision, recall, f1-score, or MCC? empirical evidence from advanced statistics, ML, and XAI for evaluating business predictive models', *Journal of Big Data*, 12(1). Available at: https://doi.org/10.1186/s40537-025-01313-4.

Vijaypriya, V. and Uma, M. (2023) ‘Facial Feature-Based Drowsiness Detection with Multi-Scale Convolutional Neural Network’, *IEEE Access*, 11, pp. 63417–63429. Available at: https://doi.org/10.1109/ACCESS.2023.3288008.

L

A

M

P

I

R

A

N

Code Sistem

```
import os
os.environ["TF_CPP_MIN_LOG_LEVEL"] = "2"

import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers, models
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.applications.mobilenet_v2 import
preprocess_input
import matplotlib.pyplot as plt

print("====================================")
print("  TRAINING MobileNetV2 RINGAN      ")
print("====================================")

# ==============================
# KONFIGURASI
# ==============================
train_dir = "dataset/train"  # folder dataset
IMG_SIZE = 160      # ukuran gambar input
BATCH_SIZE = 8      # batch lebih kecil untuk ringan
EPOCHS = 20

# ==============================
# DATA GENERATOR
# ==============================
train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    rotation_range=10,
    zoom_range=0.1,
    horizontal_flip=True
)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)

# ==============================
# BASE MODEL LEBIH RINGAN
```

```python
# ================================
base_model = MobileNetV2(
    weights='imagenet',
    include_top=False,
    input_shape=(IMG_SIZE, IMG_SIZE, 3),
    alpha=0.5   # model lebih ringan
)

# Fine-tuning sebagian layer terakhir
for layer in base_model.layers[:-20]:
    layer.trainable = False

for layer in base_model.layers[-20:]:
    layer.trainable = True

# ================================
# BUILD MODEL
# ================================
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(4, activation='softmax')  # 4 kelas
])

# ================================
# COMPILE
# ================================
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

# ================================
# TRAINING
# ================================
history = model.fit(
    train_generator,
    epochs=EPOCHS
)

# ================================
# SAVE MODEL
```

```
# ================================
os.makedirs("models", exist_ok=True)
model.save("models/model_mobilenet_4kelas.keras")
print("\nModel berhasil disimpan!")

# ================================
# SIMPAN GRAFIK AKURASI & LOSS KE FILE
# ================================
# Akurasi Training
plt.figure()
plt.plot(history.history['accuracy'], marker='o')
plt.title('Akurasi Training')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.grid(True)
plt.savefig("models/accuracy_training.png")
plt.close()

# Loss Training
plt.figure()
plt.plot(history.history['loss'], marker='o', color='red')
plt.title('Loss Training')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.grid(True)
plt.savefig("models/loss_training.png")
plt.close()

print("Grafik akurasi dan loss berhasil disimpan di folder
'models'")
```

```
import tensorflow as tf
import numpy as np
import cv2
import os
from sklearn.metrics import classification_report, accuracy_score

print("====================================")
print("   TESTING DATASET HANYA TEST       ")
print("====================================")

# ================================
# LOAD MODEL
```

```
# ================================
model =
tf.keras.models.load_model("models/model_mobilenet_4kelas.keras")

# ================================
# NAMA KELAS
# ================================
class_names = [
    "lingkaran_hitam",
    "mata_merah",
    "normal",
    "wajah_kusam"
]

# Folder dataset test
test_dir = "dataset/test"

y_true = []
y_pred = []

# ================================
# LOOP DATA TEST
# ================================
for label in class_names:

    folder = os.path.join(test_dir, label)

    if not os.path.exists(folder):
        print(f"[WARNING] Folder {folder} tidak ditemukan!")
        continue

    files = sorted(os.listdir(folder))

    for file in files:

        if not file.lower().endswith((".jpg", ".jpeg", ".png")):
            continue

        img_path = os.path.join(folder, file)
        img = cv2.imread(img_path)

        if img is None:
            print(f"[WARNING] Gagal membaca {img_path}")
            continue
```

```
        print(f"[INFO] Memproses {img_path}")

        # ================================
        # PREPROCESSING
        # ================================
        img_model = cv2.resize(img, (160,160))
        img_norm = img_model / 255.0
        img_input = np.expand_dims(img_norm, axis=0)

        # ================================
        # PREDIKSI MODEL
        # ================================
        pred = model.predict(img_input, verbose=0)
        pred_class = np.argmax(pred)
        confidence = np.max(pred) * 100
        kondisi = class_names[pred_class]

        y_true.append(label)
        y_pred.append(kondisi)

        # ================================
        # CEK BENAR ATAU SALAH
        # ================================
        if kondisi == label:
            warna_prediksi = (0,255,0)   # hijau
        else:
            warna_prediksi = (0,0,255)   # merah

        hasil_deteksi = f"TERDETEKSI: {kondisi}"

        # ================================
        # STATUS & SARAN (BERDASARKAN LABEL ASLI)
        # ================================
        if label == "normal":
            status = "WAJAH NORMAL"
            saran = "Tidak perlu tindakan khusus"

        elif label == "mata_merah":
            status = "DATA ASLI: MATA MERAH"
            saran = "Disarankan istirahat atau periksa ke dokter
mata"

        elif label == "lingkaran_hitam":
            status = "DATA ASLI: LINGKARAN HITAM"
            saran = "Disarankan tidur cukup"
```

```
        elif label == "wajah_kusam":
            status = "DATA ASLI: WAJAH KUSAM"
            saran = "Disarankan perawatan kulit atau tidur cukup"

        else:
            status = "KONDISI TIDAK DIKETAHUI"
            saran = "-"

        # ==============================
        # TAMPILKAN GAMBAR
        # ==============================
        img_display = cv2.resize(img, (400,400))

        cv2.putText(img_display,f"Foto: {file}",(10,30),
                    cv2.FONT_HERSHEY_SIMPLEX,0.6,(255,255,0),2)

        cv2.putText(img_display,f"Label Asli: {label}",(10,60),
                    cv2.FONT_HERSHEY_SIMPLEX,0.6,(255,255,0),2)

        cv2.putText(img_display,hasil_deteksi,(10,90),
                    cv2.FONT_HERSHEY_SIMPLEX,0.7,warna_prediksi,2)

        cv2.putText(img_display,f"Confidence:
{confidence:.2f}%",(10,120),
                    cv2.FONT_HERSHEY_SIMPLEX,0.7,warna_prediksi,2)

        cv2.putText(img_display,status,(10,150),
                    cv2.FONT_HERSHEY_SIMPLEX,0.7,(0,255,0),2)

        cv2.putText(img_display,f"Saran: {saran}",(10,180),
                    cv2.FONT_HERSHEY_SIMPLEX,0.6,(255,255,0),2)

        cv2.imshow("Deteksi Wajah", img_display)

        # Tekan Q untuk berhenti
        key = cv2.waitKey(0) & 0xFF
        if key == ord('q'):
            print("Deteksi dihentikan oleh user.")
            cv2.destroyAllWindows()
            exit()

# ==============================
# EVALUASI MODEL
# ==============================
```

```
cv2.destroyAllWindows()

if len(y_true) == 0:
    print("Tidak ada data test yang diproses.")
else:

    print("\n====================================")
    print("        HASIL EVALUASI MODEL")
    print("====================================")

    acc = accuracy_score(y_true,y_pred)

    print("\nAccuracy :", round(acc*100,2),"%")

    print("\nClassification Report :\n")

    print(classification_report(
        y_true,
        y_pred,
        target_names=class_names
    ))
```

```
import tensorflow as tf
import numpy as np
import cv2
from tkinter import Tk
from tkinter.filedialog import askopenfilename
from tensorflow.keras.applications.mobilenet_v2 import
preprocess_input
from PIL import Image

print("====================================")
print(" DETEKSI KONDISI WAJAH MAHASISWA ")
print("====================================")

# ================================
# LOAD MODEL
# ================================
model =
tf.keras.models.load_model("models/model_mobilenet_4kelas.keras")

# ================================
# NAMA KELAS
```

```python
# ================================
class_names = [
    "lingkaran_hitam",
    "mata_merah",
    "normal",
    "wajah_kusam"
]

# ================================
# FACE DETECTOR
# ================================
face_cascade = cv2.CascadeClassifier(
    cv2.data.haarcascades + "haarcascade_frontalface_default.xml"
)

# ================================
# PILIH FOTO
# ================================
Tk().withdraw()

file_path = askopenfilename(
    title="Pilih Foto",
    filetypes=[("Image Files","*.jpg *.jpeg *.png *.webp")]
)

if not file_path:
    print("Tidak ada foto dipilih")
    exit()

print("Foto dipilih:", file_path)

# ================================
# LOAD GAMBAR
# ================================
try:
    img_pil = Image.open(file_path).convert("RGB")
    img = np.array(img_pil)
    img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
except:
    print("Gagal membaca gambar")
    exit()

# ================================
# RESIZE GAMBAR BESAR
# ================================
```

```python
max_width = 900

if img.shape[1] > max_width:
    scale = max_width / img.shape[1]
    new_height = int(img.shape[0] * scale)
    img = cv2.resize(img,(max_width,new_height))

# ================================
# DETEKSI WAJAH
# ================================
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = face_cascade.detectMultiScale(
    gray,
    scaleFactor=1.2,
    minNeighbors=5,
    minSize=(60,60)
)

# ================================
# JIKA WAJAH TIDAK TERDETEKSI
# ================================
if len(faces) == 0:

    face = img

    face_resized = cv2.resize(face,(160,160))

    face_rgb = cv2.cvtColor(face_resized,cv2.COLOR_BGR2RGB)

    face_input = np.expand_dims(face_rgb,axis=0)

    face_input = preprocess_input(face_input)

    pred = model.predict(face_input)

    probabilities = pred[0]

    pred_class = np.argmax(probabilities)

    confidence = probabilities[pred_class] * 100

    kondisi = class_names[pred_class]

# ================================
```

```
# JIKA WAJAH TERDETEKSI
# ==============================
else:

    (x,y,w,h) = faces[0]

    face = img[y:y+h, x:x+w]

    face_resized = cv2.resize(face,(160,160))

    face_rgb = cv2.cvtColor(face_resized,cv2.COLOR_BGR2RGB)

    face_input = np.expand_dims(face_rgb,axis=0)

    face_input = preprocess_input(face_input)

    pred = model.predict(face_input)

    probabilities = pred[0]

    pred_class = np.argmax(probabilities)

    confidence = probabilities[pred_class] * 100

    kondisi = class_names[pred_class]

# ==============================
# SARAN
# ==============================
if kondisi == "normal":
    saran = "Wajah dalam kondisi normal"

elif kondisi == "mata_merah":
    saran = "Disarankan istirahat atau periksa mata"

elif kondisi == "lingkaran_hitam":
    saran = "Disarankan tidur cukup"

elif kondisi == "wajah_kusam":
    saran = "Disarankan perawatan kulit dan tidur cukup"

else:
    saran = "-"

# ==============================
```

```
# OUTPUT TERMINAL
# ================================
print("\nHasil Deteksi")
print("Prediksi :", kondisi)
print("Confidence :", round(confidence,2),"%")
print("Saran :", saran)

# ================================
# TAMPILKAN DI GAMBAR
# ================================
cv2.putText(img,f"Prediksi: {kondisi}",(20,40),
            cv2.FONT_HERSHEY_SIMPLEX,0.8,(0,255,0),2)

cv2.putText(img,f"Confidence: {confidence:.2f}%",(20,80),
            cv2.FONT_HERSHEY_SIMPLEX,0.8,(0,255,0),2)

cv2.putText(img,f"Saran: {saran}",(20,120),
            cv2.FONT_HERSHEY_SIMPLEX,0.7,(255,255,0),2)

img_display = cv2.resize(img,(650,650))

cv2.imshow("Deteksi Kondisi Wajah",img_display)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

```
import tensorflow as tf
import numpy as np
import cv2
from tensorflow.keras.applications.mobilenet_v2 import
preprocess_input

print("====================================")
print(" DETEKSI KONDISI WAJAH REALTIME ")
print(" Prediksi hanya dilakukan sekali ")
print(" Tekan Q untuk keluar ")
print("====================================")

# ================================
# LOAD MODEL
# ================================
```

```
model =
tf.keras.models.load_model("models/model_mobilenet_4kelas.keras")

# ================================
# NAMA KELAS
# ================================
class_names = [
    "lingkaran_hitam",
    "mata_merah",
    "normal",
    "wajah_kusam"
]

# ================================
# FACE DETECTOR
# ================================
face_cascade = cv2.CascadeClassifier(
    cv2.data.haarcascades + "haarcascade_frontalface_default.xml"
)

# ================================
# AKTIFKAN KAMERA
# ================================
cap = cv2.VideoCapture(0)

# variabel supaya prediksi hanya sekali
sudah_prediksi = False
kondisi = ""
confidence = 0
saran = ""

while True:

    ret, frame = cap.read()

    if not ret:
        break

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = face_cascade.detectMultiScale(
        gray,
        scaleFactor=1.3,
        minNeighbors=5,
        minSize=(60,60)
```

```
    )

    for (x,y,w,h) in faces:

        # jika belum diprediksi
        if not sudah_prediksi:

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

            face_resized = cv2.resize(face,(160,160))

            face_rgb = cv2.cvtColor(face_resized, cv2.COLOR_BGR2RGB)

            face_input = np.expand_dims(face_rgb, axis=0)

            face_input = preprocess_input(face_input)

            pred = model.predict(face_input, verbose=0)

            probabilities = pred[0]

            pred_class = np.argmax(probabilities)

            confidence = probabilities[pred_class] * 100

            kondisi = class_names[pred_class]

            # ==============================
            # SARAN
            # ==============================
            if kondisi == "normal":
                saran = "Wajah normal"

            elif kondisi == "mata_merah":
                saran = "Istirahat atau periksa mata"

            elif kondisi == "lingkaran_hitam":
                saran = "Disarankan tidur cukup"

            elif kondisi == "wajah_kusam":
                saran = "Perawatan kulit dan tidur cukup"

            else:
                saran = "-"
```

```
            sudah_prediksi = True

            print("\nHasil Deteksi")
            print("Prediksi :", kondisi)
            print("Confidence :", round(confidence,2),"%")
            print("Saran :", saran)

        # ==============================
        # TAMPILKAN HASIL
        # ==============================
        cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)

        cv2.putText(frame,f"Prediksi: {kondisi}",
                    (x,y-40),
                    cv2.FONT_HERSHEY_SIMPLEX,
                    0.7,(0,255,0),2)

        cv2.putText(frame,f"Confidence: {confidence:.2f}%",
                    (x,y-15),
                    cv2.FONT_HERSHEY_SIMPLEX,
                    0.6,(0,255,0),2)

        cv2.putText(frame,f"Saran: {saran}",
                    (x,y+h+25),
                    cv2.FONT_HERSHEY_SIMPLEX,
                    0.6,(255,255,0),2)

    cv2.imshow("Deteksi Wajah Realtime", frame)

    # tekan Q untuk keluar
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```