Designing Front End of The App

#project

Initially we will run the application on local system (Ubuntu 20.04). Later, we will migrate it to DGX

Step 1: Clone the Yolov7 repository

Type following command in Terminal

```
git clone https://github.com/WongKinYiu/yolov7.git
```

Step 2: Create a Virtual Environment and activate it

Type following command in Terminal

```
cd yolov7/
python3 -m venv virtual_env/
source virtual_env/bin/activate
```

Step 3: Install necessary dependencies to run Yolov7

Type following command in Terminal

```
pip3 install -r requirements.txt
```

Step 4: [OPTIONAL - Only If required] Add Linux modules to work on videos

Type following command in Terminal

```
sudo apt update
sudo apt install ffmpeg libsm6 libxext6  -y
```

Step 5: Install Flask and related frameworks

Type following command in Terminal

```
pip install Flask
pip install Flask-Bootstrap
```

Step 6: [OPTIONAL] Check whether Flask is working fine

Type following command in Terminal

```
gedit check_flask.py
```

Enter following code, save and exit.

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello_world():
    return "<p>Hello, NVIDIA GRIL Students!</p>"
```

Execute the above code
Type following command in Terminal

```
flask --app check_flask.py run
```

Step 7: Download and store html templates

Type following command in Terminal

```
mkdir templates
wget --no-check-certificate 'https://docs.google.com/uc?
export=download&id=19na9M-AjauZ9jagg8N-iQR0QBG-djXRR' -O templates.zip
unzip templates.zip ; rm templates.zip;
```

Step 8: Create a directory for storing uploaded data

Type following command in Terminal

```
mkdir static/files -p
```

Step 9: Download Yolov7 Tiny Weights

https://drive.google.com/file/d/14BpkMQcLX9gtv0JvqH4MVgOUUXuT2-13/view?usp=sharing
Store the weights in *yolov7/* directory

Note - later on you can also use your weights of your own custom trained model. Just save the weights (.pt) file in the yolov7/ directory

Step 10: Download hubconfCustom.py file (for making predictions)

https://drive.google.com/file/d/1lTUcdd6N-1eMbCgq-DCN0NlhWWRcoqSy/view?usp=sharing
Store the file in *yolov7/* directory

Step 11: Download a trial video

https://drive.google.com/file/d/1p0JtZAa66SgrX2rQkobE6ly_lCox5VQk/view?usp=sharing
Store the video in *yolov7/static/files/* directory

Step 12: Creating flaskApp.py

```
gedit flaskApp.py
```

Add following lines to *flaskApp.py*:

```
'''A minimalistic flask app for Yolov7'''

from flask import Flask, render_template,
Response,jsonify,request,session
from flask_bootstrap import Bootstrap
import cv2
from hubconfCustom import video_detection
```

Initialize the flask app by appending following code *flaskApp.py*:

```

app = Flask(__name__, static_folder = 'templates/assets/')
Bootstrap(app)

```

Add details about where uploaded files fill be stored in *flaskApp.py*

```
#The secret key helps to maintain a user session
app.config['SECRET_KEY'] = 'grilsessionkey'
```

Add the route page for your app *flaskApp.py*:

```
@app.route("/",methods=['GET','POST'])
@app.route("/home", methods=['GET','POST'])
def home():
        session.clear()
        return render_template('root.html')
```

Add a function in *flaskApp.py* that will generate video frames after inference from Yolov7:

```
def generate_frames(path_x = '',conf_= 0.25):
    yolo_output = video_detection(path_x,conf_)
    for detection_,FPS_,xl,yl in yolo_output:
            #The function imencode compresses the image and stores it in
the memory buffer that is resized to fit the result.
        ref,buffer=cv2.imencode('.jpg',detection_)
        frame=buffer.tobytes()
        yield (b'--frame\r\n'
                    b'Content-Type: image/jpeg\r\n\r\n' + frame +b'\r\n')
```

Add a function in *flaskApp.py* that will send the generated frames in the form of http Response:

```
@app.route('/FrontPage')
@app.route('/video')
def video():
    return Response(generate_frames(path_x =
'static/files/vid.mp4',conf_=0.75),mimetype='multipart/x-mixed-replace;
boundary=frame')
```

Add code to run app when this python script is run:

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

Step 12: Execute code

```
flask --app flaskApp.py run
```

TASK - Use Your Custom Trained Model

Step 1: Download custom trained Yolov7 weights from DGX

- The file is named *best.pt* and can be found inside *yolov7/runs/train/* directory on DGX.
    - Temporarily, you can use following link to download file
    - https://drive.google.com/file/d/18QKegyRCOrGMNOq4I1ZkPOm7cMmIav6T/view?usp=share_link
- Download it and store it in *yolov7/* directory on your local system

Step 2: Download information regarding your custom trained Yolov7 from DGX

- The file is named *custom_data.yaml* and can be found in *yolov7/data/* directory on DGX

    - Temporarily, you can use following link to download file
    - https://drive.google.com/file/d/1jHACKhLGmUWJN-485u2eZZGZCWpJoTE-/view?usp=share_link
- Download it and store it in *yolov7/data/* directory on local system

Step 3: Update hubconfCustom.py file as follows

Search for *opt* dictionary and change its contents:

```
opt  = {
    "weights": "best.pt", # Path to weights file default weights are for nano model
    "yaml"   : "data/custom_data.yaml",
    "img-size": 640, # default image size
    "conf-thres": 0.25, # confidence threshold for inference.
    "iou-thres" : 0.45, # NMS IoU threshold for inference.
    "device" : 'cpu',  # device to run our model i.e. 0 or 0,1,2,3 or cpu
    "classes" : classes_to_filter  # list of classes to filter or None
}
```

Step 4: Run your Flask app

```
flask --app flaskApp.py run
```