APPLICATION BUILDING

Build python code

Date	18 November 2022
Team ID	PNT2022TMID01254
Project Name	AI - based localization and classification of skin disease with erythema

app.py

```python
import re

import numpy as np

import os

from flask import Flask, app,request,render_template

import sys

from flask import Flask, request, render_template, redirect, url_for

import argparse

from tensorflow import keras

from PIL import Image

from timeit import default_timer as timer

import test

import pandas as pd

import numpy as np

import random


def get_parent_dir(n=1):
    """ returns the n-th parent dicrectory of the current
    working directory """
    current_path = os.path.dirname(os.path.abspath(__file__))
    for k in range(n):
        current_path = os.path.dirname(current_path)
```

```python
    return current_path



src_path =r'C:\Users\MadhuVasanth1606\Desktop\yolo_structure\2_Training\src'
print(src_path)
utils_path = r'C:\Users\MadhuVasanth1606\Desktop\yolo_structure\Utils'
print(utils_path)


sys.path.append(src_path)
sys.path.append(utils_path)


import argparse
from keras_yolo3.yolo import YOLO, detect_video
from PIL import Image
from timeit import default_timer as timer
from utils import load_extractor_model, load_features, parse_input, detect_object
import test
import utils
import pandas as pd
import numpy as np
from Get_File_Paths import GetFileList
import random


os.environ["TF_CPP_MIN_LOG_LEVEL"] = "3"


# Set up folder names for default values
data_folder = os.path.join(get_parent_dir(n=1), "yolo_structure", "Data")


image_folder = os.path.join(data_folder, "Source_Images")
```

```python
image_test_folder = os.path.join(image_folder, "Test_Images")


detection_results_folder = os.path.join(image_folder, "Test_Image_Detection_Results")
detection_results_file = os.path.join(detection_results_folder, "Detection_Results.csv")


model_folder = os.path.join(data_folder, "Model_Weights")


model_weights = os.path.join(model_folder, "trained_weights_final.h5")
model_classes = os.path.join(model_folder, "data_classes.txt")


anchors_path = os.path.join(src_path, "keras_yolo3", "model_data", "yolo_anchors.txt")


FLAGS = None



from cloudant.client import Cloudant


# Authenticate using an IAM API key
client = Cloudant.iam('5b73f72f-2449-4298-88e8-3f887f8bbd2d-
bluemix','t3wXXORf8KoIMLzYFX2sk4e22uluSBKhM9-K4Q5b1zuK', connect=True)



# Create a database using an initialized client
my_database = client.create_database('skindisease')



app=Flask(__name__)


#default home page or route
@app.route('/')
def index():
```

```python
    return render_template('index.html')



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



#registration page
@app.route('/register')
def register():
    return render_template('register.html')


@app.route('/afterreg', methods=['POST'])
def afterreg():
    x = [x for x in request.form.values()]
    print(x)
    data = {
    '_id': x[1], # Setting _id is optional
    'name': x[0],
    'psw':x[2]
    }
    print(data)


    query = {'_id': {'$eq': data['_id']}}


    docs = my_database.get_query_result(query)
    print(docs)
```

```python
    print(len(docs.all()))


    if(len(docs.all())==0):

        url = my_database.create_document(data)

        #response = requests.get(url)

        return render_template('register.html', pred="Registration Successful, please login using
your details")

    else:

        return render_template('register.html', pred="You are already a member, please login
using your details")


#login page
@app.route('/login')
def login():

    return render_template('login.html')


@app.route('/afterlogin',methods=['POST'])
def afterlogin():

    user = request.form['_id']

    passw = request.form['psw']

    print(user,passw)


    query = {'_id': {'$eq': user}}


    docs = my_database.get_query_result(query)

    print(docs)


    print(len(docs.all()))



    if(len(docs.all())==0):
```

```python
            return render_template('login.html', pred="The username is not found.")
        else:
            if((user==docs[0][0]['_id'] and passw==docs[0][0]['psw'])):
                return redirect(url_for('prediction'))
            else:
                print('Invalid User')


@app.route('/logout')
def logout():
    return render_template('logout.html')


@app.route('/prediction')
def prediction():
    return render_template('prediction.html')


@app.route('/result',methods=["GET","POST"])
def res():
    # Delete all default flags
    parser = argparse.ArgumentParser(argument_default=argparse.SUPPRESS)
    """
    Command line options
    """

    parser.add_argument(
        "--input_path",
        type=str,
        default=image_test_folder,
        help="Path to image/video directory. All subdirectories will be included. Default is "
```

```python
        + image_test_folder,
    )

    parser.add_argument(
        "--output",
        type=str,
        default=detection_results_folder,
        help="Output path for detection results. Default is "
        + detection_results_folder,
    )

    parser.add_argument(
        "--no_save_img",
        default=False,
        action="store_true",
        help="Only save bounding box coordinates but do not save output images with
annotated boxes. Default is False.",
    )

    parser.add_argument(
        "--file_types",
        "--names-list",
        nargs="*",
        default=[],
        help="Specify list of file types to include. Default is --file_types .jpg .jpeg .png .mp4",
    )

    parser.add_argument(
        "--yolo_model",
        type=str,
        dest="model_path",
```

```python
        default=model_weights,
        help="Path to pre-trained weight files. Default is " + model_weights,
    )

    parser.add_argument(
        "--anchors",
        type=str,
        dest="anchors_path",
        default=anchors_path,
        help="Path to YOLO anchors. Default is " + anchors_path,
    )

    parser.add_argument(
        "--classes",
        type=str,
        dest="classes_path",
        default=model_classes,
        help="Path to YOLO class specifications. Default is " + model_classes,
    )

    parser.add_argument(
        "--gpu_num", type=int, default=1, help="Number of GPU to use. Default is 1"
    )

    parser.add_argument(
        "--confidence",
        type=float,
        dest="score",
        default=0.25,
        help="Threshold for YOLO object confidence score to show predictions. Default is
0.25.",
```

```python
)

parser.add_argument(
    "--box_file",
    type=str,
    dest="box",
    default=detection_results_file,
    help="File to save bounding box results to. Default is "
    + detection_results_file,
)

parser.add_argument(
    "--postfix",
    type=str,
    dest="postfix",
    default="_disease",
    help='Specify the postfix for images with bounding boxes. Default is "_disease"',
)

FLAGS = parser.parse_args()

save_img = not FLAGS.no_save_img

file_types = FLAGS.file_types
#print(input_path)

if file_types:
    input_paths = GetFileList(FLAGS.input_path, endings=file_types)
    print(input_paths)
else:
```

```python
    input_paths = GetFileList(FLAGS.input_path)
    print(input_paths)

# Split images and videos
img_endings = (".jpg", ".jpeg", ".png")
vid_endings = (".mp4", ".mpeg", ".mpg", ".avi")

input_image_paths = []
input_video_paths = []
for item in input_paths:
    if item.endswith(img_endings):
        input_image_paths.append(item)
    elif item.endswith(vid_endings):
        input_video_paths.append(item)

output_path = FLAGS.output
if not os.path.exists(output_path):
    os.makedirs(output_path)

# define YOLO detector
yolo = YOLO(
    **{
        "model_path": FLAGS.model_path,
        "anchors_path": FLAGS.anchors_path,
        "classes_path": FLAGS.classes_path,
        "score": FLAGS.score,
        "gpu_num": FLAGS.gpu_num,
        "model_image_size": (416, 416),
    }
)
```

```python
# Make a dataframe for the prediction outputs
out_df = pd.DataFrame(
    columns=[
        "image",
        "image_path",
        "xmin",
        "ymin",
        "xmax",
        "ymax",
        "label",
        "confidence",
        "x_size",
        "y_size",
    ]
)


# labels to draw on images
class_file = open(FLAGS.classes_path, "r")
input_labels = [line.rstrip("\n") for line in class_file.readlines()]
print("Found {} input labels: {} ...".format(len(input_labels), input_labels))


if input_image_paths:
    print(
        "Found {} input images: {} ...".format(
            len(input_image_paths),
            [os.path.basename(f) for f in input_image_paths[:5]],
        )
    )
    start = timer()
```

```python
text_out = ""

# This is for images
for i, img_path in enumerate(input_image_paths):
    print(img_path)
    prediction, image,lat,lon= detect_object(
        yolo,
        img_path,
        save_img=save_img,
        save_img_path=FLAGS.output,
        postfix=FLAGS.postfix,
    )
    print(lat,lon)
    y_size, x_size, _ = np.array(image).shape
    for single_prediction in prediction:
        out_df = out_df.append(
            pd.DataFrame(
                [
                    [
                        os.path.basename(img_path.rstrip("\n")),
                        img_path.rstrip("\n"),
                    ]
                    + single_prediction
                    + [x_size, y_size]
                ],
                columns=[
                    "image",
                    "image_path",
                    "xmin",
                    "ymin",
```

```python
                "xmax",
                "ymax",
                "label",
                "confidence",
                "x_size",
                "y_size",
            ],
        )
    )
end = timer()
print(
    "Processed {} images in {:.1f}sec - {:.1f}FPS".format(
        len(input_image_paths),
        end - start,
        len(input_image_paths) / (end - start),
    )
)
out_df.to_csv(FLAGS.box, index=False)


# This is for videos
if input_video_paths:
    print(
        "Found {} input videos: {} ...".format(
            len(input_video_paths),
            [os.path.basename(f) for f in input_video_paths[:5]],
        )
    )
    start = timer()
    for i, vid_path in enumerate(input_video_paths):
        output_path = os.path.join(
```

```python
                FLAGS.output,
                os.path.basename(vid_path).replace(".", FLAGS.postfix + "."),
            )
            detect_video(yolo, vid_path, output_path=output_path)


        end = timer()
        print(
            "Processed {} videos in {:.1f}sec".format(
                len(input_video_paths), end - start
            )
        )
    # Close the current yolo session
    yolo.close_session()
    return render_template('prediction.html')



""" Running our application """
if __name__ == "__main__":
    app.run(debug=True)
```

File   Edit   Selection   View   Go   Run   Terminal   Help

C: > Users > LENOVO > Desktop > yolo_structure > ● app.py > ...

```python
import re
import numpy as np
import os
from flask import Flask, app,request,render_template
import sys
from flask import Flask, request, render_template, redirect, url_for
import argparse
from tensorflow import keras
from PIL import Image
from timeit import default_timer as timer
import test
import pandas as pd
import numpy as np
import random

def get_parent_dir(n=1):
    """ returns the n-th parent dicrectory of the current
    working directory """
    current_path = os.path.dirname(os.path.abspath(__file__))
    for k in range(n):
        current_path = os.path.dirname(current_path)
    return current_path


src_path =r'C:\Users\MadhuVasanth1606\Desktop\yolo_structure\2_Training\src'
print(src_path)
utils_path = r'C:\Users\MadhuVasanth1606\Desktop\yolo_structure\Utils'
print(utils_path)

sys.path.append(src_path)
sys.path.append(utils_path)

import argparse
from keras_yolo3.yolo import YOLO, detect_video
from PIL import Image
from timeit import default_timer as timer
from utils import load_extractor_model, load_features, parse_input, detect_object
import test
```