```python
import os
import numpy as np
import PIL.Image as Image
import torch
from torch.utils import data
import pdb
import random


class WSFolder(data.Dataset):
    def __init__(self, root, gt_dir, crop=None, flip=False,
                 source_transform=None, target_transform=None,
                 mean=None, std=None, training=False):
        super(WSFolder, self).__init__()
        self.training = training
        self.mean, self.std = mean, std
        self.flip = flip
        self.s_transform, self.t_transform, self.crop = source_transform, target_transform, crop
        img_dir = os.path.join(root, 'images')
        names = [name[:-4] for name in os.listdir(gt_dir)]
        self.img_filenames = [os.path.join(img_dir, name+'.jpg') for name in names]
        self.gt_filenames = [os.path.join(gt_dir, name+'.png') for name in names]
        self.names = names

    def __len__(self):
        return len(self.names)

    def __getitem__(self, index):
        # load image
        img_file = self.img_filenames[index]
        img = Image.open(img_file)
        gt_file = self.gt_filenames[index]
        name = self.names[index]
        gt = Image.open(gt_file)
        WW, HH = gt.size
        if self.crop is not None:
            # random crop size of crop
            w, h = img.size
            th, tw = int(self.crop*h), int(self.crop*w)
            if w == tw and h == th:
                return 0, 0, h, w
            i = random.randint(0, h - th)
            j = random.randint(0, w - tw)
            img = img.crop((j, i, j + tw, i + th))
            gt = gt.crop((j, i, j + tw, i + th))
        if self.s_transform is not None:
            img = self.s_transform(img)
        if self.t_transform is not None:
            gt = self.t_transform(gt)
        img = img.resize((256, 256))
        gt = gt.resize((256, 256))
        img = np.array(img, dtype=np.uint8)
        if len(img.shape) < 3:
            img = np.stack((img, img, img), 2)
        if img.shape[2] > 3:
            img = img[:, :, :3]
        gt = np.array(gt, dtype=np.uint8)
```

```
58          if len(gt.shape) > 2:
59              gt = gt[:, :, 0]
60          if self.flip and random.randint(0, 1):
61              gt = gt[:, ::-1].copy()
62              img = img[:, ::-1].copy()
63          gt[gt != 0] = 1
64          img = img.astype(np.float64) / 255
65          if self.mean is not None:
66              img -= self.mean
67          if self.std is not None:
68              img /= self.std
69          img = img.transpose(2, 0, 1)
70          img = torch.from_numpy(img).float()
71          gt = torch.from_numpy(gt).float()
72          if self.training:
73              return img, gt, name
74          else:
75              return img, gt, name, WW, HH
76
77
78  class Folder(data.Dataset):
79      def __init__(self, root=None, crop=None, flip=False,
80                   source_transform=None, target_transform=None,
81                   mean=None, std=None, training=False, num=None,
    img_dir=None, gt_dir=None):
82          super(Folder, self).__init__()
83          self.training = training
84          self.mean, self.std = mean, std
85          self.flip = flip
86          self.s_transform, self.t_transform, self.crop = source_transform,
    target_transform, crop
87          if img_dir is None or gt_dir is None:
88              gt_dir = os.path.join(root, 'masks')
89              img_dir = os.path.join(root, 'images')
90          names = ['.'.join(name.split('.')[:-1]) for name in
    os.listdir(gt_dir)]
91          if num is not None:
92              names = random.sample(names, num)
93          self.img_filenames = [os.path.join(img_dir, name+'.jpg') for name
    in names]
94          self.gt_filenames = [os.path.join(gt_dir, name+'.png') for name
    in names]
95          self.names = names
96
97      def __len__(self):
98          return len(self.names)
99
100     def __getitem__(self, index):
101         # load image
102         img_file = self.img_filenames[index]
103         img = Image.open(img_file)
104         gt_file = self.gt_filenames[index]
105         name = self.names[index]
106         gt = Image.open(gt_file)
107         WW, HH = gt.size
108         if self.crop is not None:
109             # random crop size of crop
110             w, h = img.size
111             th, tw = int(self.crop*h), int(self.crop*w)
112             if w == tw and h == th:
```

```
                return 0, 0, h, w
            i = random.randint(0, h - th)
            j = random.randint(0, w - tw)
            img = img.crop((j, i, j + tw, i + th))
            gt = gt.crop((j, i, j + tw, i + th))
        if self.s_transform is not None:
            img = self.s_transform(img)
        if self.t_transform is not None:
            gt = self.t_transform(gt)
        img = img.resize((256, 256))
        gt = gt.resize((256, 256))
        img = np.array(img, dtype=np.uint8)
        if len(img.shape) < 3:
            img = np.stack((img, img, img), 2)
        if img.shape[2] > 3:
            img = img[:, :, :3]
        gt = np.array(gt, dtype=np.uint8)
        if len(gt.shape) > 2:
            gt = gt[:, :, 0]
        if self.flip and random.randint(0, 1):
            gt = gt[:, ::-1].copy()
            img = img[:, ::-1].copy()
        gt[gt != 0] = 1
        img = img.astype(np.float64) / 255
        if self.mean is not None:
            img -= self.mean
        if self.std is not None:
            img /= self.std
        img = img.transpose(2, 0, 1)
        img = torch.from_numpy(img).float()
        gt = torch.from_numpy(gt).float()
        if self.training:
            return img, gt, name
        else:
            return img, gt, name, WW, HH


if __name__ == "__main__":
    sb = Folder('/home/zhang/data/datasets/saliency_Dataset/ECSSD')
    sb.__getitem__(0)
    pdb.set_trace()
```