DAFTAR PUSTAKA

A. A Sukmandhani, I. Sutedja, "Face Recognition Method for Online Exams" 2019 *International Conference on Information Management and Technology (ICIMTech), doi: 10.1109/ICIMTech.2019.8843831*

Abdul Kadir. 2017. *Pemrograman Arduino & Android Menggunakan App Inventor*. Jakarta: PT Alex Media Komputindo.

Akbar, S. 2013. *Instrumen Perangkat Pembelajaran*. Bandung: Remaja Rosdakarya.

Basuki, Achmad. 2005. Pengolahan Citra Digital Menggunakan Visual Basic. Graha. Ilmu. Yogyakarta.

Gonzales, R.C.; Woods, R.E; Eddins, S.L. 2004. "Digital Image Processing Using MATLAB. *Pearson LPE*"

H. Lami, S. I. Pella (2019), "Implementasi Deteksi Dan Pengenalan Wajah Pada Sistem Ujian Online Menggunakan Metode Deep Learning Berbasis Raspberry Pi", *Jurnal Media Elektro Undana.* *https://ejurnal.undana.ac.id/jme/article/view/1394*

H. Sajati, A. Ayuningsi & D Kholistyanto (2017),"Penerapan Eigenface Untuk Computer Based Test (Cbt) Penerimaan Mahasiswa Baru Sekolah Tinggi Teknologi Adisutjipto" *Jurnal STTA.* *http://ejournals.stta.ac.id/index.php/compiler/article/view/228*

Intania dkk.. 2012. *Sekali Baca Langsung Inget Mengupas Lengkap All About Android*. Jakarta: Kuncikom.

Kulkarni, A. D. (2001). Computer Vision and Fuzzy Neural Sytems. New Jersey: Prentice Hall PTR.

Luka Dulčić (2019). Face Recognition with FaceNet and MTCNN, 2019, Available: https://arsfutura.com/magazine/face-recognitionwith-facenet-and-mtcnn/. [diakses 23 oktober 2021]

Mika Tandililing 2016, "Aplikasi Pengenalan Wajah Untuk Validasi Peserta Ujian Online Menggunakan Metode Haar Cascade Dan Eigen Face Vector", *Jurnal STMIK Profesional.* *http://jurnal.stmikprofesional.ac.id/index.php/siberpro/article/view/96*

ingsih, M. 2017. *Pengembangan Dan Analisis Media Pembelajaran ulasi Merakit Komputer Berbasis Desktop*. Pengembangan Dan Analisis dia, 9.

Nixon, M.S. and Aguado, A.S. (2019) Feature extraction and image processing for computer vision, *Feature Extraction and Image Processing for Computer Vision*. Available at: https://doi.org/10.1016/C2017-0-02153-5.

Novrianti. (2014). Pengembangan Computer Based Testing (CBT) sebagai Alternatif Teknik Penilaian Hasil Belajar. Jurnal Lentera Pendidikan. 7 (1), 34-42.

Putu Putra Yana Wardana1, I A Dwi Giriantari2, Made Sudarma3, "aplikasi verifikasi wajah untuk absensi pada platform android dengan menggunakan algoritma fisherface", *Teknologi Elektro, Vol. 15, No.2, 2016*

Rahim, M.A. 2013. Perancangan Aplikasi Pengenalan Wajah Menggunakan Metode Viola Jonnes.Medan: STMIK Budi Dharma.

Schroff F, Kalenichenko D, Philbin J, "*FaceNet: A Unified Embedding for Face Recognition and Clustering" 2015 IEEE Conference on Computer Vision and Pattern Recognation (CVPR), doi: 10.1109/CVPR.2015.7298682*

Sena Samuel (2017). *Pengenalan Deep Learning Part 7: Convolutional Neural Network (CNN)* kses 22 Desember 2023 dari https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94

SHI Jun, LI Hui,GU Hang, & Zho Li-dong, "Research and Development of Intelligent Online Examination Monitoring System" 2017 *International Conference on Computer Science and Education (ICCSE), doi:10.1109/ICCSE.2017.8085463*

Sherief, S. 2014. Buku Pintar Gadget Android Untuk Pemula. Jakarta: Kunci Komunikasi.

S. Prathis, A. Narayanan, and K. Bijlani, "*An Intelligent System For Online Exam Monitoring" 2016 International Conference on Information Science (ICIS), doi: 10.1109/INFOSCI.2016.7845315*

Suprianto, D. 2013. Sistem Pengenalan Wajah Secara Real-Time, dengan Adaboost, Eigenface PCA & MySql

Williams, B.K., & Sawyer, S.C. 2011. Using Information Technology: A Practical Introduction to Computers & Communications. (9th edition). New York: McGraw-Hill.

W. S. Eka Putra, "Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) pada Caltech 101," J. Tek. ITS, vol. 5, no. 1, 2016, doi: 12962/j23373539.v5i1.15696.

.H., Kriegman, D.J. and Ahuja, N. (2002) 'Detecting faces in images: A vey', *IEEE Transactions on Pattern Analysis and Machine Intelligence, 1), pp. 34–58*. Available at: https://doi.org/10.1109/34.982883.

LAMPIRAN

Lampiran 1 *Sorce Code Training* Wajah *Python*

```
import os
import argparse
import joblib
import numpy as np
from PIL import Image
from torchvision import transforms, datasets
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn import metrics
from face_recognition import preprocessing, FaceFeaturesExtractor,
FaceRecogniser

MODEL_DIR_PATH = 'model'

def parse_args():
    parser = argparse.ArgumentParser(
        description='Script for training Face Recognition model.
You can either give path to dataset or provide path '
                    'to pre-generated embeddings, labels and
class_to_idx. You can pre-generate this with '
                    'util/generate_embeddings.py script.')
    parser.add_argument('-d', '--dataset-path', help='Path to
folder with images.')
    parser.add_argument('-e', '--embeddings-path', help='Path to
file with embeddings.')
    parser.add_argument('-l', '--labels-path', help='Path to file
with labels.')
    parser.add_argument('-c', '--class-to-idx-path', help='Path to
pickled class_to_idx dict.')
    parser.add_argument('--grid-search', action='store_true',
                        help='If this option is enabled, grid
search will be performed to estimate C parameter of '
                             'Logistic Regression classifier. In
order to use this option you have to have at least '
                             '3 examples of every class in your
dataset. It is recommended to enable this option.')
    return parser.parse_args()


def dataset_to_embeddings(dataset, features_extractor):
    transform = transforms.Compose([
        preprocessing.ExifOrientationNormalize(),
        transforms.Resize(1024)
    ])

    embeddings = []
        els = []
        img_path, label in dataset.samples:
        print(img_path)
        _, embedding =
    s_extractor(transform(Image.open(img_path).convert('RGB')))
        if embedding is None:
```

```
            print("Could not find face on {}".format(img_path))
            continue
        if embedding.shape[0] > 1:
            print("Multiple faces detected for {}, taking one with
highest probability".format(img_path))
            embedding = embedding[0, :]
        embeddings.append(embedding.flatten())
        labels.append(label)

    return np.stack(embeddings), labels


def load_data(args, features_extractor):
    if args.embeddings_path:
        return np.loadtxt(args.embeddings_path), \
               np.loadtxt(args.labels_path, dtype='str').tolist(),
\
               joblib.load(args.class_to_idx_path)

    dataset = datasets.ImageFolder(args.dataset_path)
    embeddings, labels = dataset_to_embeddings(dataset,
features_extractor)
    return embeddings, labels, dataset.class_to_idx


def train(args, embeddings, labels):
    softmax = LogisticRegression(solver='lbfgs',
multi_class='multinomial', C=10, max_iter=10000)
    if args.grid_search:
        clf = GridSearchCV(
            estimator=softmax,
            param_grid={'C': [0.001, 0.01, 0.1, 1, 10, 100,
1000]},
            cv=3
        )
    else:
        clf = softmax
    clf.fit(embeddings, labels)

    return clf.best_estimator_ if args.grid_search else clf


def main():
    args = parse_args()

    features_extractor = FaceFeaturesExtractor()
    embeddings, labels, class_to_idx = load_data(args,
features_extractor)
    clf = train(args, embeddings, labels)

    _to_class = {v: k for k, v in class_to_idx.items()}

    get_names = map(lambda i: i[1],
    idx_to_class.items(), key=lambda i: i[0]))
    nt(metrics.classification_report(labels,
    dict(embeddings), target_names=list(target_names)))
```

```
    if not os.path.isdir(MODEL_DIR_PATH):
        os.mkdir(MODEL_DIR_PATH)
    model_path = os.path.join('model', 'face_recogniser.pkl')
    joblib.dump(FaceRecogniser(features_extractor, clf,
idx_to_class), model_path)


if __name__ == '__main__':
    main()
```

Lampiran 2 *Sorce Code Detect* Wajah *Python*

```
import os
import joblib
import argparse
from PIL import Image
from .util import draw_bb_on_img
from .constants import MODEL_PATH
from face_recognition import preprocessing

def recognise_faces(img):
    faces = joblib.load(MODEL_PATH)(img)
    if faces:
        draw_bb_on_img(faces, img)
    return faces, img

def check(face_id, image_path, threshold):
    preprocess = preprocessing.ExifOrientationNormalize()
    img = Image.open(image_path)
    img = preprocess(img)
    img = img.convert('RGB')

    faces, img = recognise_faces(img)
    if not faces:
        print('No faces found in this image.')

    accuracy = faces[0].top_prediction.confidence
    if (face_id == faces[0].top_prediction.label and accuracy >=
threshold):
        return {"status": True, "accuracy": accuracy, "faces":
faces}
    else:
        return {"status": False, "faces": faces}
```

Lampiran 3 *Sorce Code API Python*

```
from flask import Flask, request, jsonify
import time
import os
import inference.detect
import subprocess
import shutil

from waitress import serve

app = Flask(__name__)

@app.route('/', methods=['POST'])
def face_recognition():
    if request.method == 'POST':
        # Check if the POST request contains a file
        if 'file' not in request.files:
            return jsonify({'message': 'No file part'}), 400

        file = request.files['file']

        # Check if the file is empty
        if file.filename == '':
            return jsonify({'message': 'No selected file'}), 400

        # Get the filename and extension
        filename = file.filename
        _, file_extension = os.path.splitext(filename)

        # You can access other form fields like this
        face_id = request.form.get('id')

        current_time = int(time.time())

        base_dir_upload = "uploads/"
        new_filename = face_id + "_" + str(current_time) +
file_extension;

        # create face id directory if not exist
        if not os.path.exists(base_dir_upload):
            # If it doesn't exist, create the directory
            os.makedirs(base_dir_upload)

        image_path = base_dir_upload + new_filename
        # Process the uploaded file as needed
        # For example, save it to a location
        file.save(image_path)

        threshold = 0.75
        face_match = inference.detect.check(face_id, image_path,
ld)
        if (face_match["status"]):
```

```
            return jsonify({"match": True, "message": "Face match
for id: " + face_id, "accuracy": str(face_match["accuracy"]),
"faces": face_match["faces"][0][2]}), 200
        else:
            return jsonify({"match": False, "message": "Face
didn't match for id: " + face_id, "faces":
face_match["faces"][0][2]}), 200

@app.route('/upload-dataset', methods=['POST'])
def upload_dataset():
    if request.method == 'POST':
        # Check if the POST request contains a file
        if 'file' not in request.files:
            return jsonify({'message': 'No file part'}), 400

        file = request.files['file']

        # Check if the file is empty
        if file.filename == '':
            return jsonify({'message': 'No selected file'}), 400

        # Get the filename and extension
        filename = file.filename
        _, file_extension = os.path.splitext(filename)

        # You can access other form fields like this
        face_id = request.form.get('id')

        current_time = int(time.time())

        base_dir_upload = "images_dataset/"
        new_filename = face_id + "_" + str(current_time) +
file_extension;

        # create face id directory if not exist
        if not os.path.exists(base_dir_upload + face_id):
            # If it doesn't exist, create the directory
            os.makedirs(base_dir_upload + face_id)

        image_path = base_dir_upload + face_id + "/" +
new_filename
        # Process the uploaded file as needed
        # For example, save it to a location
        file.save(image_path)

        return jsonify({"status": "Success", "message": "File
upload successfully for face id: " + face_id}), 200

@app.route('/delete-dataset', methods=['POST'])
def delete_dataset():
    if request.method == 'POST':
        # You can access other form fields like this
        face_id = request.form.get('id')

        base_dir_upload = "images_dataset/"
```

```
        # create face id directory if not exist
        if os.path.exists(base_dir_upload + face_id):
            try:
                shutil.rmtree(base_dir_upload + face_id)
                status = "Success"
                message = "Directory " + base_dir_upload + face_id
+ " was successfully removed."
            except OSError as e:
                status = "Error"
                message = "Error: " + str(e)
            return jsonify({"status": status, "message":
message}), 200
        else:
            return jsonify({"status": "Error", "message": "ID not
found"}), 400

@app.route('/train', methods=['GET'])
def train():
    if request.method == 'GET':
        # Command to execute
        command = "tasks/train.sh images_dataset"  # Replace with
your desired terminal command

        # Execute the command
        result = subprocess.run(command, shell=True,
stdout=subprocess.PIPE, stderr=subprocess.PIPE)

        # Check the return code to see if the command was
successful
        if result.returncode == 0:
            return jsonify({"message": "Training success"}), 200
        else:
            # Print the command's standard error
            return jsonify({"message": "Training failed with
error: " + result.stderr}), 400

if __name__ == '__main__':
    # app.run(debug=True)
    # serve(app, host='103.76.50.207', port=8080)
    serve(app, host='0.0.0.0', port=8000)
```

Lampiran 4 *Sorce Code Training* Wajah pada Sistem Admin

```
public function training()
    {
        $apiUrl = 'http://103.76.50.207:8000/train';
        $response = Http::get($apiUrl);

        // Check the response from the other site's API
        if ($response->successful()) {
            // Successful response
            $responseBody = $response->json();

            session()->flash('success', 'Training Berhasil.');
        } else {
            // Handle the error
            return $errorMessage = $response->json();
            return $errorMessage;

            // You can log the error or take appropriate action
        }
        if (request()->ajax()) {
            return Datatables::of(User::where('role', 1)->get())-
>addIndexColumn()
                ->addColumn(
                    'aksi',
                    function ($data) {
                        $dataj = json_encode($data);

                        $btn = "<ul class='list-inline mb-0'>
            <li class='list-inline-item'>
            <a href='/admin/foto/" . $data->id . "' class='btn
btn-primary btn-sm'>
            Foto</a>
            <button type='button' data-toggle='modal'
onclick='hapus(" . $data->id . ")'   class='btn btn-danger btn-
sm'>Hapus</button>
            </li>
            </ul>";
                        return $btn;
                    }
                )->rawColumns(['aksi'])->make(true);
        }
        return view('admin.va_peserta');
    }
```

Lampiran 5 *Sorce Code Detect* Wajah pada Sistem Mobile

```
kirim() async {
    setState(() {
      is_loading = true;
    });
      final SharedPreferences sharedPreferences = await
SharedPreferences.getInstance();
      final _dio = Dio();
      String fileName = foto!.path.split('/').last;
      FormData formData = FormData.fromMap({
        'file':
        await MultipartFile.fromFile(_path, filename:fileName),
        'id':  sharedPreferences.getInt('id')?? "s",
      });
      try{ Response response = await _dio.post(
          "http://103.76.50.207:8000/",
          data: formData,
        );
        var jsonResponse = response.data;
        if (response.statusCode == 200) {
          bool match =  jsonResponse['match'];
          if(match == true){
            setState(() {
              is_loading = false;
            });
            notif(context, "Berhasil", "Wajah Sesuai");
            SharedPreferences sharedPreferences = await
SharedPreferences.getInstance();
            String accuracy = jsonResponse['accuracy'];
            sharedPreferences.setString("accuracy", accuracy);
            Navigator.push(context, MaterialPageRoute(builder:
(context) => const Questions()),);
          }else{
            setState(() {is_loading = false;});
            notif(context, "Gagal", "Wajah Tidak Sesuai, Tingkat
Akurasi : " + jsonResponse['message']) ;
          }
        } else {
          setState(() {is_loading = false;});
          notif(context, "Gagal", "Gagal Melakukan Transaksi");
        }
      }on DioError catch (e){
        notif(context, 'gagal', 'Gagal DIO$e');
      }
  }
```

Lampiran 6 Posisi Wajah Terdeteksi Dibawah 75%