Appendix H: Python Code (Model validation)

from models.YOLOv8.YOLOv8 import AnnotationYOLOv8

from models.DertResNet50.ResNet import AnnotationResNet

from models.InsightFace.InsightFace import AnnotationInsightFace

from models.SAHI\_YOLO.SAHI\_YOLOv8 import AnnotationSahiYOLOv8

import sys

# AnnotationInsightFace(annotations='./models/YOLOv8/BruceBeach426/test/test.json',

# images='./models/YOLOv8/BruceBeach426/test/images/',

# model\_name='insightface\_pre\_BB426',

# model\_weights='scrfd\_person\_2.5g.onnx').foo()

# AnnotationResNet(annotations='./models/YOLOv8/BruceBeach426/test/test.json',

# images='./models/YOLOv8/BruceBeach426/test/images/',

# model\_name='DertResNet50\_pre\_BB426',

# model\_weights="facebook/detr-resnet-50").foo()

# AnnotationYOLOv8(annotations='./models/YOLOv8/BruceBeach426/test/test.json',

# images='./models/YOLOv8/BruceBeach426/test/images/',

# model\_name='YOLOv8x\_pre\_BB426',

# model\_weights='./models/YOLOv8/weights/yolov8x').foo()

# AnnotationSahiYOLOv8(annotations='./models/YOLOv8/BruceBeach426/test/test.json',

# images='./models/YOLOv8/BruceBeach426/test/images/',

# model\_name='YOLOv8x\_pre\_BB426\_SAHI',

# model\_weights='./models/YOLOv8/weights/yolov8x').foo()

# AnnotationYOLOv8(annotations='./models/YOLOv8/BruceBeach426/test/test.json',

# images='./models/YOLOv8/BruceBeach426/test/images/',

# model\_name='YOLOv8x\_BB158\_500\_t18\_b',

# model\_weights='./models/YOLOv8/weights/YOLOv8x\_BB158\_500\_t18\_b.pt').foo()

AnnotationYOLOv8(annotations='./models/YOLOv8/BruceBeach426/test/test.json',

images='./models/YOLOv8/BruceBeach426/test/images/',

model\_name='YOLOv8x\_BB426\_760\_t31\_b',

model\_weights='./models/YOLOv8/weights/YOLOv8x\_BB426\_760\_t31\_b.pt').foo()

AnnotationSahiYOLOv8(annotations='./models/YOLOv8/BruceBeach426/test/test.json',

images='./models/YOLOv8/BruceBeach426/test/images/',

model\_name='YOLOv8x\_BB426\_760\_t31\_b\_SAHI',

model\_weights='./models/YOLOv8/weights/YOLOv8x\_BB426\_760\_t31\_b.pt').foo()

Appendix I: Python Code (Model training)

import glob

import json

import pandas as pd

import numpy as np

from ultralytics import YOLO

pd.set\_option('display.max\_columns', 30)

pd.set\_option('display.width', 2000)

pd.set\_option('display.precision', 3)

trains = glob.glob('../../runs/detect/\*')

if not trains:

train\_result\_folder = "train"

else:

train\_result\_folder = "train" + str(np.array([int(path.rsplit('train', 1)[1])

if path.rsplit('train', 1)[1] != '' else 0

for path in trains]).max() + 1)

# YOLOv5 trainin quickstart: https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data

# Train from the beginning with the right split of training/validation datasets.

with open('config.json', 'r') as f:

JSON\_Obj = json.load(f)

# YOLO('./weights/yolov8x.pt').train(

YOLO(JSON\_Obj["weights\_file"]).train(

data=JSON\_Obj["data"],

imgsz=JSON\_Obj["imgsz"],

epochs=JSON\_Obj["epochs"],

batch=JSON\_Obj["batch"],

patience=JSON\_Obj["patience"], # use `patience=0` to disable EarlyStopping

device=JSON\_Obj["device"]

)

Appendix J: Python Code (Test case analyzing)

import fiftyone as fo

import numpy as np

import seaborn as sns

from collections import defaultdict

import pandas as pd

from matplotlib import pyplot as plt

from pandas import IndexSlice

from fiftyone import ViewField as F

import os

import plotly

import kaleido

import plotly.io as pio

import sys

sys.path.append("../..")

from models.utils.Consts import MODEL\_LIST

pd.set\_option('display.max\_columns', 30)

pd.set\_option('display.width', 2000)

pd.set\_option('display.precision', 3)

def visualize\_mAP\_with\_plotly(default\_dataset, model\_name):

# predictions\_view = default\_dataset.take(default\_dataset.count(), seed=51)

# high\_conf\_view = predictions\_view.filter\_labels("predictions", F("confidence") > 0.5, only\_matches=False)

results = default\_dataset.view().evaluate\_detections(

"predictions",

gt\_field="detections",

eval\_key="eval",

compute\_mAP=True,

)

# Get the 10 most common classes in the dataset

counts = default\_dataset.count\_values("detections.detections.label")

classes\_top10 = sorted(counts, key=counts.get, reverse=True)[:10]

# Print a classification report for the top-10 classes

results.print\_report(classes=classes\_top10)

print("mAP score: " % results.mAP())

plot = results.plot\_pr\_curves(classes=["1"])

plot.update\_layout(

title\_text=model\_name,

font\_family="Courier New",

font\_color="blue",

title\_font\_family="Times New Roman",

title\_font\_color="red",

legend\_title\_font\_color="green"

)

plot.show()

# # running kaleido in a venv meets this bug: https://github.com/plotly/Kaleido/issues/78

# filepath = os.getcwd() + '/models/' + model\_name + '/' + model\_name + '.png'

# pio.write\_image(plot, filepath, format='png', engine='kaleido')

def generate\_img\_clf\_gt(metrics\_dic, model\_name):

try:

dataset = fo.load\_dataset(model\_name)

# # running kaleido in a venv meets this bug: https://github.com/plotly/Kaleido/issues/78

visualize\_mAP\_with\_plotly(dataset, model\_name)

for i, sample in enumerate(dataset):

print("[%d/%d] %s" % (i, dataset.count(), model\_name))

df = metrics\_dic[sample.filename]

if sample.get\_field('detections') is None:

lst = sample.get\_field('predictions').get\_field('detections')

ap = 1.0 if lst == [] else 0.0

[acc, pre, rec, f1, sup] = [1.] \* 5 if lst == [] else [0.] \* 5

else:

results = dataset.select(sample.id) \

.evaluate\_detections("predictions", gt\_field="detections", iou=0.4,

eval\_key="eval", compute\_mAP=True, )

ap = results.mAP()

[acc, pre, rec, f1, sup] = results.metrics().values()

df.loc[:, model\_name] = [acc, pre, rec, f1, sup, ap]

except Exception: # model\_name is not exist

raise

def new\_metric\_frame():

return pd.DataFrame(data=np.zeros(len(MODEL\_LIST) \* 6).reshape(-1, len(MODEL\_LIST)),

columns=MODEL\_LIST,

index=['accuracy', 'precision', 'recall', 'f1score', 'support', 'mAP'],

).copy()

# visualization

def metric\_visualization(filename, dpi=300):

name\_of\_file = filename.split('.', 1)[0]

plot\_df = pd.read\_csv(filename)

x = int(plot\_df.index.size / 4)

y = int(x / 3)

plot\_df = plot\_df.sort\_values(by=list(MODEL\_LIST)[::-1], ascending=False)

plot\_df = plot\_df.set\_index(['file'])

plot\_df = plot\_df.stack()

plot\_df.index = plot\_df.index.rename('model', level=1)

plot\_df.name = name\_of\_file

plot\_df = plot\_df.reset\_index()

fig, ax = plt.subplots(1, 1, figsize=(x, y), dpi=dpi)

sns.lineplot(data=plot\_df[plot\_df['model'] != MODEL\_LIST[-1]], x="file", y=name\_of\_file, hue='model', style="model",

markers=False, dashes=False, lw=1, palette=sns.color\_palette("bright", 8), ax=ax)

sns.lineplot(data=plot\_df[plot\_df['model'] == MODEL\_LIST[-1]], x="file", y=name\_of\_file, hue='model', style="model",

markers='\*', dashes=False, lw=1.5, palette=['r'], ax=ax)

ax.set\_title('Model ' + name\_of\_file + ' comparison')

ax.set\_xticklabels(ax.get\_xticklabels(), rotation=90, size=8)

plt.tight\_layout()

plt.savefig('./' + name\_of\_file + '.jpg')

path = './'

dict\_metrics = defaultdict(lambda: new\_metric\_frame())

for model\_id in range(len(MODEL\_LIST)):

generate\_img\_clf\_gt(dict\_metrics, MODEL\_LIST[model\_id])

tmp = pd.DataFrame(index=pd.MultiIndex(levels=[[], []], codes=[[], []], names=[u'file', u'metric']),

columns=MODEL\_LIST, dtype=float, )

for (n, a) in dict\_metrics.items():

for metric in a.index:

mt = a[a.index == metric]

tmp.loc[(n, metric), MODEL\_LIST] = mt.values.flatten().round(3)

tmp.loc[(n, metric), 'best\_model\_name'] = mt.T.idxmax().values[0]

tmp.to\_csv(path + 'rawdata.csv', float\_format='%.3f')

for metric in tmp.index.get\_level\_values('metric').unique():

tmp.loc[IndexSlice[:, metric], MODEL\_LIST].droplevel('metric').to\_csv(path + '%s.csv' % metric, float\_format='%.3f')

tmp.reset\_index('metric').pivot(columns='metric', values='best\_model\_name') \

.to\_csv(path + 'labels.csv', float\_format='%.3f')

metric\_visualization('accuracy.csv', dpi=150)

metric\_visualization('mAP.csv', dpi=150)