



MANUAL DOCUMENT OF SUPPALUK API

PROJECT REPORT

CPE463 Image Processing and Computer Vision 2/2563

GROUP MEMBERs

Suppachai	Booncharoen	61070501051
Piyaluk	Tunsirichaiya	61070501063
Natthanan	Bhukan	61070507206
Athipoom	Sawatwong	61070507235

1. Project Overview

SUPPALUK API คือ API สำหรับการคัดแยกขยะ โดยผู้ใช้งานสามารถอัปโหลดรูปภาพขยะหรือวัตถุที่สงสัยลงไป จากนั้น API จะบอกประเภทของขยะนั้นแก่ผู้ใช้ ทำให้เกิดความสะดวกสบายในการคัดแยกขยะมากขึ้น โดยโจทย์ของ Suppaluk API จะเป็น Image classification

2. Scopes / Limitations

1. SUPPALUK API สามารถแยกขยะได้ 3 ประเภทด้วยกัน ได้แก่
 - a. ขวดน้ำ
 - b. ถุงขยะ
 - c. กระป๋องน้ำ
2. ภาพที่นำมาใช้กับ API จำเป็นต้องมีวัตถุเดียวในภาพนั้น และมีพื้นหลังของภาพตัดกับวัตถุชัดเจน
3. หลังจาก API ประมวลผลเรียบร้อย จะแสดง label สำหรับบอกว่าเป็นขยะประเภทใด

3. Dependency version

- Python 3.8.X or higher
- Click 7.1.2 or higher
- Cycler 0.10.0 or higher
- Decorator 4.4.2 or higher
- Fastapi 0.65.1 or higher
- H11 0.12.0 or higher
- Httptools 0.1.2 or higher
- Imageio 2.9.0 or higher
- Joblib 1.0.1 or higher
- Kiwisolver 1.3.1 or higher
- Matplotlib 3.4.2 or higher

- Networkx 2.5.1or higher
- Numpy 1.20.3 or higher
- Opencv-python 4.5.2.52 or higher
- Pickle5 0.0.11 or higher
- Pillow 8.2.0 or higher
- Pydantic 1.8.2 or higher
- Pyparsing 2.4.7or higher
- Python-dateutil 2.8.1or higher
- Python-dotenv 0.17.1 or higher
- Python-multipart 0.0.5 or higher
- PyWavelets 1.1.1 or higher
- PyYAML5.4.1 or higher
- Scikit-image 0.18.1 or higher
- Scikit-learn 0.22.2.post1
- Scipy 1.6.3
- Six 1.16.0
- Starlette 0.14.2 or higher
- Tifffile 2021.4.8 or higher
- Typing-extensions 3.10.0.0 or higher
- Uvicorn 0.13.4 or higher
- Uvloop 0.15.2 or higher
- Watchgod 0.7 or higher
- Websockets 8.1 or higher

4. Installation

4.1. Install environment control tools

ในการติดตั้ง Environment control tools จะมี 2 ทางเลือกในการใช้

1.) Pipenv สามารถ Download ได้ผ่านลิงค์ <https://pypi.org/project/pipenv/> โดยต้องทำการติดตั้งผ่าน pip installer ของ python โดยต้องเป็น python ที่มี version มากกว่าหรือเท่ากับ 3.8.X

```
▶ >>> pip install pipenv
```

2.) Anaconda สามารถ Download ได้ผ่านลิงค์ <https://www.anaconda.com> (สามารถดาวน์โหลดตั้งผ่านเว็บไซต์ได้เลย)

4.2. Create new environment for run application

ในการสร้าง Environment ใหม่ เพื่อไว้ใช้สำหรับ Application จะมี 2 ทางเลือก

1.) Pipenv

1.1. ให้เปิด terminal shell ขึ้นมาแล้วทำการเข้าไปที่ folder ที่มี Application ของ Suppaluk อยู่

```
cd suppaluk
```

1.2. ต่อมาทำการสร้าง Environment ใหม่

```
pipenv shell
```

2.) Anaconda

2.1. ให้เปิด Terminal shell ขึ้นมาแล้วเข้าไปที่ Folder ที่มี Application ของ Suppaluk อุย!

```
cd suppaluk
```

2.2. ต้องมาทำการสร้าง Environment ใหม่ ในการพิมพ์คำสั่งนี้ โดยต้องตั้งชื่อ Environment ด้วย

```
conda create -n your_env_name python=3.8
```

4.3. Install software dependency

ในการติดตั้ง Software dependency หรือ Software package ของ Suppaluk API ให้ทำการพิมพ์คำสั่ง

```
pip install -r requirements.txt
```

5. How to run application

หลังจากติดตั้ง Environment และ Software dependency หรือ Software package เสร็จแล้วให้ run คำสั่งต่อไปนี้เพื่อเป็นการใช้งาน Suppaluk API โดยจะแบ่งเป็น 2 หมวด

5.1. สำหรับทดลองใช้งาน

a) ให้ทำการ Run คำสั่งนี้

```
uvicorn main:app --reload
```

ในหน้า Terminal จะได้ผลลัพธ์ดังภาพนี้

```

>>> uvicorn main:app --reload                         0:53:31
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to
quit)
INFO:     Started reloader process [1624] using watchdog
Pipeline(memory=None,
          steps=[('standardscaler',
                  StandardScaler(copy=True, with_mean=True, with_std
=True)),
                  ('linearsvc',
                  LinearSVC(C=1.0, class_weight=None, dual=True,
                  fit_intercept=True, intercept_scaling=1,
                  loss='squared_hinge', max_iter=200,
                  multi_class='ovr', penalty='l2', random_
state=42,
                  tol=0.0001, verbose=0))],
          verbose=False)
INFO:     Started server process [1626]
INFO:     Waiting for application startup.
INFO:     Application startup complete.

```

b) แล้วทำการเปิด Web browser และไปที่ URL

<http://127.0.0.1:8000/docs>

จะได้หน้านี้ขึ้นมา ก็จะสามารถใช้งานได้เลย

FastAPI 0.1.0 [OAS3](#)
[/openapi.json](#)

default

[GET](#) / Index

[POST](#) /classify Detect

Schemas

Body_detect_classify_post >

HTTPValidationError >

ValidationError >

c) ถ้าต้องการใช้ Model ที่ได้ทำการ Train จาก Dataset เพิ่มเติม

สามารถปรับเปลี่ยนได้โดยการเปิดไฟล์ main.py ด้วย IDE ที่ต้องการ
(จากตัวอย่างจะเป็น vim)

```
vim main.py
```

```
1 from src.model import suppaluk
2 from fastapi.middleware.cors import CORSMiddleware
3 from starlette.responses import StreamingResponse
4 from fastapi import FastAPI, File, UploadFile
5 from src.model import suppaluk
6 import io
7 model = suppaluk()
8 app = FastAPI()
9
10 app.add_middleware(
11     CORSMiddleware,
12     allow_origins=['*'],
13     allow_credentials=True,
14     allow_methods=['*'],
15     allow_headers=['*'],
16 )
17
18 @app.get('/')
19 async def index():
20     return "Suppaluk"
21
22
23 @app.post('/classify')
24 async def detect(file: UploadFile = File(...)):
25     contents = await file.read()
26     img = model.readImgByte(contents)
27     img_pred = model.predict(img)
28     result = model.showImgBlob(img_pred)
29
30     return StreamingResponse(io.BytesIO(result), media_type='image/jpg')
```

จากนั้นทำการแก้ไขที่บรรทัด 7 โดยให้ใส่ Parameter ของ Function suppaluk เป็น

```
model = suppaluk(path="path/to/your_new_model.sav")
```

5.2. สำหรับในการ Train model จาก Dataset เพิ่มเติม

ก) เปิดไฟล์ train.py ด้วย IDE ที่ต้องการ (จากตัวอย่างจะเป็น vim)

```
vim train.py
```

จะได้ผลลัพธ์ ออกมาแบบนี้

```
1 from src.model import suppaluk
2 # load model
3 model = suppaluk(train=True)
4
5 # prepare dataset
6 path = "./datasets"
7 dataset = model.prepareData(path)
8
9 # train
10 clf = model.fit(dataset)
11
12 # evaluate
13 model.result(clf, dataset)
14
15 # save model
16 name = "your_model_name"
17 model.saveModel(clf, name=f"{name}.sav")
18
19 # Test
20 model = suppaluk(path=f"{name}.sav")
21 path_img = "./datasets/grabage_real/train/bottle/bt02.jpg"
22 img = model.readImg(path_img)
23 img_pred = model.predict(img)
24 print(f"Ground Truth: {path_img.split('/')[4]}")
25 model.showImg(img_pred)
```

ให้ระบุ Path ของ Dataset ก่อน (บรรทัดที่ 6) เพื่อใช้สำหรับการ Train ต่อมาทำการตั้งชื่อ Model ที่ทำการ Train ใหม่ สุดท้ายให้ระบุ Path ของ Test Image ที่ต้องการใช้ทดสอบ model ใหม่ ในส่วนของ Dataset ให้จัดรูปแบบของ folder ตามลักษณะนี้

```

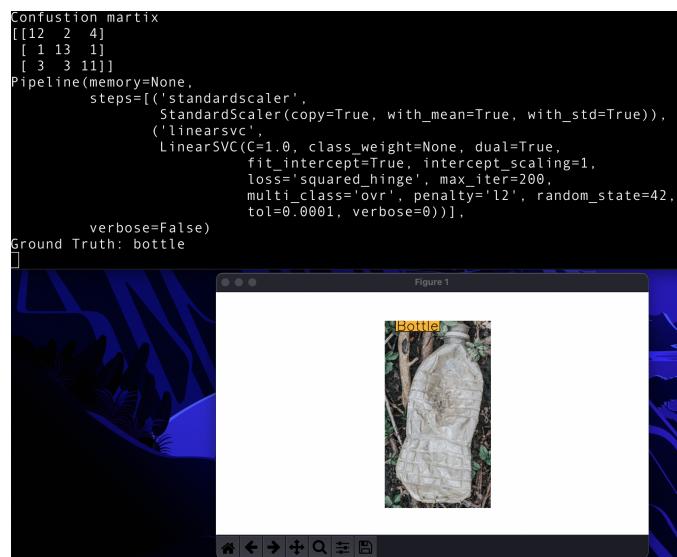
>>> tree datasets
datasets
|-- garbage
|   |-- test
|   |   |-- bottle
|   |   |   |-- plastic11.jpg
|   |   |   |-- plastic22.jpg
|   |   |   |-- plastic23.jpg
|   |   |   |-- plastic29.jpg
|   |   |   |-- plastic34.jpg
|   |   |   |-- plastic37.jpg
|   |   |-- metal
|   |   |   |-- metal230.jpg
|   |   |   |-- metal262.jpg
|   |   |   |-- metal386.jpg
|   |   |   |-- meta14.jpg
|   |   |   |-- meta71.jpg
|   |   |   |-- meta99.jpg
|   |   |-- snack
|   |   |   |-- trash10.jpg
|   |   |   |-- trash21.jpg
|   |   |   |-- trash27.jpg
|   |   |   |-- trash33.jpg
|   |   |   |-- trash40.jpg
|   |   |   |-- trash41.jpg
|   |-- train
|       |-- bottle
|           |-- plastic1.jpg
|           |-- plastic10.jpg
|           |-- plastic100.jpg
|           |-- plastic104.jpg
|           |-- plastic105.jpg
|           |-- plastic106.jpg
|           |-- plastic116.jpg
|           |-- plastic122.jpg
|           |-- plastic123.jpg
|           |-- plastic124.jpg
|           |-- plastic127.jpg
|           |-- plastic128.jpg
|           |-- plastic14.jpg
|           |-- plastic21.jpg
|           |-- plastic32.jpg
|           |-- plastic38.jpg

```

b) Run application ในหน้า Terminal โดยใช้คำสั่ง

```
python train.py
```

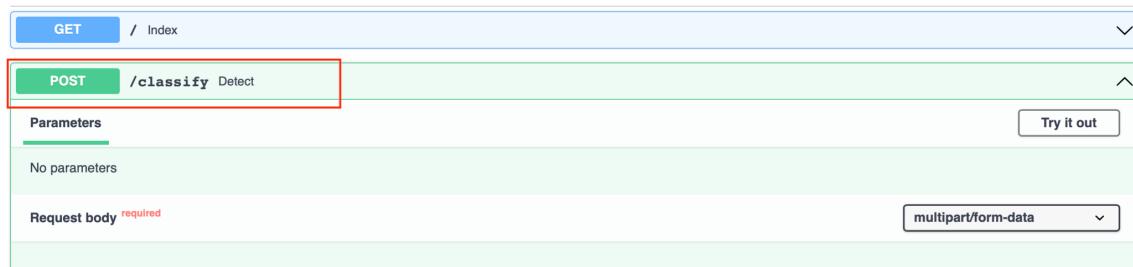
ในหน้า Terminal จะได้ผลลัพธ์ตามนี้ โดยจะแสดงผล Evluation ของ model, Confustion matrix, Parameter ของ model หลังจาก Train เสร็จ จะได้ไฟล์อยู่ใน Folder ของ Suppaluk เลย และ รูปที่ใช้ในการ Test ที่มีการ Predict และ Label ร่วมไปถึง Ground truth ของรูปนั้น (กด q ที่รูปภาพเพื่อออก จำกำสั่ง)



6. How to use application

ในการใช้งานหมวดของทดลองสำหรับใช้งาน สามารถทำตามขั้นตอนได้ดังนี้

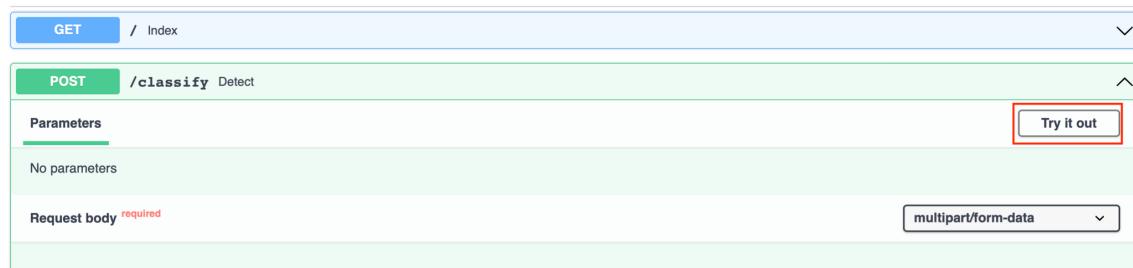
- เริ่มแรกให้เข้าไปที่ Web application ที่ได้ทำการ Run จากข้อ 5.1 และ ทำการคลิกที่ `/classify` จะได้ผลลัพธ์ดังภาพ



API endpoint details:

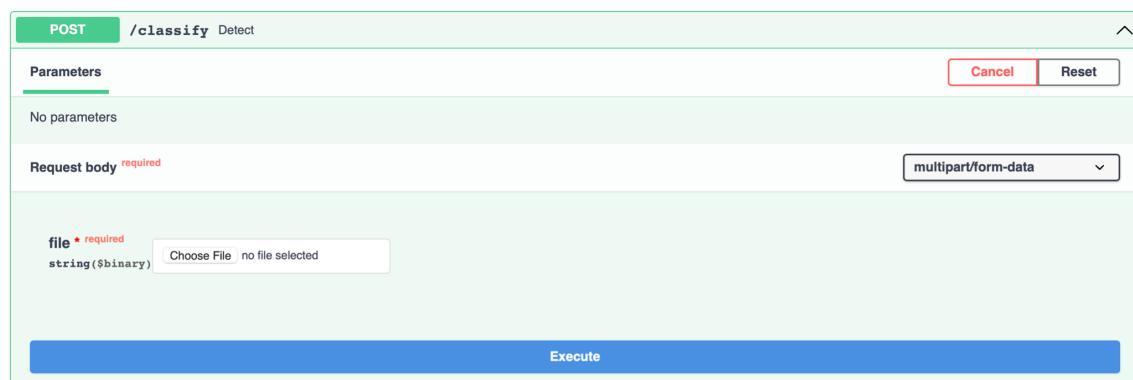
- Method: POST
- Path: /classify Detect
- Parameters: No parameters
- Request body: required (multipart/form-data)

- ต่อมากดที่ Try it out จะได้ผลลัพธ์ดังภาพ



API endpoint details:

- Method: POST
- Path: /classify Detect
- Parameters: No parameters
- Request body: required (multipart/form-data)



API endpoint details:

- Method: POST
- Path: /classify Detect
- Parameters: No parameters
- Request body: required (multipart/form-data)

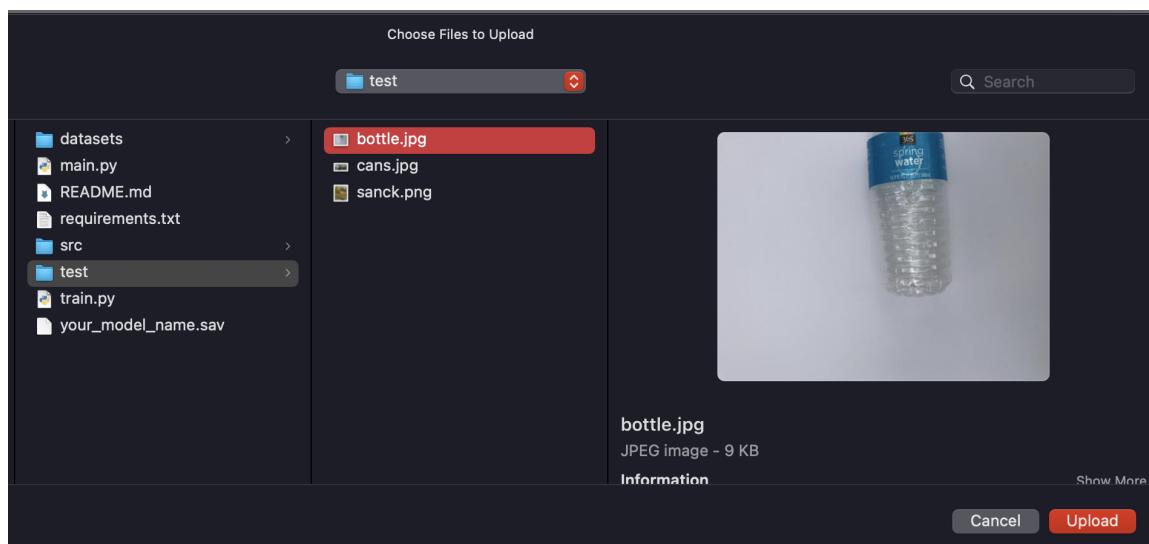
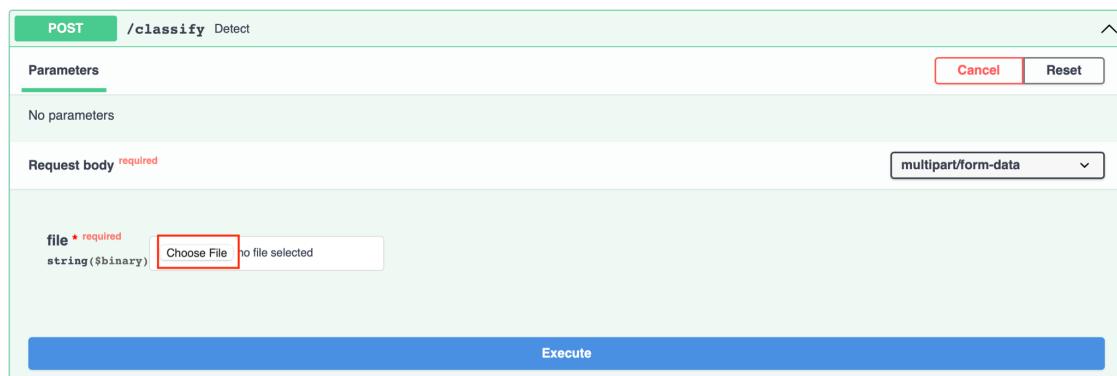
Request body fields:

- file * required (string(\$binary))
Choose File no file selected

Buttons:

- Execute
- Cancel
- Reset

- c) หลังจากนั้นกดที่ choose file เพื่อทำการเลือกไฟล์ที่จะทดสอบโดยทางผู้จัดทำได้เตรียมไฟล์ ทดสอบเอาไว้ให้แล้วอยู่ใน folder “ test ” ซึ่งจะได้ผลลัพธ์ตามภาพ (หรือผู้ใช้งานสามารถลองทดสอบกับไฟล์รูปภาพอื่นได้ แต่ต้องอยู่ Limitation ของ Application)



d) uploadไฟล์รูป ที่ต้องการทดสอบ และ กด Execute เพื่อทดสอบการใช้งาน

POST /classify Detect

Parameters

No parameters

Request body required

file required
string(\$binary)
Choose File sanck.png

Execute

e) หลังจากกด Execute จะได้ผลลัพธ์ดังภาพ

Responses

Curl

```
curl -X 'POST' \
'http://127.0.0.1:8000/classify' \
-H 'accept: application/json' \
-H 'Content-Type: multipart/form-data' \
-F 'file=@sanck.png;type=image/png'
```

Request URL

<http://127.0.0.1:8000/classify>

Server response

Code	Details
200	Response body



Snack

