DAFTAR PUSTAKA

Agrawal, Aakash. (2021, 11 Januari). *The Why and the How of Deep Metric Learning*. https://towardsdatascience.com/the-why-and-the-how-of-deep-metric-learning-e70e16e199c0

Alhanaee, K., Alhammadi, M. dkk. (2021). *Face Recognition Smart Attendance System using Deep Transfer Learning*. Procedia Computer Science, 192, 4093-4102.

Cahyono, Ferry. (2020). *Pengenalan Wajah Menggunakan Model Facenet Untuk Presensi Pegawai*. Tesis. Institut Teknologi Sepuluh November.

Devi, Putri. & Rosyid, Harunur. (2022). *Pemaparan Materi Dasar Pengolahan Citra Digital untuk Upgrade Wawasan Siswa di SMK Dharma Wanita Gresik*. Jurnal Abdi Masyarakat, 2(4), 1259-1264.

Dharmadinata, Owen. & Kusuma, Gede. (2023). *Improving Face Recognition In Low Illumination Condition Using Combination Of Image Enhancement And Face Recognition Methods*. Journal of Theoretical and Applied Information Technology, 101(4), 1538-1551.

Evelyn., Rudi, A., & Gunadi, K. (2022). *Sistem Presensi Mahasiswa Menggunakan Face Recognition Dengan Metode Facenet Pada Android*. Skripsi. Universitas Kristen Petra.

Fajri, Effendi, R., & Fadillah, N. (2020). *Sistem Absensi Berbasis Pengenalan Wajah Secara Real Time menggunakan Metode Fisherface*. Jurnal Nasional Informatika dan Teknologi Jaringan, 4(2), 350-353.

Geitgey, Adam. (2016, 24 Juli). *Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning*. https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

Gupta, Divam. (2023). *Image Segmentation Keras: Implementation of Segnet, FCN, UNet, PSPNet and other models in Keras*. Journal of Computer Vision and Pattern Recognition, 102, 2-4.

Jyotsana. (2023, 26 Maret). *Deep Metric Learning- Fundamentals*. https://medium.com/@jyotsana.cg/deep-metric-learning-part-1-4c2dc7d7ca17

Kaya, M., & Bilge, H.S. (2019). *Deep Metric Learning: A Survey*. Symmetry, 11(9), 1066.

Krizheysky, A., Sutskever, I., & Hinton, G. (2012). *Imagenet classification with deep convolutional neural networks*. Advances in Neural Information Processing Systems, 25, 1097–1105.

Kurniawati, Yunita. (2019). *Sistem Presensi Kelas Menggunakan Pengenalan Wajah Dengan Metode Haar Cascade Classifier*. Skripsi. Universitas Semarang.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). *Deep learning*. Nature, 521, 436–444.

LinovHR. (2022, 25 Mei). *Mengenal Absen Biometrik: Jenis, Kelebihan, dan Mengapa harus Menggunakannya*. https://www.linovhr.com/absen-biometrik-adalah/

Nurfadillah, Fikri. (2020). *Sistem Absensi Dengan Face Recognition Menggunakan OpenCV Berbasis Web*. Skripsi. Politeknik Negeri Jakarta.

Nurfita, Royani Darma. (2018). *Implementasi Deep Learning Berbasis Tensorflow Untuk Pengenalan Sidik Jari*. Tesis. Universitas Muhammadiyah Surakarta.

Pypi. (2020, 20 Februari). Face-recognition 1.3.0. https://pypi.org/project/face-recognition/

Ramadhan, Anugrah. (2022). *Aplikasi Face Recognition Untuk Absensi Mahasiswa Dengan Menggunakan Metode 128D Embedding*. Skripsi. Sekolah Tinggi Manajemen Informatika dan Komputer Indonesia Mandiri.

Sabili, S., Rachmadi, R., & Yuniarno, E. (2021). *Verifikasi Wajah Menggunakan Deep Metric Learning pada Data Wajah dengan Disparitas Umur yang Besar*. Jurnal Teknik ITS, 10(2), 432-437.

Sarigoz, Yusuf. (2022, 25 Maret). *Triplet Loss — Advanced Intro*. https://towardsdatascience.com/triplet-loss-advanced-intro-49a07b7d8905

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). *FaceNet: A Unified Embedding for Face Recognition and Clustering*. IEEE conference on computer vision and pattern recognition, 815-823.

Setiono, P., Sompie, S., & Najoan, M. (2020). *Aplikasi Pengenalan Wajah Untuk Sistem Absensi Kelas Berbasis Raspberry Pi*. Jurnal Teknik Informatika, 15(3), 179-188.

Siahaan, Bernad. (2021). *Implementasi Pengenalan Wajah Untuk Absensi Karyawan Dengan Metode Eigenface*. Skripsi. Universitas Putera Batam.

Stack Overflow. (2022). *2022 Developer Survey*. https://survey.stackoverflow.co/2022/

Thammaiah, C., & Nagavi, T. (2021). *An Approach to Partial Occlusion Using Deep Metric Learning*. International Journal of Informatics and Communication Technology, 10(3), 204-211.

Utaminingrum, Fitri. (2017). *Pengolahan Citra Digital*. Universitas Brawijaya.

Wiatowski, T., & Bolcskei, H. (2018). *A Mathematical Theory of Deep Convolutional Neural Networks for Feature Extraction*. IEEE Transactions on Information Theory, 64(3), 1845-1866.

Wijaya, Alexander Eric, dkk. (2021). *Implementasi Transfer Learning Pada Convolutional Neural Network Untuk Diagnosis Covid-19 Dan Pneumonia Pada Citra X-Ray*. Tesis. Universitas Ma Chung.

Zihamussholihin, Z., Kartikowati, R.S., & Azhar, F. (2021). *Implementasi Kedisiplinan Dan Sasaran Kerja Pegawai Dalam Meningkatkan Kinerja Di Balai Bahasa Riau*. Jurnal Manajemen Pendidikan, 9(1), 82-95.

LAMPIRAN

Lampiran 1 Kode Program Untuk Melatih Identifikasi Masker

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers.legacy import Adam
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import argparse
import os

ap = argparse.ArgumentParser()
ap.add_argument("-d", "--dataset", required=True,
  help="path to input dataset")
ap.add_argument("-p", "--plot", type=str, default="plot.png",
  help="path to output loss/accuracy plot")
ap.add_argument("-m", "--model", type=str,
  default="mask_detector.model",
  help="path to output face mask detector model")
args = vars(ap.parse_args())

INIT_LR = 0.001
EPOCHS = 50
BS = 32
imagePaths = list(paths.list_images(args["dataset"]))
data = []
labels = []
```

```
for imagePath in imagePaths:
  label = imagePath.split(os.path.sep)[-2]
  image = load_img(imagePath, target_size=(128, 128))
  image = img_to_array(image)
  image = preprocess_input(image)
  data.append(image)
  labels.append(label)

data = np.array(data, dtype="float32")
labels = np.array(labels)
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)
(trainX, testX, trainY, testY) = train_test_split(data, labels, test_size=0.20,
stratify=labels, random_state=42)

aug = ImageDataGenerator(rotation_range=20, zoom_range=0.15,
width_shift_range=0.2, height_shift_range=0.2, shear_range=0.15,
horizontal_flip=True, fill_mode="nearest")

baseModel = MobileNetV2(weights="imagenet", include_top=False,
input_tensor=Input(shape=(224, 224, 3)))

headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)
model = Model(inputs=baseModel.input, outputs=headModel)
for layer in baseModel.layers:
  layer.trainable = False
headModel.summary()
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt, metrics=["accuracy"])

H = model.fit(aug.flow(trainX, trainY, batch_size=BS), steps_per_epoch=len(trainX)
// BS, validation_data=(testX, testY), validation_steps=len(testX) // BS,
epochs=EPOCHS)
```

```
predIdxs = model.predict(testX, batch_size=BS)
predIdxs = np.argmax(predIdxs, axis=1)
model.save(args["model"], save_format="h5")

N = EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig(args["plot"])
```

Lampiran 2 Kode Program Untuk Melatih Pengenalan Wajah

```
import cv2 as cv
import os
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from keras_facenet import FaceNet
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, GlobalAveragePooling2D,
        BatchNormalization, Dense, Dropout, Lambda
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.resnet50 import ResNet50
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

class FACELOADING:
    def __init__(self, directory):
        self.directory = directory
        self.target_size = (128, 128)
        self.X = []
        self.Y = []
        self.detector = cv.CascadeClassifier("haarcascade_frontalface_default.xml")

    def extract_face(self, filename):
        img = cv.imread(filename)
        img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
        x, y, w, h = self.detector.detectMultiScale(img, 1.3, 5)(img)[0]['box']
        x, y = abs(x), abs(y)
        face = img[y:y + h, x:x + w]
        face_arr = cv.resize(face, self.target_size)
        return face_arr

    def load_faces(self, dir):
        FACES = []
        for im_name in os.listdir(dir):
            try:
                path = dir + im_name
                single_face = self.extract_face(path)
                FACES.append(single_face)
            except Exception as e:
                pass
```

```
        return FACES

    def load_classes(self):
        for sub_dir in os.listdir(self.directory):
            path = self.directory + '/' + sub_dir + '/'
            FACES = self.load_faces(path)
            labels = [sub_dir for _ in range(len(FACES))]
            print(f"Loaded successfully: {len(labels)}")
            self.X.extend(FACES)
            self.Y.extend(labels)
        return np.asarray(self.X), np.asarray(self.Y)

faceloading = FACELOADING("images")
X, Y = faceloading.load_classes()

embedder = FaceNet()

def get_embedding(face_img):
    face_img = face_img.astype('float32')
    face_img = np.expand_dims(face_img, axis=0)
    yhat= embedder.embeddings(face_img)
    return yhat[0] #512D image

EMBEDDED_X = []
for img in X:
    EMBEDDED_X.append(get_embedding(img))
EMBEDDED_X = np.asarray(EMBEDDED_X)
np.savez_compressed('faces_embeddings_done_4classes.npz', EMBEDDED_X, Y)

X_train, X_test, Y_train, Y_test = train_test_split(EMBEDDED_X, Y, test_size =
        0.2, random_state=17)

def create_model(input_shape):
    base_model = ResNet50(input_shape=input_shape, include_top=False,
        weights='imagenet')
    base_model.trainable = False

    input = tf.keras.Input(shape=input_shape)
    x = base_model(input, training=False)
    x = GlobalAveragePooling2D()(x)
    x = Dense(1024, activation='relu')(x)
```

```
    x = BatchNormalization()(x)
    x = Dropout(0.5)(x)
    x = Dense(512, activation='relu')(x)
    x = BatchNormalization()(x)
    x = Dropout(0.5)(x)
    output = Dense(128, activation='relu')(x)

    # Normalisasi L2
    output = Lambda(lambda x: tf.math.l2_normalize(x, axis=1))(output)
    model = Model(input, output)
    return model

def triplet_loss(y_true, y_pred, margin=1):
    anchor, positive, negative = y_pred[:, 0], y_pred[:, 1], y_pred[:, 2]
    pos_dist = tf.reduce_sum(tf.square(anchor - positive), axis=-1)
    neg_dist = tf.reduce_sum(tf.square(anchor - negative), axis=-1)
    basic_loss = pos_dist - neg_dist + margin
    loss = tf.reduce_mean(tf.maximum(basic_loss, 0.0))
    return loss

def accuracy(embeddings, labels, threshold=0.6):
    n_correct = 0
    n_total = 0
    for i in range(len(labels)):
        for j in range(i+1, len(labels)):
            if labels[i] == labels[j]:
                if np.linalg.norm(embeddings[i] - embeddings[j]) < threshold:
                    n_correct += 1
            else:
                if np.linalg.norm(embeddings[i] - embeddings[j]) >= threshold:
                    n_correct += 1
            n_total += 1
    accuracy = n_correct / n_total
    return accuracy

def generate_triplets(embeddings, labels, num_triplets):
    triplets = []
    while len(triplets) < num_triplets:
        anchor_idx = np.random.choice(len(labels))
        anchor_label = labels[anchor_idx]
        positive_indices = np.where(labels == anchor_label)[0]
```

```
        positive_idx = np.random.choice(positive_indices)
        negative_indices = np.where(labels != anchor_label)[0]
        if len(negative_indices) == 0:
            continue
        negative_idx = np.random.choice(negative_indices)
        triplets.append([embeddings[anchor_idx], embeddings[positive_idx],
        embeddings[negative_idx]])
    return np.array(triplets)

input_shape = (160, 160, 3)
embedding_model = create_model(input_shape)
embedding_model.compile(optimizer = Adam(0.001), loss=triplet_loss)

triplets_train = generate_triplets(X_train, Y_train, len(X_train))
triplets_test = generate_triplets(X_test, Y_test, len(X_test))

anchor_input = Input(input_shape, name='anchor_input')
positive_input = Input(input_shape, name='positive_input')
negative_input = Input(input_shape, name='negative_input')

encoded_anchor = embedding_model(anchor_input)
encoded_positive = embedding_model(positive_input)
encoded_negative = embedding_model(negative_input)

merged_output = Lambda(lambda x: tf.stack(x, axis=1))([encoded_anchor,
        encoded_positive, encoded_negative])

triplet_model = Model([anchor_input, positive_input, negative_input],
        merged_output)
triplet_model.compile(optimizer=Adam(0.001), loss=triplet_loss,
        metrics=[accuracy])

history = triplet_model.fit(
    [triplets_train[:, 0], triplets_train[:, 1], triplets_train[:, 2]], np.zeros(len(X_train)),
        epochs=50, validation_data = ([triplets_test[:, 0], triplets_test[:, 1],
        triplets_test[:, 2]], len(X_test)))

plt.figure(figsize=(12, 6))
plt.plot(history.history['loss'], label='Training loss')
plt.plot(history.history['val_loss'], label='Validation loss')
plt.ylabel('Loss')
```

```
plt.xlabel('Epoch')
plt.legend()
plt.show()

plt.figure(figsize=(12, 6))
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```

Lampiran 3 Kode Program Sistem Absensi

```
import tkinter as tk
from tkinter import font
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from datetime import datetime
import argparse
import time
import cv2
import numpy as np
import os
from sklearn.preprocessing import LabelEncoder
from keras_facenet import FaceNet
import gspread
from openpyxl import Workbook
from oauth2client.service_account import ServiceAccountCredentials
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

def update_time():
    current_time = time.strftime('%H.%M.%S')
    time_label.config(text=current_time)
    root.after(1000, update_time)
    if current_time == "21.00.00":
        date = datetime.now().strftime("%d %B %Y")
        data = spreadsheet.get_all_values()
        workbook = Workbook()
        sheet_excel = workbook.active
        for row in data:
            sheet_excel.append(row)
        workbook.save(f'{date}.xlsx')
        spreadsheet.batch_clear(['C5:I14'])
def terjemahan_hari():
    hari_mapping = {
    "Monday": "Senin",
     "Tuesday": "Selasa",
     "Wednesday": "Rabu",
     "Thursday": "Kamis",
     "Friday": "Jumat",
     "Saturday": "Sabtu",
     "Sunday": "Minggu"}
```

```
    day_string = time.strftime("%A")
    return hari_mapping.get(day_string, day_string)

def update_date():
    day = terjemahan_hari()
    date = datetime.now().strftime("%d %B %Y")
    date_label.config(text=f"{day}, {date}")

def set_button_state(button, state, color):
    button.config(state=state, bg=color)

def datang():
    jam = int(time.strftime("%H"))
    menit = int(time.strftime("%M"))
    lanjut = False
    video_capture = cv2.VideoCapture(0)
    time.sleep(2.0)
    start_time = time.time()
    #cek masker
    while True:
        _, frame = video_capture.read()
        (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)
        for (box, pred) in zip(locs, preds):
            (startX, startY, endX, endY) = box
            (mask, withoutMask) = pred
            if mask > withoutMask:
                label = "Buka masker"
                color = (0, 255, 255)
            else:
                lanjut = True
                break
            cv2.putText(frame, label, (startX, startY - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.9, color, 4)
            cv2.rectangle(frame, (startX, startY), (endX, endY), color, 7)
            remark_label.config(text="Buka masker", fg="black")
        if lanjut:
            break
        cv2.imshow("attendence system", frame)
        if time.time() - start_time > 4:
            current_time = time.strftime('%H.%M.%S')
            time_in_label.config(text=current_time)
```

```
            break
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    if not lanjut:
        video_capture.release()
        cv2.destroyAllWindows()
        set_button_state(arrival_button, tk.DISABLED, '#16753D')
        set_button_state(clear_button, tk.NORMAL, '#455a64')
        set_button_state(departure_button, tk.DISABLED, '#0D4577')
    #proses absensi
    else:
        start_time = time.time()
        isi = True
        while True:
            _, frame = video_capture.read()
            rgb_img = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            gray_img = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            faces = haarcascade.detectMultiScale(gray_img, 1.3, 5)
            for x, y, w, h in faces:
                img = rgb_img[y:y + h, x:x + w]
                img = cv2.resize(img, (128, 128))
                img = np.expand_dims(img, axis=0)
                xpred = facenet.embeddings(img)
                distances = np.linalg.norm(X - xpred, axis=1)
                min_distance_index = np.argmin(distances)
                min_distance = distances[min_distance_index]
                predicted_label = Y[min_distance_index]
                threshold = 0.6
                if min_distance <= threshold:
                    cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 7)
                    cv2.putText(frame, predicted_label, (x, y - 10),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 4, cv2.LINE_AA)
                    name_label.config(text=predicted_label)
                    if jam < 8 or (jam == 8 and menit <= 55):
                        remark_label.config(text="Lebih awal", fg="#1E9C43")
                        data_absen = [predicted_label, time.strftime("%H.%M.%S"), "Lebih
awal"]
                    elif (jam == 8 and menit > 55) or (jam == 9 and menit <= 5):
                        remark_label.config(text="Tepat waktu", fg="blue")
                        data_absen = [predicted_label, time.strftime("%H.%M.%S"), "Tepat
waktu"]
```

```
                else:
                    remark_label.config(text="Terlambat", fg="red")
                    data_absen = [predicted_label, time.strftime("%H.%M.%S"),
"Terlambat"]
                baris = 5
                while isi:
                    cell_value = spreadsheet.acell(f'C{baris}').value
                    if cell_value == "None":
                        spreadsheet.update([data_absen], f"C{baris}:E{baris}")
                        isi = False
                        break
                    elif cell_value == predicted_label:
                        remark_label.config(text="Sudah absen", fg="black")
                        break
                    else:
                        baris = baris + 1
            else:
                cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 7)
                cv2.putText(frame, "Tidak dikenali", (x, y - 10),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 4, cv2.LINE_AA)
                remark_label.config(text="Tidak dikenali", fg="black")
        current_time = time.strftime('%H.%M.%S')

        time_in_label.config(text=current_time)
        cv2.imshow("attendence system", frame)
        if time.time() - start_time > 4:
            break
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    video_capture.release()
    cv2.destroyAllWindows()
    set_button_state(arrival_button, tk.DISABLED, '#16753D')
    set_button_state(clear_button, tk.NORMAL, '#455a64')
    set_button_state(departure_button, tk.DISABLED, '#0D4577')

def bersihkan():
    name_label.config(text="")
    time_in_label.config(text="")
    remark_label.config(text="")

    set_button_state(clear_button, tk.DISABLED, '#1A2225')
```

```
    set_button_state(arrival_button, tk.NORMAL, '#00c853')
    set_button_state(departure_button, tk.NORMAL, '#1e88e5')

def pulang():
    jam = int(time.strftime("%H"))
    menit = int(time.strftime("%M"))
    lanjut = False
    video_capture = cv2.VideoCapture(0)
    time.sleep(2.0)
    start_time = time.time()
    #cek masker
    while True:
        _, frame = video_capture.read()
        (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)
        for (box, pred) in zip(locs, preds):
            (startX, startY, endX, endY) = box
            (mask, withoutMask) = pred
            if mask > withoutMask:
                label = "Buka masker"
                color = (0, 255, 255)
            else:
                lanjut = True
                break
            cv2.putText(frame, label, (startX, startY - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.9, color, 4)
            cv2.rectangle(frame, (startX, startY), (endX, endY), color, 7)
            remark_label.config(text="Buka masker", fg="black")
        if lanjut:
            break
        cv2.imshow("attendence system", frame)
        if time.time() - start_time > 4:
            current_time = time.strftime('%H.%M.%S')
            time_in_label.config(text=current_time)
            break
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    if not lanjut:
        video_capture.release()
        cv2.destroyAllWindows()
        set_button_state(departure_button, tk.DISABLED, '#0D4577')
        set_button_state(clear_button, tk.NORMAL, '#455a64')
```

```
            set_button_state(arrival_button, tk.DISABLED, '#16753D')
        #proses absen pulang
        else:
            start_time = time.time()
            isi = True
            while True:
                _, frame = video_capture.read()
                rgb_img = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                gray_img = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
                faces = haarcascade.detectMultiScale(gray_img, 1.3, 5)
                for x, y, w, h in faces:
                    img = rgb_img[y:y + h, x:x + w]
                    img = cv2.resize(img, (128, 128))
                    img = np.expand_dims(img, axis=0)
                    xpred = facenet.embeddings(img)
                    distances = np.linalg.norm(X - xpred, axis=1)
                    min_distance_index = np.argmin(distances)
                    min_distance = distances[min_distance_index]
                    predicted_label = Y[min_distance_index]
                    threshold = 0.6
                    if min_distance <= threshold:
                        cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 7)
                        cv2.putText(frame, predicted_label, (x, y - 10),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 4,cv2.LINE_AA)
                        name_label.config(text=predicted_label)
                        if jam < 16 or (jam == 16 and menit <= 59):
                            remark_label.config(text="Lebih awal", fg="red")
                            data_pulang = [predicted_label, time.strftime("%H.%M.%S")]
                        elif jam == 17 and menit <= 59:
                            remark_label.config(text="Tepat waktu", fg="blue")
                            data_pulang = [predicted_label, time.strftime("%H.%M.%S")]
                        else:
                            remark_label.config(text="Lembur", fg="#1E9C43")
                            data_pulang = [predicted_label, time.strftime("%H.%M.%S")]
                        baris = 0
                        for i in range(5, 15):
                            nama = spreadsheet.acell(f"C{i}").value
                            if data_pulang[0] == nama:
                                baris = i
                                break
                        cell_value = spreadsheet.acell(f"F{baris}").value
```

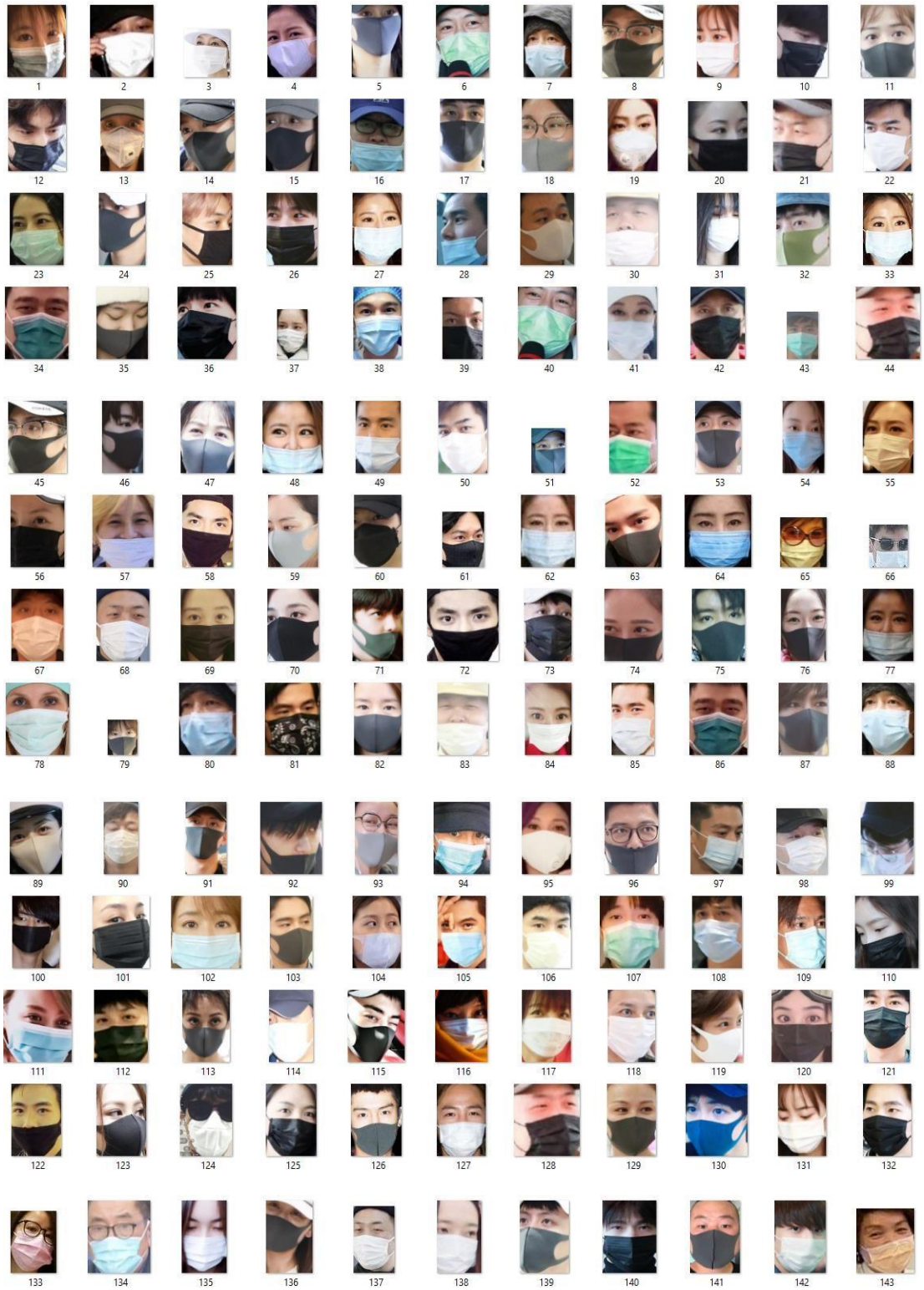
```
                datang = spreadsheet.acell(f"D{baris}").value
                format_waktu = "%H.%M.%S"
                datang = datetime.strptime(datang, format_waktu)
                pulang = time.strftime('%H.%M.%S')
                pulang = datetime.strptime(pulang, format_waktu)
                total = pulang - datang
                jam_total, sisa_detik = divmod(total.seconds, 3600)
                menit_total, detik_total = divmod(sisa_detik, 60)
                total_jam_kerja = f"{jam_total:02}.{menit_total:02}.{detik_total:02}"
                if jam_total < 8:
                    keterangan = "Lebih awal"
                elif jam_total == 8:
                    keterangan = "Tepat waktu"
                else:
                    keterangan = "Lembur"
                if cell_value == "None":
                    spreadsheet.update([[data_pulang[1], total_jam_kerja, keterangan]],
f"F{baris}:H{baris}")
                    isi = False
                elif cell_value != "None" and isi:
                    remark_label.config(text="Sudah pulang", fg="black")
            else:
                cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 7)
                cv2.putText(frame, "Tidak dikenali", (x, y - 10),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 4, cv2.LINE_AA)
                remark_label.config(text="Tidak dikenali", fg="black")
        current_time = time.strftime('%H.%M.%S')

        time_in_label.config(text=current_time)
        cv2.imshow("attendence system", frame)
        if time.time() - start_time > 4:
            break
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    video_capture.release()
    cv2.destroyAllWindows()
    set_button_state(departure_button, tk.DISABLED, '#0D4577')
    set_button_state(clear_button, tk.NORMAL, '#455a64')
    set_button_state(arrival_button, tk.DISABLED, '#16753D')

def detect_and_predict_mask(frame, faceNet, maskNet):
```

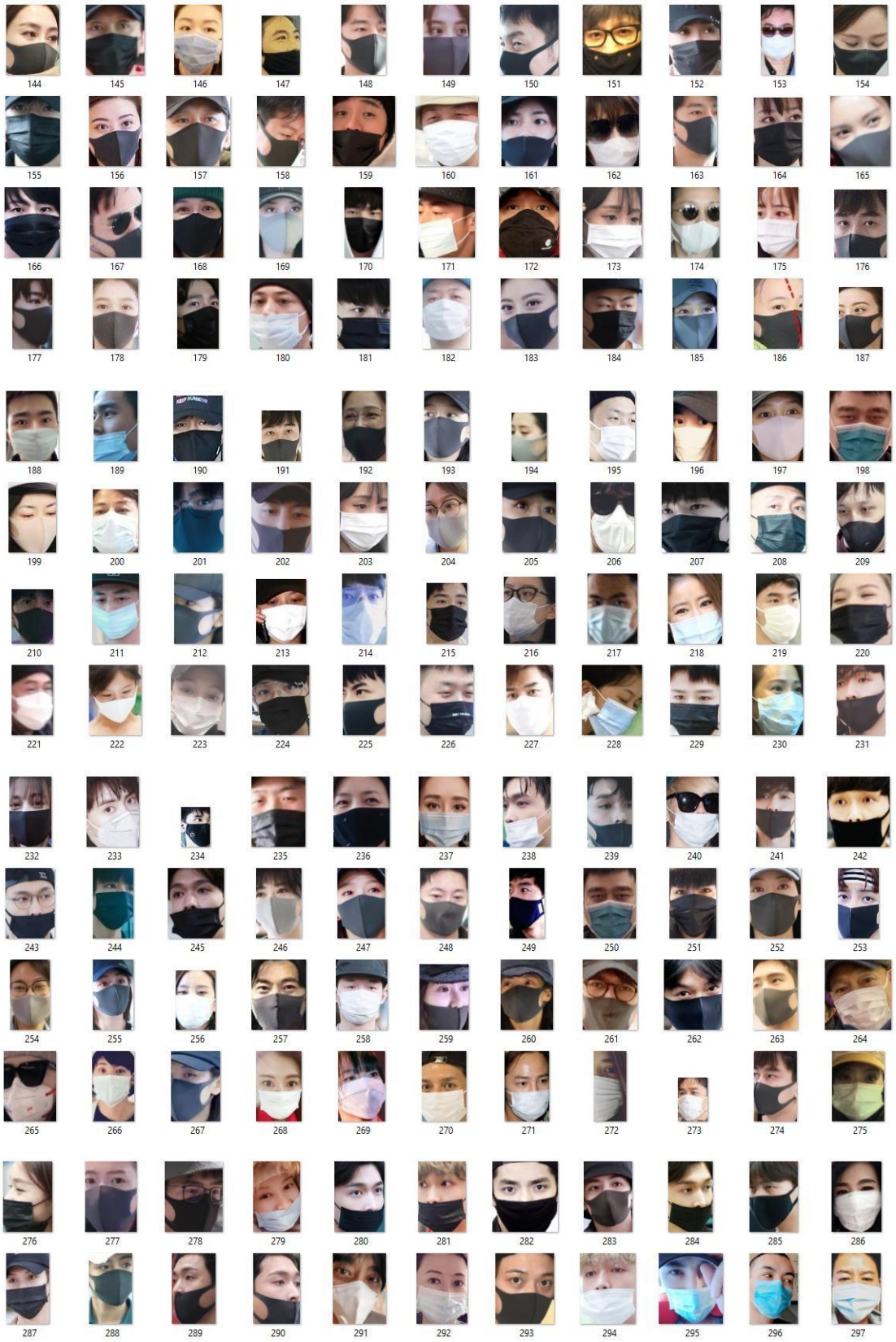
```
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (300, 300), (104.0, 177.0, 123.0))
    faceNet.setInput(blob)
    detections = faceNet.forward()
    faces = []
    locs = []
    preds = []
    for i in range(0, detections.shape[2]):
        confidence = detections[0, 0, i, 2]
        if confidence > args["confidence"]:
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")
            (startX, startY) = (max(0, startX), max(0, startY))
            (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

            face = frame[startY:endY, startX:endX]
            face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
            face = cv2.resize(face, (224, 224))
            face = img_to_array(face)
            face = preprocess_input(face)
            face = np.expand_dims(face, axis=0)
            faces.append(face)
            locs.append((startX, startY, endX, endY))
    if len(faces) > 0:
        preds = maskNet.predict(faces)
    return (locs, preds)

#Memuat embeddings
facenet = FaceNet()
faces_embeddings = np.load("faces_embeddings_done_4classes.npz")
X = faces_embeddings['arr_0']
Y = faces_embeddings['arr_1']

encoder = LabelEncoder()
encoder.fit(Y)
haarcascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")

ap = argparse.ArgumentParser()
ap.add_argument("-f", "--face", type=str, default="face_detector", help="path to face
detector model directory")
ap.add_argument("-m", "--model", type=str, default="mask_detector.model",
```

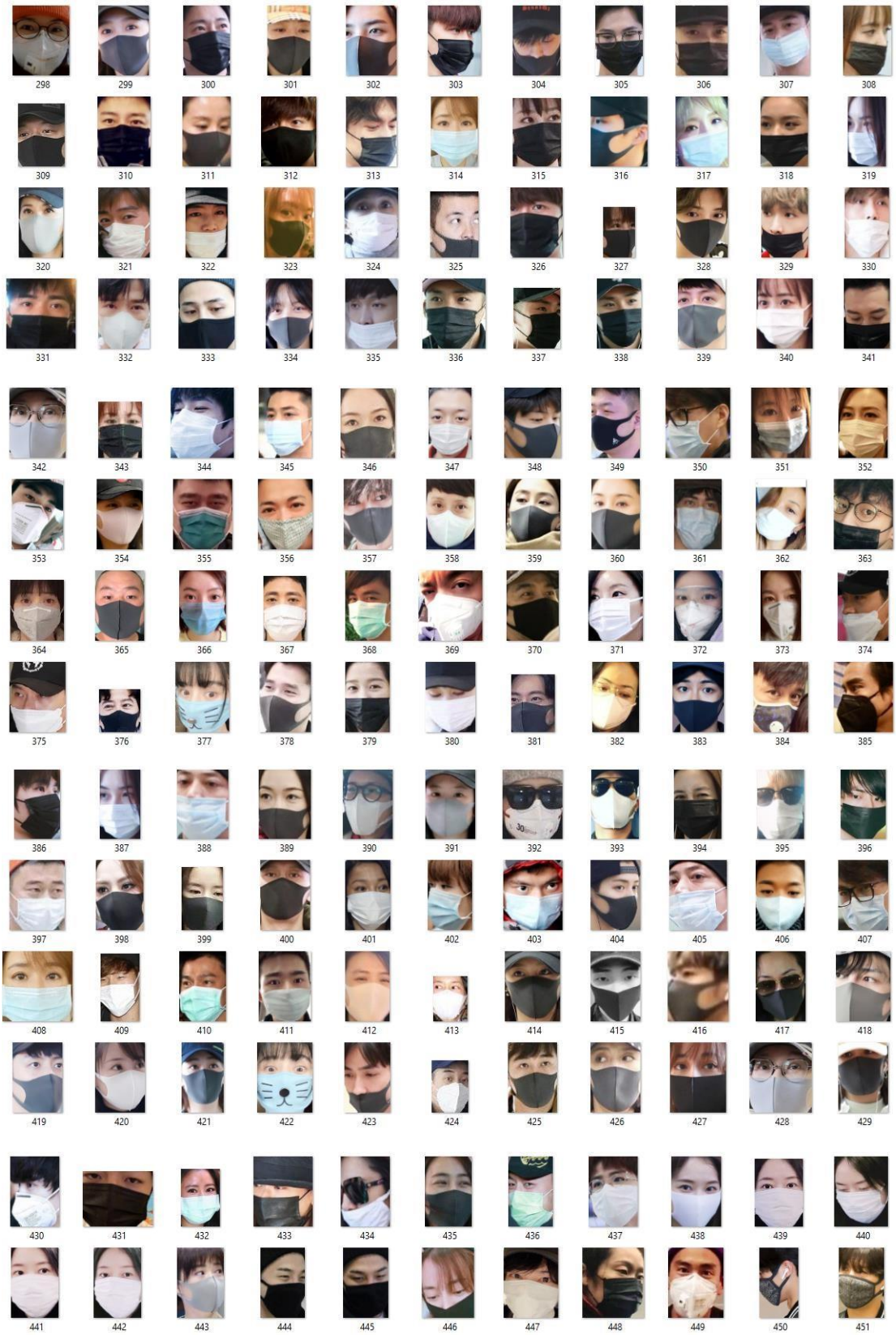
```
help="path to trained face mask detector model")
ap.add_argument("-c", "--confidence", type=float, default=0.5, help="minimum
probability to filter weak detections")
args = vars(ap.parse_args())

prototxtPath = os.path.sep.join([args["face"], "deploy.prototxt"])
weightsPath = os.path.sep.join([args["face"],
"res10_300x300_ssd_iter_140000.caffemodel"])
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)
maskNet = load_model(args["model"])

scope = ['https://www.googleapis.com/auth/spreadsheets',
'https://www.googleapis.com/auth/drive.file',
'https://www.googleapis.com/auth/drive']
credentials = ServiceAccountCredentials.from_json_keyfile_name('hadir.json', scope)
gc = gspread.authorize(credentials)
spreadsheet_name = 'data kehadiran'
spreadsheet = gc.open(spreadsheet_name).sheet1

day_string = terjemahan_hari()
current_date = datetime.now().strftime("%d %B %Y")
tanggal = f"{day_string}, {current_date}"
spreadsheet.update_cell(3, 3, tanggal)

root = tk.Tk()
root.title("Absen Pegawai")
root.geometry("800x600")
root.configure(bg='#000000')

title_font = font.Font(family='Helvetica', size=40, weight='bold')
label_font = font.Font(family='Helvetica', size=18)
time_font = font.Font(family='Helvetica', size=40)
date_font = font.Font(family='Helvetica', size=24)
button_font = font.Font(family='Helvetica', size=14, weight='bold')

header_frame = tk.Frame(root, bg='#383838', height=50)
header_frame.pack(fill='x')
title_label = tk.Label(header_frame, text="Absen", font=title_font, bg='#383838',
fg='white')
title_label.pack(pady=10)
time_label = tk.Label(root, text="", font=time_font, bg='#000000', fg='white')
```

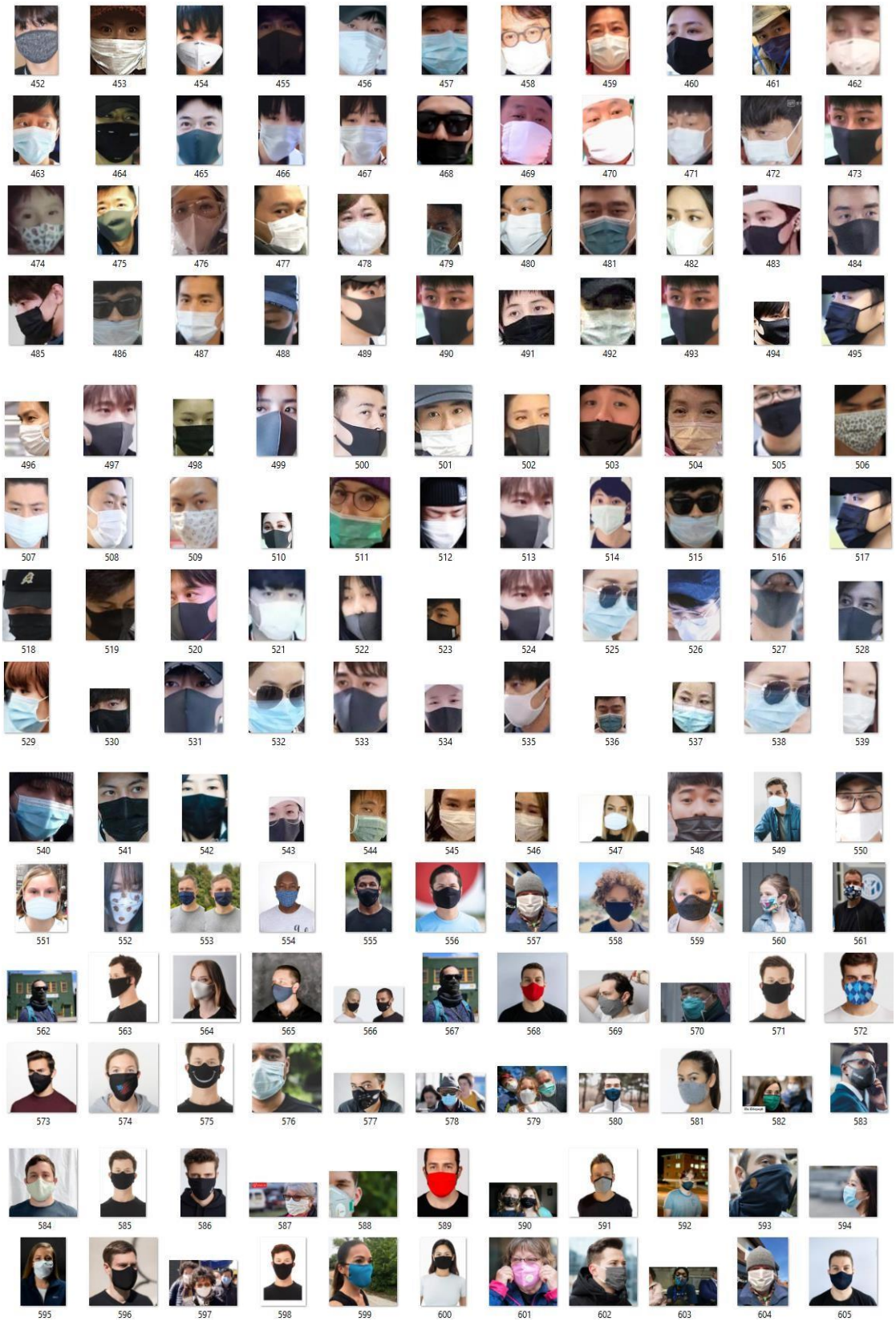
```
time_label.pack(pady=(20, 5))
date_label = tk.Label(root, text="", font=date_font, bg='#000000', fg='white')
date_label.pack()
input_frame = tk.Frame(root, bg='#000000')
input_frame.pack(pady=20)
name_label_text = tk.Label(input_frame, text="Nama", font=label_font,
bg='#000000', fg='white')
name_label_text.grid(row=0, column=0, sticky='w', padx=20, pady=(0, 20))
name_label = tk.Label(input_frame, text="", font=label_font, bg='#FFFFFF',
fg='black', width=30, anchor='w')
name_label.grid(row=0, column=1, padx=20, pady=(0, 20))
time_in_label_text = tk.Label(input_frame, text="Waktu Absen", font=label_font,
bg='#000000', fg='white')
time_in_label_text.grid(row=1, column=0, sticky='w', padx=20, pady=(0, 20))
time_in_label = tk.Label(input_frame, text="", font=label_font, bg='#FFFFFF',
fg='black', width=30, anchor='w')
time_in_label.grid(row=1, column=1, padx=20, pady=(0, 20))
remark_label_text = tk.Label(input_frame, text="Keterangan", font=label_font,
bg='#000000', fg='white')
remark_label_text.grid(row=2, column=0, sticky='w', padx=20, pady=(0, 20))
remark_label = tk.Label(input_frame, text="", font=label_font, bg='#FFFFFF',
fg='black', width=30, anchor='w')
remark_label.grid(row=2, column=1, padx=20, pady=(0, 20))

button_frame = tk.Frame(root, bg='#000000')
button_frame.pack(pady=20)

arrival_button = tk.Button(button_frame, text="Datang", font=button_font,
bg='#00c853', fg='white', command=datang, height=1, width=13)
arrival_button.grid(row=0, column=0, padx=(0, 28), sticky='w')
clear_button = tk.Button(button_frame, text="Bersihkan", font=button_font,
bg='#1A2225', fg='white', command=bersihkan, height=1, width=13, state =
tk.DISABLED)
clear_button.grid(row=0, column=1, padx=28)
departure_button = tk.Button(button_frame, text="Pulang", font=button_font,
bg='#1e88e5', fg='white', command=pulang, height=1, width=13)
departure_button.grid(row=0, column=2, padx=(28, 0), sticky='e')

update_time()
update_date()
root.mainloop()
```

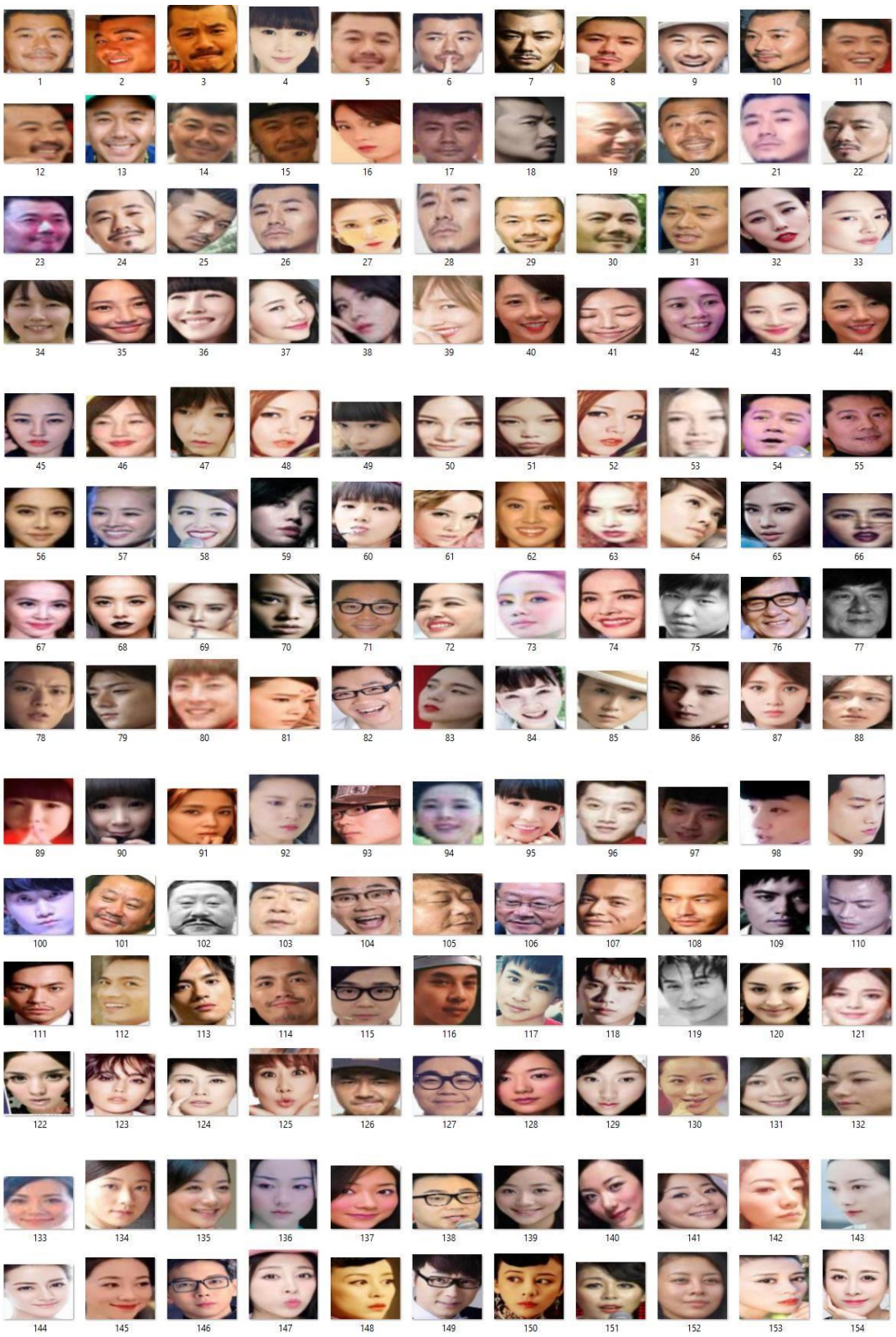
Lampiran 4 Dataset Pelatihan Identifikasi Masker

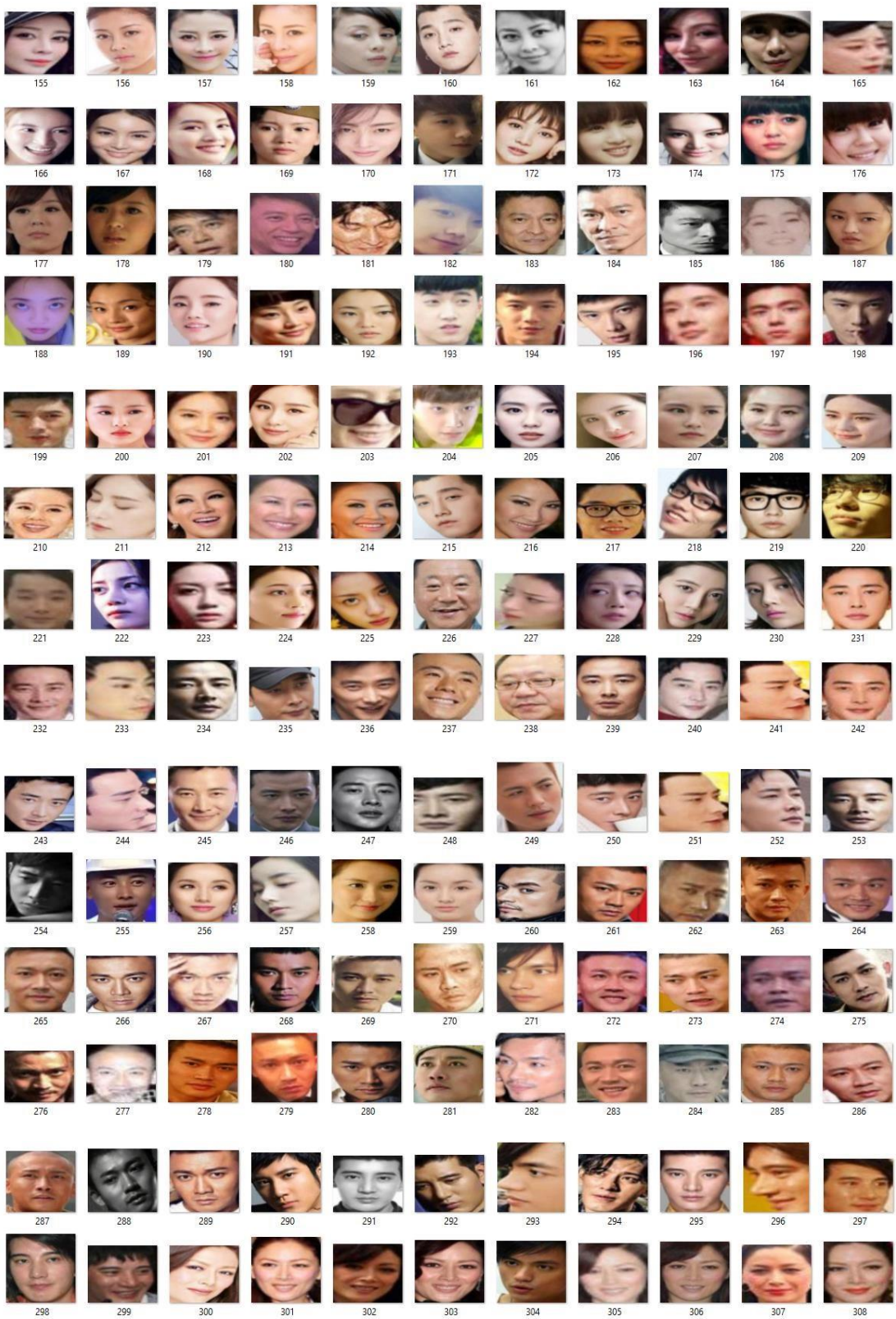


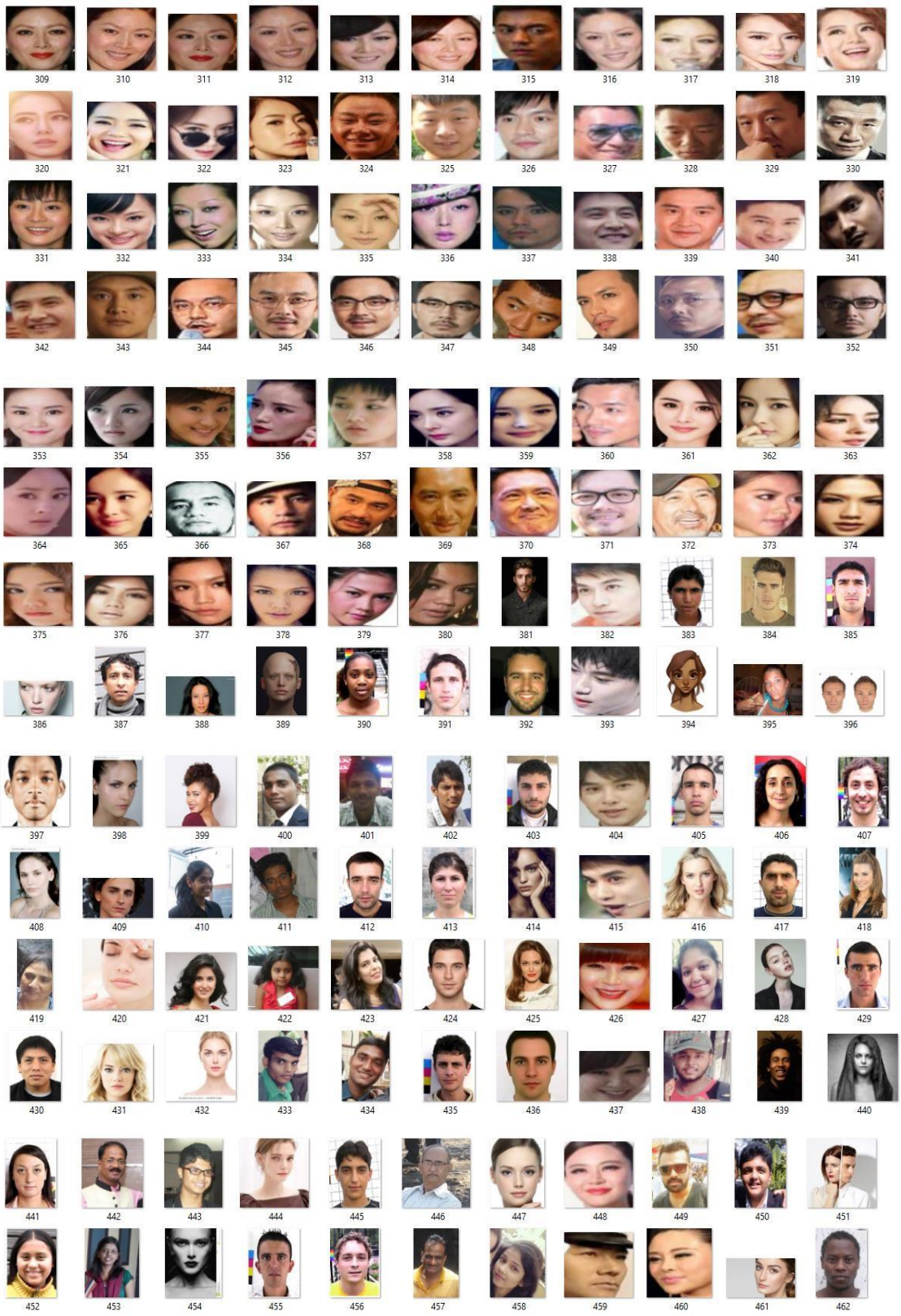

144 145 146 147 148 149 150 151 152 153 154
155 156 157 158 159 160 161 162 163 164 165
166 167 168 169 170 171 172 173 174 175 176
177 178 179 180 181 182 183 184 185 186 187
188 189 190 191 192 193 194 195 196 197 198
199 200 201 202 203 204 205 206 207 208 209
210 211 212 213 214 215 216 217 218 219 220
221 222 223 224 225 226 227 228 229 230 231
232 233 234 235 236 237 238 239 240 241 242
243 244 245 246 247 248 249 250 251 252 253
254 255 256 257 258 259 260 261 262 263 264
265 266 267 268 269 270 271 272 273 274 275
276 277 278 279 280 281 282 283 284 285 286
287 288 289 290 291 292 293 294 295 296 297

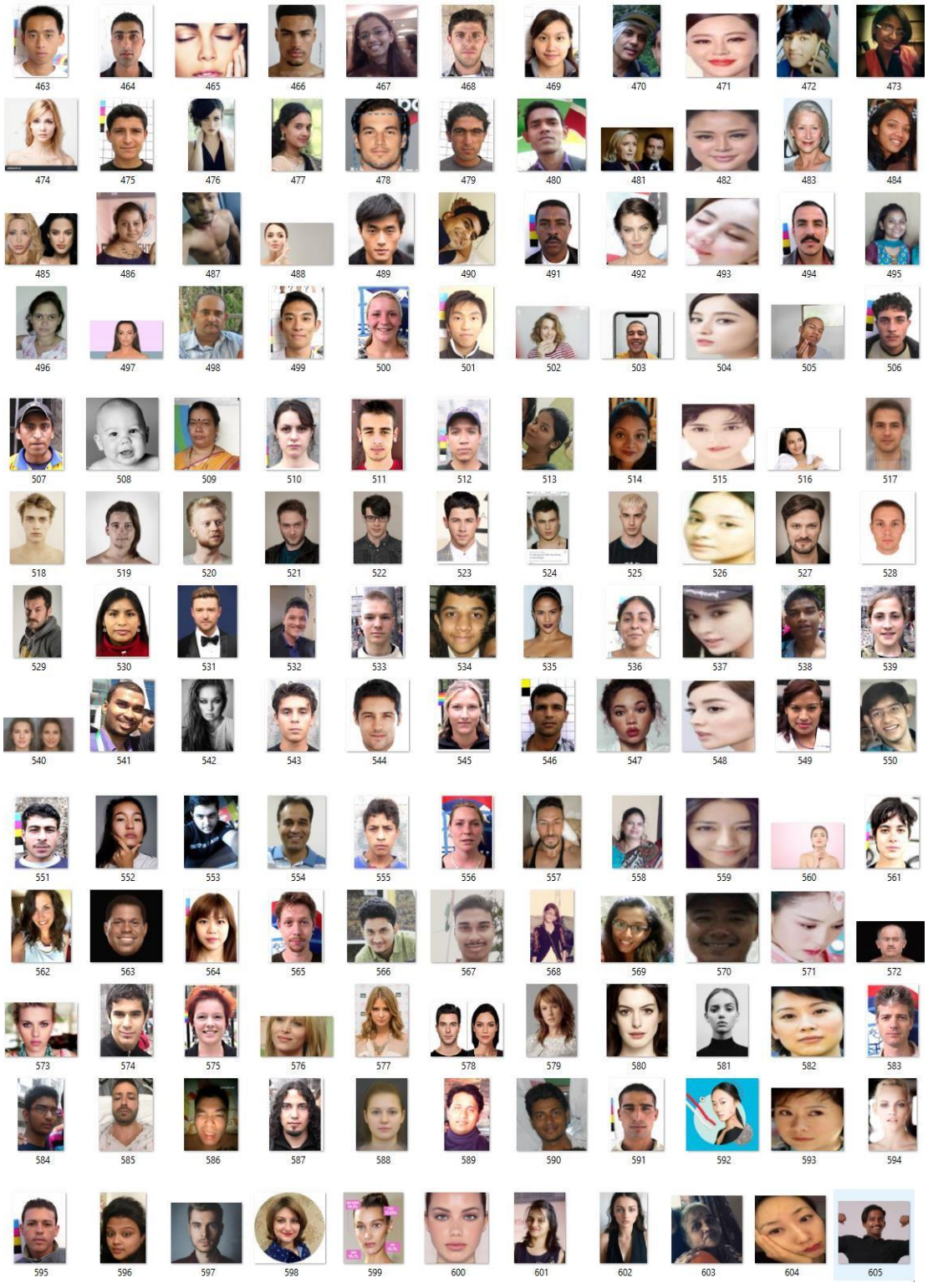

298 299 300 301 302 303 304 305 306 307 308
309 310 311 312 313 314 315 316 317 318 319
320 321 322 323 324 325 326 327 328 329 330
331 332 333 334 335 336 337 338 339 340 341
342 343 344 345 346 347 348 349 350 351 352
353 354 355 356 357 358 359 360 361 362 363
364 365 366 367 368 369 370 371 372 373 374
375 376 377 378 379 380 381 382 383 384 385
386 387 388 389 390 391 392 393 394 395 396
397 398 399 400 401 402 403 404 405 406 407
408 409 410 411 412 413 414 415 416 417 418
419 420 421 422 423 424 425 426 427 428 429
430 431 432 433 434 435 436 437 438 439 440
441 442 443 444 445 446 447 448 449 450 451


452 453 454 455 456 457 458 459 460 461 462
463 464 465 466 467 468 469 470 471 472 473
474 475 476 477 478 479 480 481 482 483 484
485 486 487 488 489 490 491 492 493 494 495
496 497 498 499 500 501 502 503 504 505 506
507 508 509 510 511 512 513 514 515 516 517
518 519 520 521 522 523 524 525 526 527 528
529 530 531 532 533 534 535 536 537 538 539
540 541 542 543 544 545 546 547 548 549 550
551 552 553 554 555 556 557 558 559 560 561
562 563 564 565 566 567 568 569 570 571 572
573 574 575 576 577 578 579 580 581 582 583
584 585 586 587 588 589 590 591 592 593 594
595 596 597 598 599 600 601 602 603 604 605

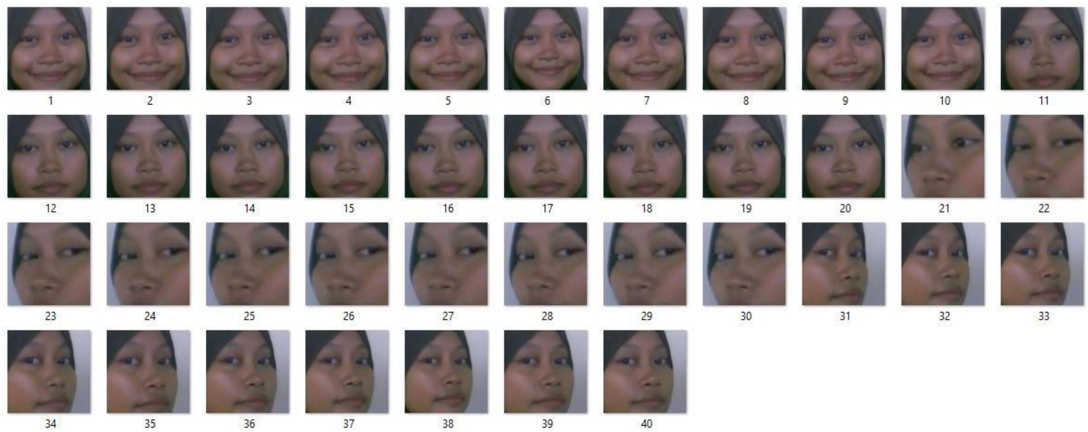



155 156 157 158 159 160 161 162 163 164 165
166 167 168 169 170 171 172 173 174 175 176
177 178 179 180 181 182 183 184 185 186 187
188 189 190 191 192 193 194 195 196 197 198
199 200 201 202 203 204 205 206 207 208 209
210 211 212 213 214 215 216 217 218 219 220
221 222 223 224 225 226 227 228 229 230 231
232 233 234 235 236 237 238 239 240 241 242
243 244 245 246 247 248 249 250 251 252 253
254 255 256 257 258 259 260 261 262 263 264
265 266 267 268 269 270 271 272 273 274 275
276 277 278 279 280 281 282 283 284 285 286
287 288 289 290 291 292 293 294 295 296 297
298 299 300 301 302 303 304 305 306 307 308

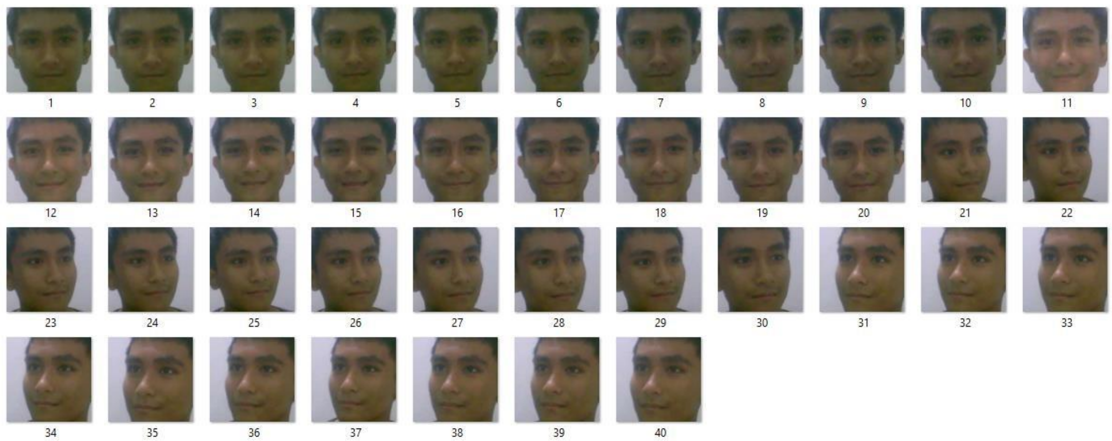

309 310 311 312 313 314 315 316 317 318 319
320 321 322 323 324 325 326 327 328 329 330
331 332 333 334 335 336 337 338 339 340 341
342 343 344 345 346 347 348 349 350 351 352
353 354 355 356 357 358 359 360 361 362 363
364 365 366 367 368 369 370 371 372 373 374
375 376 377 378 379 380 381 382 383 384 385
386 387 388 389 390 391 392 393 394 395 396
397 398 399 400 401 402 403 404 405 406 407
408 409 410 411 412 413 414 415 416 417 418
419 420 421 422 423 424 425 426 427 428 429
430 431 432 433 434 435 436 437 438 439 440
441 442 443 444 445 446 447 448 449 450 451
452 453 454 455 456 457 458 459 460 461 462


463 464 465 466 467 468 469 470 471 472 473
474 475 476 477 478 479 480 481 482 483 484
485 486 487 488 489 490 491 492 493 494 495
496 497 498 499 500 501 502 503 504 505 506
507 508 509 510 511 512 513 514 515 516 517
518 519 520 521 522 523 524 525 526 527 528
529 530 531 532 533 534 535 536 537 538 539
540 541 542 543 544 545 546 547 548 549 550
551 552 553 554 555 556 557 558 559 560 561
562 563 564 565 566 567 568 569 570 571 572
573 574 575 576 577 578 579 580 581 582 583
584 585 586 587 588 589 590 591 592 593 594
595 596 597 598 599 600 601 602 603 604 605

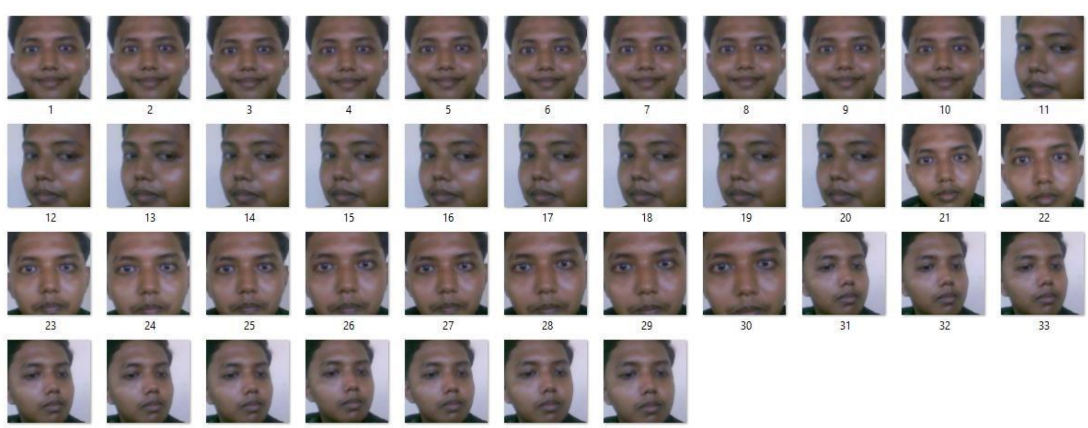
Lampiran 5 Dataset Pelatihan Pengenalan Wajah



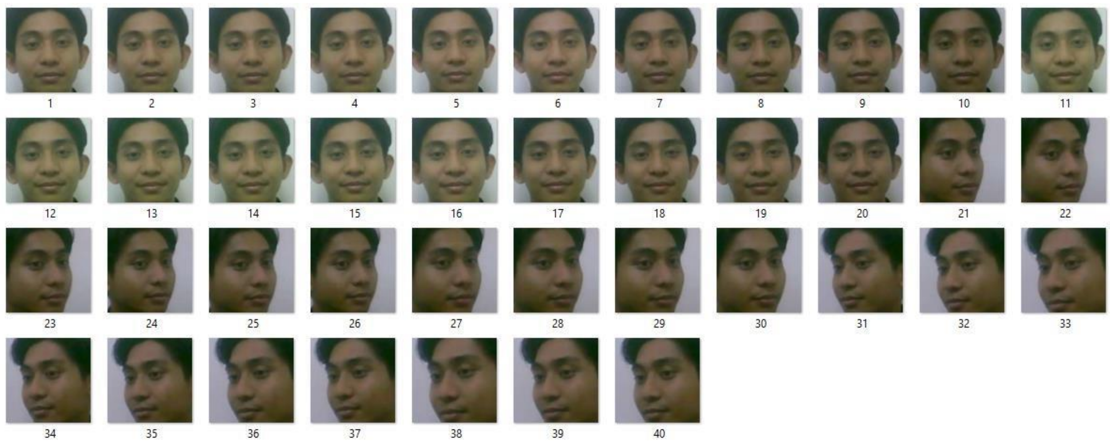

23 24 25 26 27 28 29 30 31 32 33
34 35 36 37 38 39 40

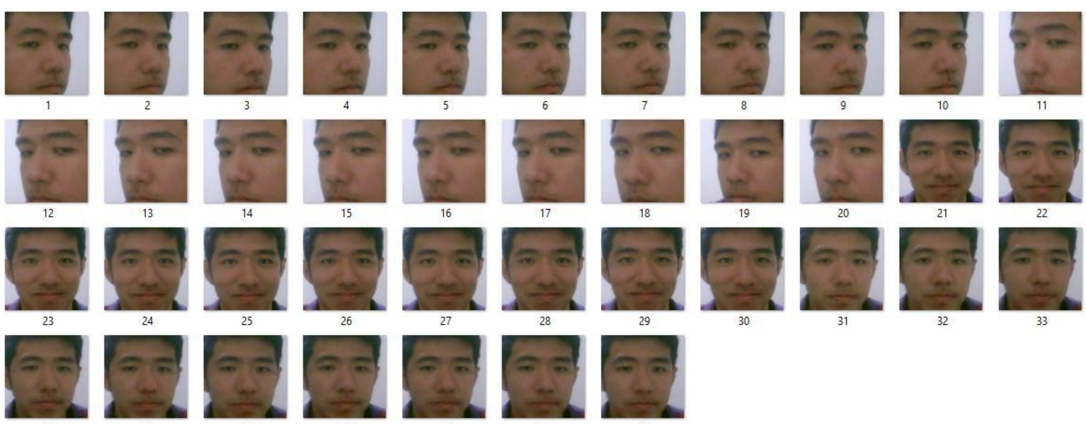

1 2 3 4 5 6 7 8 9 10 11
12 13 14 15 16 17 18 19 20 21 22
23 24 25 26 27 28 29 30 31 32 33
34 35 36 37 38 39 40


1 2 3 4 5 6 7 8 9 10 11
12 13 14 15 16 17 18 19 20 21 22
23 24 25 26 27 28 29 30 31 32 33
34 35 36 37 38 39 40

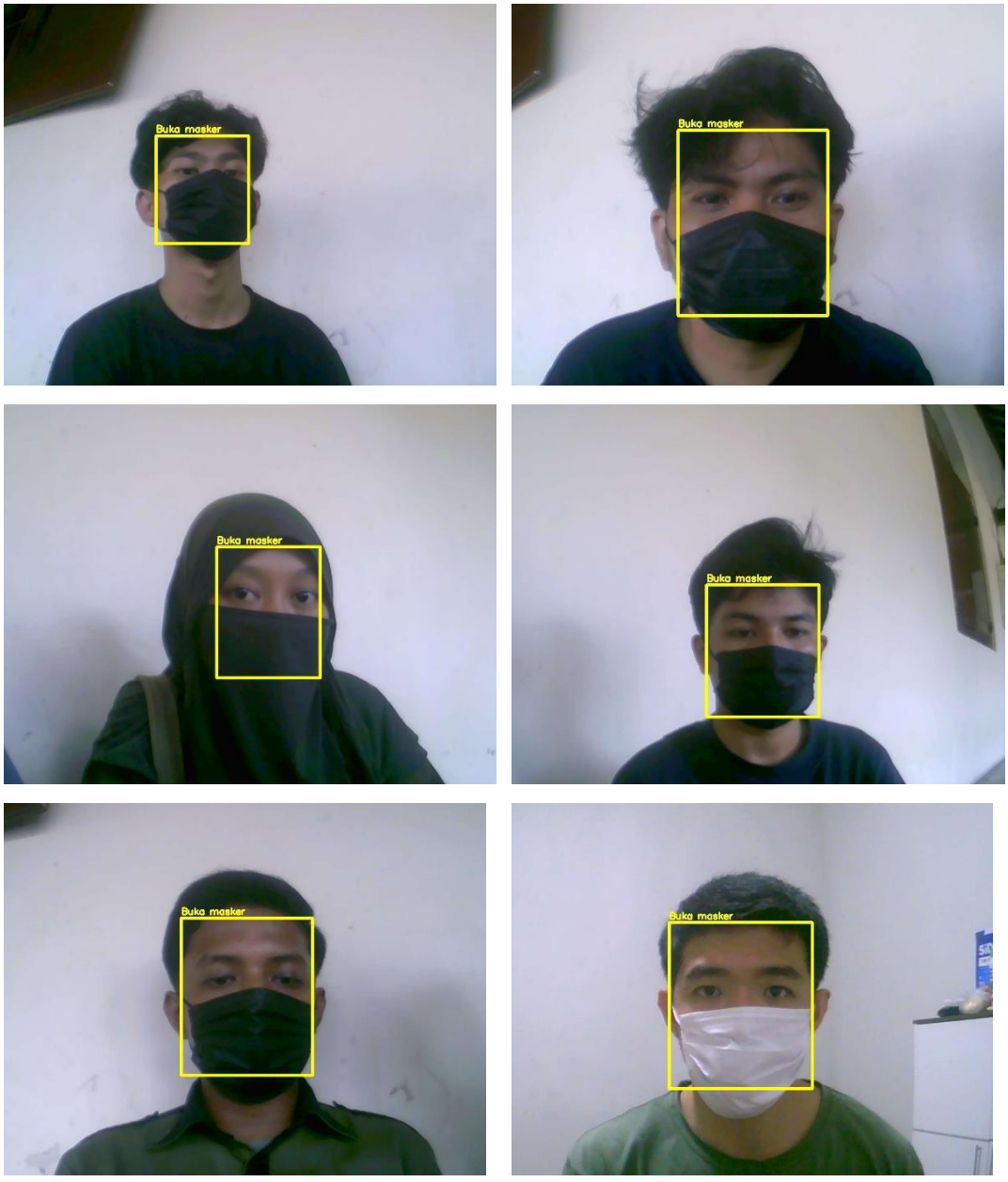

1 2 3 4 5 6 7 8 9 10 11
12 13 14 15 16 17 18 19 20 21 22
23 24 25 26 27 28 29 30 31 32 33
34 35 36 37 38 39 40

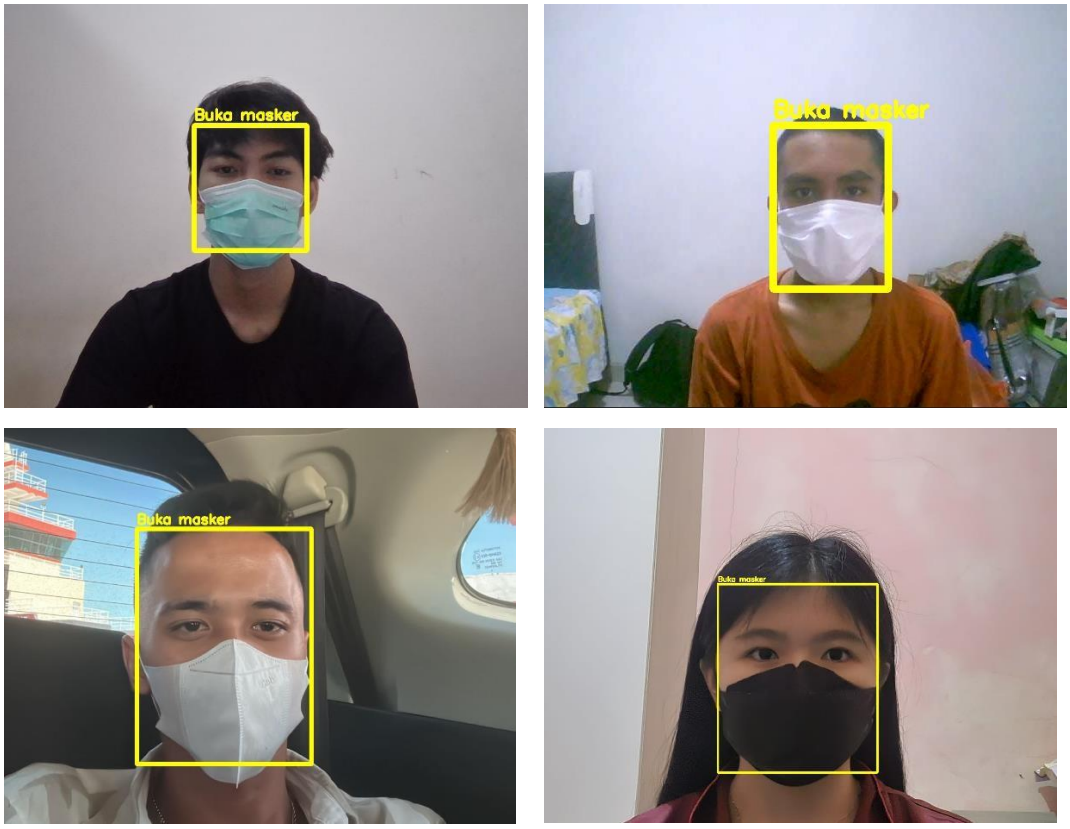

1 2 3 4 5 6 7 8 9 10 11
12 13 14 15 16 17 18 19 20 21 22
23 24 25 26 27 28 29 30 31 32 33
34 35 36 37 38 39 40


1 2 3 4 5 6 7 8 9 10 11
12 13 14 15 16 17 18 19 20 21 22
23 24 25 26 27 28 29 30 31 32 33
34 35 36 37 38 39 40

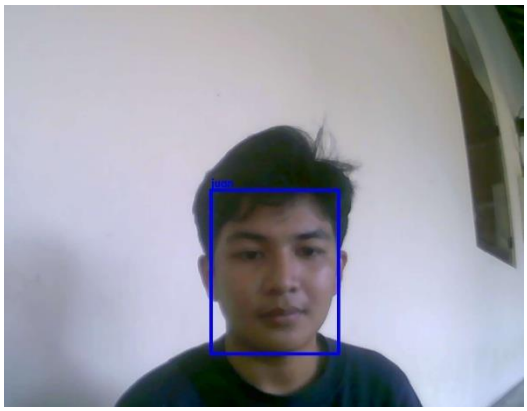

1 2 3 4 5 6 7 8 9 10 11
12 13 14 15 16 17 18 19 20 21 22
23 24 25 26 27 28 29 30 31 32 33
34 35 36 37 38 39 40

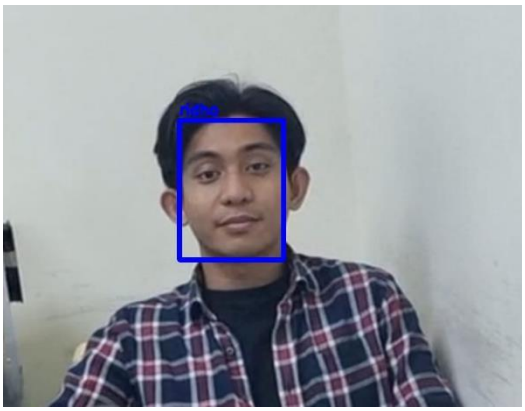
Lampiran 6 Pengujian identifikasi masker

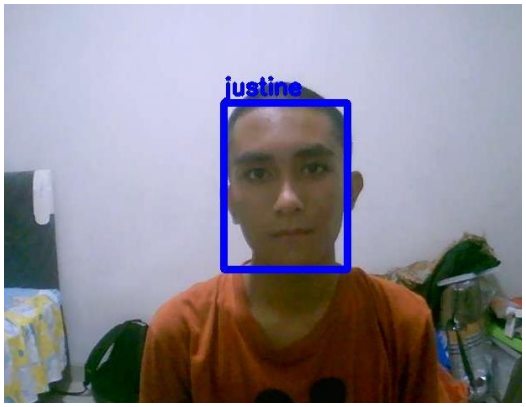


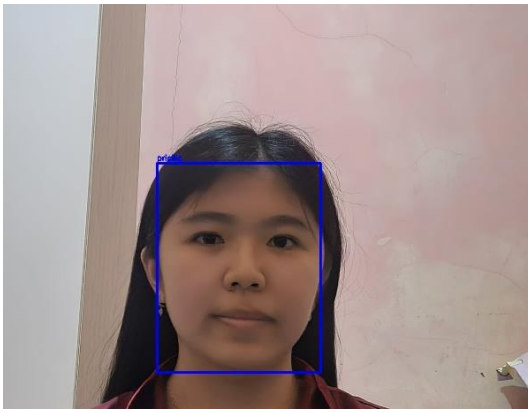

Buka masker
Buka masker
Buka masker
Buka masker

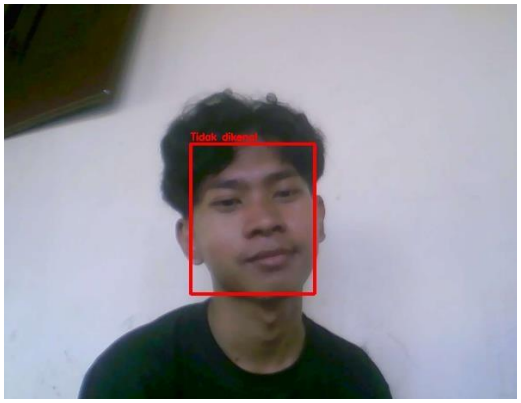
Lampiran 7 Pengujian Pengenalan Wajah

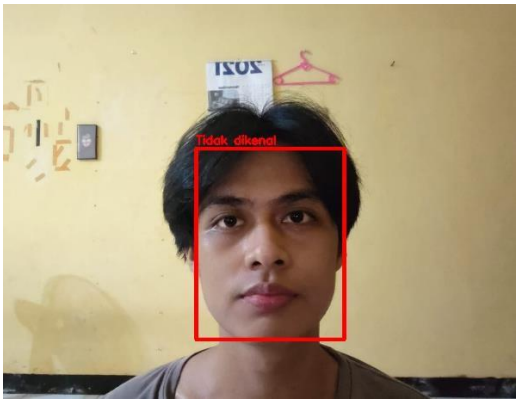

justine

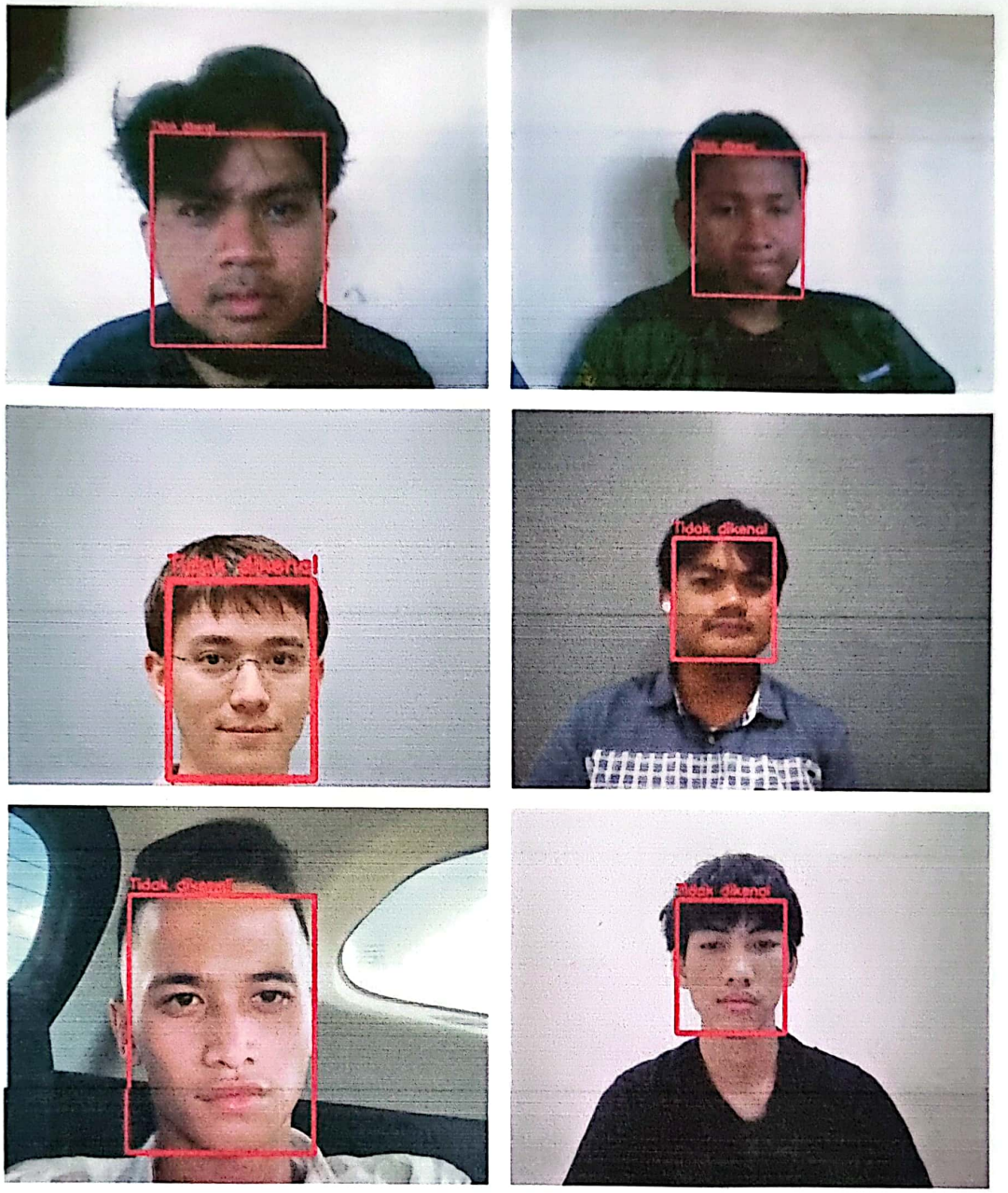

justine

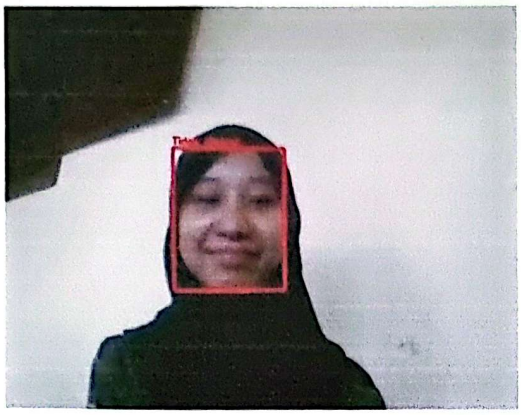

justine

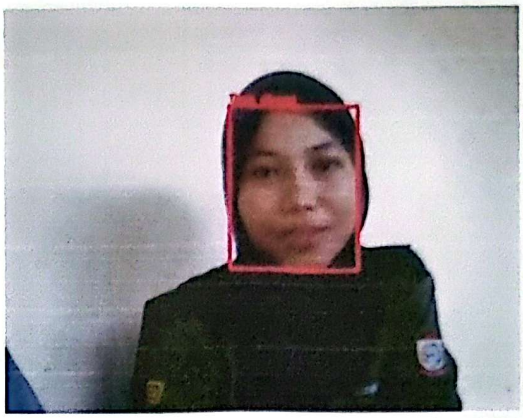

justine

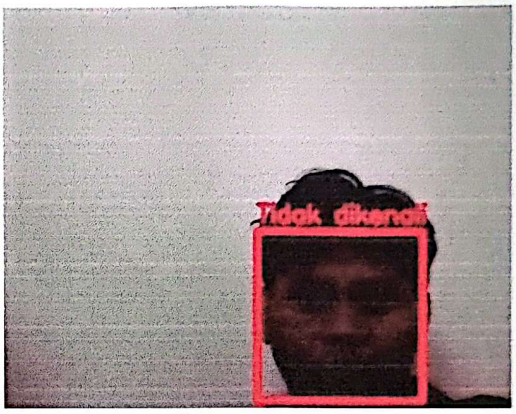

Tidak dikenal

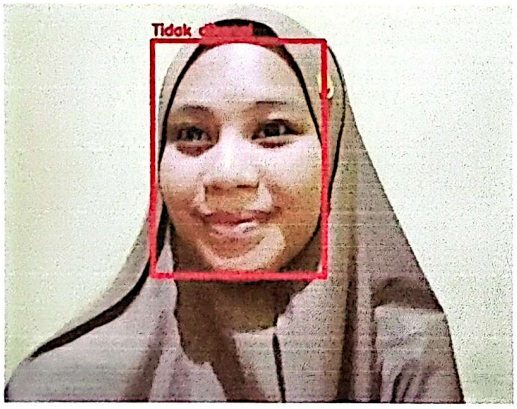

Tidak dikenal


Tidak dikenal
Tidak dikenal
Tidak dikenal
Tidak dikenal
Tidak dikenal
Tidak dikenal






Tidak dikenali


Tidak dikenali


Tidak dikenali