```python
from __future__ import print_function
import numpy as np
import os
import PIL.Image as Image
import pdb
from multiprocessing import Pool
from functools import partial
import matplotlib.pyplot as plt

eps = np.finfo(float).eps


def print_table():
    base_dir = '/home/zeng/data/datasets/saliency_Dataset'
    algs = ['WSS', 'BSCA', 'MB+', 'MST', 'MR', 'HS']
    # algs = ['WSS', 'LEGS', 'RFCN', 'DCL', 'DHS', 'MCDL', 'MDF']
    # algs = ['Ours_cap', 'Ours_cls', 'Ours_woAux', 'Ours_woUn', 'Ours_woDeeplab', 'Ours']
    # datasets = ['ECSSD', 'PASCALS', 'SOD', 'MSRA5K', 'OMRON']
    # algs = ['Ours_woDeeplab', 'Ours_ps_woDeeplab']
    datasets = ['SED1']
    for alg in algs:
        print(alg + '& ', end='')
        for i, dset in enumerate(datasets):
            input_dir = '{}/results/{}-Sal/{}'.format(base_dir, dset, alg)
            gt_dir = '{}/{}/masks'.format(base_dir, dset)
            output_dir = '{}/results/{}-npy'.format(base_dir, dset)
            if os.path.exists(os.path.join(output_dir, alg + '.npz')):
                sb = np.load(os.path.join(output_dir, alg + '.npz'))
                maxfm, mae = sb['maxfm'], sb['mea']
            else:
                maxfm, mae, _, _ = fm_and_mae(input_dir, gt_dir, output_dir, alg)
            if i != len(datasets) - 1:
                print('%.3f&%.3f& ' % (round(maxfm, 3), round(mae, 3)), end='')
            else:
                print('%.3f&%.3f\\\\' % (round(maxfm, 3), round(mae, 3)), end='\n')
                print('\hline', end='\n')


def draw_curves():
    base_dir = '/home/zeng/data/datasets/saliency_Dataset'
    algs = ['BSCA', 'MR', 'HS', 'Ours', 'WSS', 'DRFI', 'LEGS', 'MCDL', 'MDF']
    datasets = ['ECSSD', 'PASCALS', 'SOD', 'OMRON']
    # color = iter(plt.cm.rainbow(np.linspace(0, 1, len(algs))))
    for dset in datasets:
        fig = plt.figure()
        ax = fig.add_subplot(111)
        for i, alg in enumerate(algs):
            sb = np.load('{}/results/{}-npy/{}.npz'.format(base_dir, dset, alg))
            ax.plot(sb['recs'], sb['pres'], linewidth=2, label=alg)
        ax.grid(True)
        ax.set_xlabel('Recall', fontsize=14)
        ax.set_ylabel('Precision', fontsize=14)
        handles, labels = ax.get_legend_handles_labels()
        lgd = ax.legend(handles, labels, loc='center left',
```

```
    bbox_to_anchor=(0.5, -0.5), ncol=8, fontsize=14)
55          fig.savefig('%s.pdf' % dset, bbox_extra_artists=(lgd,),
    bbox_inches='tight')
56
57
58  def eva_one(param):
59      input_name, gt_name = param
60      mask = Image.open(input_name)
61      gt = Image.open(gt_name)
62      mask = mask.resize(gt.size)
63      mask = np.array(mask, dtype=np.float)
64      if len(mask.shape) != 2:
65          mask = mask[:, :, 0]
66      mask = (mask - mask.min()) / (mask.max() - mask.min() + eps)
67      gt = np.array(gt, dtype=np.uint8)
68      if len(gt.shape) > 2:
69          gt = gt[:, :, 0]
70      gt[gt != 0] = 1
71      pres = []
72      recs = []
73      mea = np.abs(gt - mask).mean()
74      # threshold fm
75      binary = np.zeros(mask.shape)
76      th = 2 * mask.mean()
77      if th > 1:
78          th = 1
79      binary[mask >= th] = 1
80      sb = (binary * gt).sum()
81      pre = sb / (binary.sum() + eps)
82      rec = sb / (gt.sum() + eps)
83      thfm = 1.3 * pre * rec / (0.3 * pre + rec + eps)
84      for th in np.linspace(0, 1, 21):
85          binary = np.zeros(mask.shape)
86          binary[mask >= th] = 1
87          pre = (binary * gt).sum() / (binary.sum() + eps)
88          rec = (binary * gt).sum() / (gt.sum() + eps)
89          pres.append(pre)
90          recs.append(rec)
91      pres = np.array(pres)
92      recs = np.array(recs)
93      return thfm, mea, recs, pres
94
95
96  def fm_and_mae(input_dir, gt_dir, output_dir=None, name=None):
97      if output_dir is not None and not os.path.exists(output_dir):
98          os.mkdir(output_dir)
99
100     filelist_gt = os.listdir(gt_dir)
101     gt_format = filelist_gt[0].split('.')[-1]
102     filelist_gt = ['.'.join(f.split('.')[:-1]) for f in filelist_gt]
103
104     filelist_pred = os.listdir(input_dir)
105     pred_format = filelist_pred[0].split('.')[-1]
106     filelist_pred = ['.'.join(f.split('.')[:-1]) for f in filelist_pred]
107
108     filelist = list(set(filelist_gt) & set(filelist_pred))
109
110     inputlist = [os.path.join(input_dir, '.'.join([_name, pred_format]))
    for _name in filelist]
111     gtlist = [os.path.join(gt_dir, '.'.join([_name, gt_format])) for
```

```
    _name in filelist]
112
113     pool = Pool(4)
114     results = pool.map(eva_one, zip(inputlist, gtlist))
115     thfm, m_mea, m_recs, m_pres = list(map(list, zip(*results)))
116     m_mea = np.array(m_mea).mean()
117     m_pres = np.array(m_pres).mean(0)
118     m_recs = np.array(m_recs).mean(0)
119     thfm = np.array(thfm).mean()
120     fms = 1.3 * m_pres * m_recs / (0.3 * m_pres + m_recs + eps)
121     maxfm = fms.max()
122     if not (output_dir is None or name is None):
123         np.savez('%s/%s.npz' % (output_dir, name), mea=m_mea, thfm=thfm,
    maxfm=maxfm, recs=m_recs, pres=m_pres, fms=fms)
124     return maxfm, m_mea, m_recs, m_pres
125
126
127 if __name__ == '__main__':
128     # dset = 'HKU-IS'
129     # fm, mae, _, _ =
    fm_and_mae('/home/zeng/data/datasets/saliency_Dataset/results/HKU-IS-Sal/WS
    S',
130     #
    '/home/zeng/data/datasets/saliency_Dataset/%s/masks'%dset)
131     # dsets = os.listdir('results')
132     # for dset in dsets:
133     #     fm, mae, _, _ = fm_and_mae('./results/%s'%dset,
134     #
    '/home/crow/data/datasets/saliency_Dataset/%s/masks'%dset)
135     #     print(dset)
136     #     print(fm)
137     #     print(mae)
138     #     print('==================')
139     print_table()
140     # draw_curves()
141
```