LAMPIRAN

Pelatihan Khusus dengan YOLOv5

Step 1 : Hubungkan dataset yang terdapat pada google drive kedalam google colab.

```
#import dataset dari google drive
   1. from google.colab import drive
   2. drive.mount('/content/drive')
#memanggil variabel torch dan IPython.display dan Image
   1. import torch
   2. import os
   3. from IPython.display import Image
#mendownload data yolov5 yg sudah tersedia pada github
   1. !git clone https://github.com/ultralytics/yolov5
   2. %cd yolov5
```

Step 2 : Install requirements

```
#install requirements untuk membaca PyYAML pada torch dan torch
vision
   1. !pip install -r requirements.txt
```

Step 3 : Train model YOLOv5

```
   1. !python train.py --img 640 --batch 10 --epochs 30 --
      data custom.yaml --weights yolov5s.pt --cache
```

Deployment YOLOv5

App.py

```
'''
      Deloyment untuk Domain Computer Vision (CV)
      SHILVI YANTI S - 2018230050 - TEKNOLOGI INFORMASI
      UNIVERSITAS DARMA PERSADA
      2023
'''


# =[Modules dan Packages]========================
   1. from flask import Flask, render_template, request, jsonify,
      redirect, Response
   2. from werkzeug.utils import secure_filename
   3. import pandas as pd
   4. import numpy as np
   5. import os
   6. import tensorflow as tf
   7. from tensorflow.keras.models import Sequential
   8. from tensorflow.keras.layers import Conv2D, MaxPooling2D,
      Flatten, Dense, Activation, Dropout, LeakyReLU
   9. from PIL import Image
```

```
    from fungsi import make_model
from tensorflow.keras.utils import load_img,
   img_to_array
    import tensorflow as ts
    import argparse
    import io
    from PIL import Image
    import cv2
    import torch
model_yolo = torch.hub.load("ultralytics/yolov5",
   "custom", path = "best.pt", force_reload=True)
    from io import BytesIO
    def gen():
    cap=cv2.VideoCapture(0)

# Read until video is completed

  while(cap.isOpened()):

# Capture frame-by-fram ## read the camera frame

  success, frame = cap.read()
  if success == True
  ret,buffer=cv2.imencode('.jpg',frame)
  frame=buffer.tobytes()

#print(type(frame))

  img = Image.open(io.BytesIO(frame))
  results = model_yolo(img, size=640)

#print(results)

#print(results.pandas().xyxy[0])

#results.render()

# updates results.imgs with boxes and labels

results.print() # print results to screen

#results.show()

#print(results.imgs)

#print(type(img))

#print(results)

#plt.imshow(np.squeeze(results.render()))

#print(type(img))

#print(img.mode)


#convert remove single-dimensional entries from the shape of an
array

  img = np.squeeze(results.render()) #RGB

# read image as BGR

  img_BGR = cv2.cvtColor(img, cv2.COLOR_RGB2BGR) #BGR
```

```
#print(type(img))

#print(img.shape)

#frame = img

#ret,buffer=cv2.imencode('.jpg',img)

#frame=buffer.tobytes()

#print(type(frame))

#for img in results.imgs:

#img = Image.fromarray(img)

#ret,img=cv2.imencode('.jpg',img)

#img=img.tobytes()

#encode output image to bytes

#img = cv2.imencode('.jpg', img)[1].tobytes()

#print(type(img))

else:

break

#print(cv2.imencode('.jpg', img)[1])

#print(b)

#frame = img_byte_arr

# Encode BGR image to bytes so that cv2 will convert to RGB

   1. frame = cv2.imencode('.jpg', img_BGR)[1].tobytes()

#print(frame)

   1. yield(b'--frame\r\n'b'Content-Type: image/jpeg\r\n\r\n' +
      frame + b'\r\n')
   2. def PredGambar(file_gmbr):
   3. file = file_gmbr
   4. gmbr_array = np.asarray(file)
   5. gmbr_array = gmbr_array*(1/225)
   6. gmbr_input = tf.reshape(gmbr_array, shape=[1, 150, 150, 3])
   7. predik_array = model.predict(gmbr_input)[0]
   8. df = pd.DataFrame(predik_array)
   9. df = df.rename({0: 'NilaiKemiripan'}, axis='columns')
   10. Kualitas = ['Beras', 'Batu', 'Gabah', 'Kutu',
       'MerahHitam']
   11.       df['Kelas'] = Kualitas
   12.       df = df[['Kelas', 'NilaiKemiripan']]
   13.       predik_kelas = np.argmax(model.predict(gmbr_input))
   14.       if predik_kelas == 0:
   15.       predik_Kualitas = 'Beras'
   16.       elif predik_kelas == 1:
   17.       predik_Kualitas = 'Batu'
   18.       elif predik_kelas == 2:
   19.       predik_Kualitas = 'Gabah'
   20.       elif predik_kelas == 3:
   21.       predik_Kualitas == 'Kutu'
```

```
    else:
    predik_Kualitas == 'MerahHitam'
    return predik_Kualitas, df

# =[Variabel Global]============================

app = Flask(__name__, static_url_path='/static')
app.config['MAX_CONTENT_LENGTH'] = 22500 * 22500
app.config['UPLOAD_EXTENSIONS'] = ['.jpg', '.JPG']
app.config['UPLOAD_PATH'] = './static/images/uploads/'

# model = None

NUM_CLASSES = 5
cifar10_classes = ["Beras", "Batu", "Gabah", "Kutu",
   "MerahHitam"]

# =[Routing]====================================

# [Routing untuk Halaman Utama atau Home]

@app.route("/")
def beranda():
return render_template('index.html')
@app.route("/beranda")
def beranda_2():
return render_template('index.html')

# [Routing untuk API]

@app.route("/api/deteksi", methods=['POST'])
def apiDeteksi():

# Set nilai default untuk hasil prediksi dan gambar yang
diprediksi

hasil_prediksi = '(none)'
gambar_prediksi = '(none)'

# Get File Gambar yg telah diupload pengguna

uploaded_file = request.files['file']
filename = secure_filename(uploaded_file.filename)

# Periksa apakah ada file yg dipilih untuk diupload

if filename != '':

# Set/mendapatkan extension dan path dari file yg diupload

file_ext = os.path.splitext(filename)[1]
gambar_prediksi = '/static/images/uploads/' + filename

# Periksa apakah extension file yg diupload sesuai (jpg)

if file_ext in app.config['UPLOAD_EXTENSIONS']:

# Simpan Gambar

uploaded_file.save(os.path.join(
app.config['UPLOAD_PATH'], filename))

# Memuat Gambar

lok = '.' + gambar_prediksi
gmbr = ts.keras.utils.load_img(lok, target_size=(150, 150))
```

```
   3. x = ts.keras.utils.img_to_array(gmbr)
   4. x = np.expand_dims(x, axis=1)
   5. gmbr = np.vstack([x])

# Prediksi Gambar

   1. kelas, df = PredGambar(gmbr)
   2. hasil_prediksi = kelas

# Return hasil prediksi dengan format JSON

   1. return jsonify({
   2. "prediksi": hasil_prediksi,
   3. "gambar_prediksi": gambar_prediksi
   4. })
   5. else:

# Return hasil prediksi dengan format JSON

   1. gambar_prediksi = '(none)'
   2. return jsonify({
   3. "prediksi": hasil_prediksi,
   4. "gambar_prediksi": gambar_prediksi
   5. })
   6. @app.route('/video')
   7. def video():

    """Video streaming route. Put this in the src attribute of an
img tag."""

   1. return Response(gen(),
   2. mimetype='multipart/x-mixed-replace; boundary=frame')
   3. @app.route('/objectDetection')
   4. def objectDetection():
   5. return render_template('detection.html')

# =[Main]========================================

   1. if__name__== '__main__':
   2. parser = argparse.ArgumentParser(description="Flask app
      exposing yolov5 models")
   3. parser.add_argument("--port", default=5000, type=int,
      help="port number")
   4. args  =  parser.parse_args()

# Load model yang telah ditraining

   1. model = make_model()
   2. model.load_weights("VGG16_1.0-best.h5")

# model.load_weights("model_cifar10_cnn_tf.h5")

# Run Flask di localhost

app.run(host="localhost", port=5
```