**Practical Assignment**

**Objective: - InstanceSegmentation with DeepFashion2**

DeepFashion2 is a comprehensive fashion dataset. It contains 491K diverse images of 13 popular clothing categories from both commercial shopping stores and consumers. It totally has 801K clothing clothing items, where each item in an image is labeled with scale, occlusion, zoom-in, viewpoint, category, style, bounding box, dense landmarks and per-pixel mask.There are also 873K Commercial-Consumer clothes pairs.

**Dataset Link: -**

Link :-<https://drive.google.com/drive/folders/125F48fsMBz2EF0Cpqk6aaHet5VH399Ok>

Github :-<https://github.com/switchablenorms/DeepFashion2>

**Task: -** Create a Web Application using FASTAPI. Use the end user should be able to upload an image and get results with the prediction score, class name and bounding box co ordinates. Use any CNN architecture launched after 2018.

**Deployment: -** Any Free Platform(Try to look out for free options.)

**Assignment Submission: -** Only submit the hosted app link. OR GitHub Link

import json

import numpy as np

from pycocotools import mask as maskUtils

thresh = 0.5

# load retrieval results

results\_image\_id\_all = []

results\_query\_score\_all = []

results\_query\_cls\_all = []

results\_query\_box\_all = []

results\_gallery\_id\_all = []

results\_gallery\_box\_all = []

results\_name = ' '

with open(results\_name, 'r') as f:

results = json.loads(f.read())

for i in results:

box = i['query\_bbox']

query\_box = [box[0],box[1],box[2]-box[0],box[3]-box[1]]

box = np.array(i['gallery\_bbox'])

gallery\_box = [box[:,0], box[:,1], box[:,2] - box[:,0], box[:,3] - box[:,1]]

gallery\_box = np.transpose(gallery\_box,(1,0)).tolist()

results\_image\_id\_all.append(i['query\_image\_id'])

results\_query\_score\_all.append(i['query\_score'])

results\_query\_cls\_all.append(i['query\_cls'])

results\_query\_box\_all.append(query\_box)

results\_gallery\_id\_all.append(i['gallery\_image\_id'])

results\_gallery\_box\_all.append(gellery\_box)

f.close()

results\_image\_id\_all = np.array(results\_image\_id\_all)

results\_query\_score\_all = np.array(results\_query\_score\_all)

results\_query\_cls\_all = np.array(results\_query\_cls\_all)

results\_query\_box\_all = np.array(results\_query\_box\_all)

results\_gallery\_id\_all = np.array(results\_gallery\_id\_all)

results\_gallery\_box\_all = np.array(results\_gallery\_box\_all)

# load query ground truth

query\_image\_id\_all = []

query\_box\_all = []

query\_cls\_all = []

query\_style\_all = []

query\_pair\_all = []

query\_name = '.../query\_gt.json'

with open(query\_name, 'r') as f:

query = json.loads(f.read())

for i in query:

box = i['bbox']

box = [box[0], box[1], box[2] - box[0], box[3] - box[1]]

query\_image\_id\_all.append(i['query\_image\_id'])

query\_box\_all.append(box)

query\_cls\_all.append(i['cls'])

query\_style\_all.append(i['style'])

query\_pair\_all.append(i['pair\_id'])

f.close()

# load gallery ground truth

query\_image\_id\_all = np.array(query\_image\_id\_all)

query\_box\_all = np.array(query\_box\_all)

query\_cls\_all = np.array(query\_cls\_all)

query\_style\_all = np.array(query\_style\_all)

query\_pair\_all = np.array(query\_pair\_all)

query\_num = len(np.where(query\_style\_all>0)[0]) # the number of all query clothing items

query\_id\_real= np.unique(query\_image\_id\_all) # image ids of query clothing items

gallery\_image\_id\_all = []

gallery\_box\_all = []

gallery\_style\_all = []

gallery\_pair\_all = []

gallery\_name = '.../gallery\_gt.json'

with open(gallery\_name, 'r') as f:

gallery = json.loads(f.read())

for i in gallery:

box = i['bbox']

box = [box[0], box[1], box[2] - box[0], box[3] - box[1]]

gallery\_image\_id\_all.append(i['gallery\_image\_id'])

gallery\_box\_all.append(box)

gallery\_style\_all.append(i['style'])

gallery\_pair\_all.append(i['pair\_id'])

f.close()

gallery\_image\_id\_all = np.array(gallery\_image\_id\_all)

gallery\_box\_all = np.array(gallery\_box\_all)

gallery\_style\_all = np.array(gallery\_style\_all)

gallery\_pair\_all = np.array(gallery\_pair\_all)

correct\_num\_1 = 0

correct\_num\_5 = 0

correct\_num\_10 = 0

correct\_num\_15 = 0

correct\_num\_20 = 0

miss\_num = 0 # the number of query items that fail to be detected

for id in query\_id\_real:

results\_id\_ind = np.where(results\_image\_id\_all==id)[0]

if len(results\_id\_ind) == 0: # in case no clothing item is detected

continue

query\_id\_ind = np.where(query\_image\_id\_all==id)[0] # all query items in the given image

pair\_id = query\_pair\_all[query\_id\_ind]

assert len(np.unique(pair\_id)) == 1

pair\_id = pair\_id[0]

results\_id\_score = results\_query\_score\_all[results\_id\_ind]

results\_id\_box = results\_query\_box\_all[results\_id\_ind]

results\_id\_cls = results\_query\_cls\_all[results\_id\_ind]

results\_id\_gallery\_id = results\_gallery\_id\_all[results\_id\_ind]

results\_id\_gallery\_box = results\_gallery\_box\_all[results\_id\_ind]

query\_id\_box = query\_box\_all[query\_id\_ind]

query\_id\_cls = query\_cls\_all[query\_id\_ind]

query\_id\_style = query\_style\_all[query\_id\_ind]

is\_crowd = np.zeros(len(query\_id\_box))

iou\_id = maskUtils.iou(results\_id\_box,query\_id\_box,is\_crowd)

iou\_ind = np.argmax(iou\_id,axis=1) # assign a ground truth label to each detected clothing item

for id\_ind in range(0,len(query\_id\_ind)):

style = query\_id\_style[id\_ind]

cls = query\_id\_cls[id\_ind]

# For a given ground truth query item, select a detected item on behalf of it:

# First find out all detected items which are assigned the given ground truth label

# and are classified correctly.

# Then select the detected item with the highest score among these detected items.

if style>0:

results\_style\_ind1 = np.where(iou\_ind==id\_ind)[0]

results\_style\_ind2 = np.where(results\_id\_cls==cls)[0]

results\_style\_ind = np.intersect1d(results\_style\_ind1,results\_style\_ind2)

if len(results\_style\_ind)>0:

results\_score\_style = results\_id\_score[results\_style\_ind]

score\_max\_ind = np.argmax(results\_score\_style)

results\_style\_query\_ind = results\_style\_ind[score\_max\_ind]

results\_style\_gallery\_id = results\_id\_gallery\_id[results\_style\_query\_ind]

results\_style\_gallery\_box = results\_id\_gallery\_box[results\_style\_query\_ind]

# find out the corresponding ground truth items in the gallery, that is ground truth items which have the same pair id and style as the query item.

gt\_gallery\_ind1 = np.where(gallery\_pair\_all==pair\_id)[0]

gt\_gellery\_ind2 = np.where(gallery\_style\_all==style)[0]

gt\_gallery\_ind = np.intersect1d(gt\_gallery\_ind1,gt\_gellery\_ind2)

gt\_gallery\_image\_id = gallery\_image\_id\_all[gt\_gallery\_ind]

gt\_gallery\_box = gallery\_box\_all[gt\_gallery\_ind]

assert len(gt\_gallery\_ind)>0

if len(gt\_gallery\_ind) == 1:

gt\_gallery\_image\_id = [gt\_gallery\_image\_id]

#calculate top-1

for t in range(0,1):

# if corresponding ground truth gallery images contains retrieved gallery image,

# first find out the exact corresponding ground truth gallery image,

# then find out ground truth gallery items in this ground truth gallery image(whose number may be greater than 1)

# if the overlap between the retrieved gallery item and one of the ground truth gallery items is over the thresh, the retrieved result is positive.

if results\_style\_gallery\_id[t] in gt\_gallery\_image\_id:

which\_ind = np.where(gt\_gallery\_image\_id==results\_style\_gallery\_id[t])[0]

crowd = np.zeros(len(which\_ind))

iou\_style = maskUtils.iou([results\_style\_gallery\_box[t]],gt\_gallery\_box[which\_ind],crowd)

if len(np.where(iou\_style>=thresh)[0])>0:

correct\_num\_1 = correct\_num\_1 + 1

break

# calculate top-5

for t in range(0,5):

if results\_style\_gallery\_id[t] in gt\_gallery\_image\_id:

which\_ind = np.where(gt\_gallery\_image\_id==results\_style\_gallery\_id[t])[0]

crowd = np.zeros(len(which\_ind))

iou\_style = maskUtils.iou([results\_style\_gallery\_box[t]],gt\_gallery\_box[which\_ind],crowd)

if len(np.where(iou\_style >= thresh)[0]) > 0:

correct\_num\_5 = correct\_num\_5 + 1

break

# calculate top-10

for t in range(0,10):

if results\_style\_gallery\_id[t] in gt\_gallery\_image\_id:

which\_ind = np.where(gt\_gallery\_image\_id==results\_style\_gallery\_id[t])[0]

crowd = np.zeros(len(which\_ind))

iou\_style = maskUtils.iou([results\_style\_gallery\_box[t]],gt\_gallery\_box[which\_ind],crowd)

if len(np.where(iou\_style >= thresh)[0]) > 0:

correct\_num\_10 = correct\_num\_10 + 1

break

# calculate top-15

for t in range(0,15):

if results\_style\_gallery\_id[t] in gt\_gallery\_image\_id:

which\_ind = np.where(gt\_gallery\_image\_id==results\_style\_gallery\_id[t])[0]

crowd = np.zeros(len(which\_ind))

iou\_style = maskUtils.iou([results\_style\_gallery\_box[t]],gt\_gallery\_box[which\_ind],crowd)

if len(np.where(iou\_style >= thresh)[0]) > 0:

correct\_num\_15 = correct\_num\_15 + 1

break

# calculate top-20

for t in range(0,20):

if results\_style\_gallery\_id[t] in gt\_gallery\_image\_id:

which\_ind = np.where(gt\_gallery\_image\_id==results\_style\_gallery\_id[t])[0]

crowd = np.zeros(len(which\_ind))

iou\_style = maskUtils.iou([results\_style\_gallery\_box[t]],gt\_gallery\_box[which\_ind],crowd)

if len(np.where(iou\_style >= thresh)[0]) > 0:

correct\_num\_20 = correct\_num\_20 + 1

break

else:

miss\_num = miss\_num + 1

print 'top-1'

print float(correct\_num\_1)/ query\_num

print 'top-5'

print float(correct\_num\_5)/ query\_num

print 'top-10'

print float(correct\_num\_10)/ query\_num

print 'top-15'

print float(correct\_num\_15)/ query\_num

print 'top-20'

print float(correct\_num\_20)/ query\_num