main.py 2019/4/22

```
1 # coding=utf-8
2 import torch
3 import torch.nn as nn
4 from torchvision import transforms
5 import torch.nn.functional as F
6 import os
7 import numpy as np
8 from PIL import Image
9 import argparse
10 import pickle
11 from datasets import Folder
12 from models import EncDec, FCN, DeepLab
13 from evaluate import fm_and_mae
14 from datasets.build_vocab import Vocabulary
15
16 from tqdm import tqdm
17
18
19 home = os.path.expanduser("~")
20
21 parser = argparse.ArgumentParser()
22 parser.add_argument('--img_dir',
  default='%s/data/datasets/saliency_Dataset/ECSSD/images' % (home)) #
  training dataset
23 parser.add argument('--gt dir',
  default='%s/data/datasets/saliency_Dataset/ECSSD/masks' % (home))
  training dataset
24 parser.add_argument('--result_dir', default='./results') # training
  dataset
25 parser.add_argument('--batchSize', type=int, default=24) # batch size
26 opt = parser.parse_args()
27 print(opt)
28
29
30 def make_dir(dir):
      if not os.path.exists(dir):
31
32
           os.makedirs(dir)
33
34
35 def main():
      img size = 256
36
37
      mean = [0.485, 0.456, 0.406]
38
      std = [0.229, 0.224, 0.225]
39
      make_dir(opt.result_dir)
40
41
      # data
      # Load vocabulary wrapper
42
      with open('vocab.pkl', 'rb') as f:
43
44
           vocab = pickle.load(f)
      loader = torch.utils.data.DataLoader(
45
46
           Folder(img_dir=opt.img_dir, gt_dir=opt.gt_dir,
47
  source_transform=transforms.Compose([transforms.Resize((img_size,
  img_size))]),
  target transform=transforms.Compose([transforms.Resize((img size,
  img_size))]),
49
                  mean=mean, std=std),
           batch_size=opt.batchSize, shuffle=False, num_workers=4,
50
  pin_memory=True)
```

main.py 2019/4/22

```
# caption and classification networks
51
52
       cls net = FCN(base='densenet169')
       cls net = cls net.cuda()
53
54
       cap_net = EncDec(len(vocab))
55
       cap_net = cap_net.cuda()
56
       # saliency network
57
       sal_net = DeepLab(base='densenet169', c_output=1)
58
       sal_net = nn.DataParallel(sal_net).cuda()
59
       # the 1st, 2nd and 3rd rows of Table 1
       cls_net.load_state_dict(torch.load('net-cls-init.pth'))
60
       cap_net.load_state_dict(torch.load('net-cap-init.pth'))
61
62
       output_dir = '/'.join([opt.result_dir, 'init', 'cls'])
63
       make_dir(output_dir)
64
       validate_one(loader, cls_net, output_dir)
       fm, mae, _,_ = fm_and_mae(output_dir, opt.gt_dir)
65
       print('cls fm %.3f'%fm)
66
67
       # the 2nd row of Table 1
       output_dir = '/'.join([opt.result_dir, 'init', 'cap'])
68
69
       make_dir(output_dir)
       validate_one(loader, cap_net, output_dir)
70
71
       fm, mae, _,_ = fm_and_mae(output_dir, opt.gt_dir)
72
       print('cap fm %.3f'%fm)
73
       # the 3rd row of Table 1
74
       output_dir = '/'.join([opt.result_dir, 'init', 'avg'])
75
       make dir(output dir)
       validate_two(loader, cls_net, cap_net, output_dir)
76
       fm, mae, _,_ = fm_and_mae(output_dir, opt.gt_dir)
77
78
       print('cls cap fm %.3f'%fm)
79
       # the 4th row of Table 1
80
       cls_net.load_state_dict(torch.load('cls-two-woun.pth'))
81
       cap_net.load_state_dict(torch.load('cap-two-woun.pth'))
       output_dir = '/'.join([opt.result_dir, 'at', 'avg'])
82
83
       make_dir(output_dir)
84
       validate_two(loader, cls_net, cap_net, output_dir)
85
       fm, mae, _,_ = fm_and_mae(output_dir, opt.gt_dir)
       print('cls cap at fm %.3f'%fm)
86
       # the 5th row of Table 1
87
       cls_net.load_state_dict(torch.load('cls-two-mr.pth'))
88
89
       cap_net.load_state_dict(torch.load('cap-two-mr.pth'))
90
       output_dir = '/'.join([opt.result_dir, 'ac', 'avg'])
91
       make_dir(output_dir)
92
       validate_two(loader, cls_net, cap_net, output_dir)
       fm, mae, _,_ = fm_and_mae(output_dir, opt.gt_dir)
93
94
       print('cls cap at ac fm %.3f'%fm)
       # the 6th row of Table 1
95
       sal_net.load_state_dict(torch.load('sal.pth'))
96
97
       output_dir = '/'.join([opt.result_dir, 'sal'])
98
       make_dir(output_dir)
99
       validate_one(loader, sal_net, output_dir)
       fm, mae, _,_ = fm_and_mae(output_dir, opt.gt_dir)
100
       print('sal fm %.3f'%fm)
101
102
103
   def validate_two(loader, net_cls, net_cap, output_dir):
104
       if not os.path.exists(output_dir):
105
106
           os.mkdir(output_dir)
107
       net_cls.eval()
       net_cap.eval()
108
       loader = tqdm(loader, desc='validating')
109
110
       for ib, (data, lbl, img_name, w, h) in enumerate(loader):
```

main.py 2019/4/22

```
with torch.no_grad():
111
112
                outputs_cls, _, _ = net_cls(data.cuda())
                outputs_cap = net_cap(data.cuda())
113
114
            outputs = (F.sigmoid(outputs_cls.cpu()) +
   F.sigmoid(outputs_cap.cpu()))/2
115
            outputs = outputs.squeeze(1).cpu().numpy()
            outputs *= 255
116
117
            for ii, msk in enumerate(outputs):
118
                msk = Image.fromarray(msk.astype(np.uint8))
                msk = msk.resize((w[ii], h[ii]))
119
                msk.save('{}/{}.png'.format(output_dir, img_name[ii]), 'PNG')
120
121
       net cls.train()
122
       net_cap.train()
123
124
125 def validate_one(loader, net, output_dir):
126
       net.eval()
       loader = tqdm(loader, desc='validating')
127
       for ib, (data, lbl, img_name, w, h) in enumerate(loader):
128
            with torch.no_grad():
129
130
                outputs = net(data.cuda())
131
            if isinstance(outputs, tuple):
132
                outputs = outputs[0]
133
            outputs = F.sigmoid(outputs.cpu())
134
            outputs = outputs.squeeze(1).cpu().numpy()
            outputs *= 255
135
            for ii, msk in enumerate(outputs):
136
137
                msk = Image.fromarray(msk.astype(np.uint8))
                msk = msk.resize((w[ii], h[ii]))
139
                msk.save('{}/{}.png'.format(output_dir, img_name[ii]), 'PNG')
140
       net.train()
141
142
143 if __name__ == "__main__":
144
       main()
```