**U2net, attu2net, …**

**Loss.py**

# mix up loss to do the deep supervision with d0 as the final Salient Map

def multi\_mix\_loss\_fusion(d0, d1, d2, d3, d4, d5, d6, labels\_v): # d7,d0

loss0 = bce\_ssim\_iou\_loss(d0, labels\_v)

loss1 = bce\_ssim\_iou\_loss(d1, labels\_v)

loss2 = bce\_ssim\_iou\_loss(d2, labels\_v)

loss3 = bce\_ssim\_iou\_loss(d3, labels\_v)

loss4 = bce\_ssim\_iou\_loss(d4, labels\_v)

loss5 = bce\_ssim\_iou\_loss(d5, labels\_v)

loss6 = bce\_ssim\_iou\_loss(d6, labels\_v)

# loss7 = bce\_ssim\_loss(d7, labels\_v)

loss = loss0 + loss1 + loss2 + loss3 + loss4 + loss5 + loss6 # + 5.0\*lossa

# print("l0: %3f, l1: %3f, l2: %3f, l3: %3f, l4: %3f, l5: %3f, l6: %3f\n" % ( deprecate: 版本兼容性的问题，从[0]改为.item(0)

# loss0.data[0], loss1.data[0], loss2.data[0], loss3.data[0], loss4.data[0], loss5.data[0], loss6.data[0]))

# print("l0: %3f, l1: %3f, l2: %3f, l3: %3f, l4: %3f, l5: %3f, l6: %3f\n" % (

# loss0.item(), loss1.item(), loss2.item(), loss3.item(), loss4.item(), loss5.item(), loss6.item()))

return loss0, loss

# bce to do the deep supervision with d0 as the final Salient Map

def multi\_bce\_loss\_fusion(d0, d1, d2, d3, d4, d5, d6, labels\_v):

loss0 = bce\_loss(d0, labels\_v)

loss1 = bce\_loss(d1, labels\_v)

loss2 = bce\_loss(d2, labels\_v)

loss3 = bce\_loss(d3, labels\_v)

loss4 = bce\_loss(d4, labels\_v)

loss5 = bce\_loss(d5, labels\_v)

loss6 = bce\_loss(d6, labels\_v)

loss = loss0 + loss1 + loss2 + loss3 + loss4 + loss5 + loss6

# print("l0: %3f, l1: %3f, l2: %3f, l3: %3f, l4: %3f, l5: %3f, l6: %3f\n" % ( deprecate: same as above

# loss0.data.item(), loss1.data.item(), loss2.data.item(), loss3.data.item(), loss4.data.item(),

# loss5.data.item(),

# loss6.data.item()))

return loss0, loss

# todo more loss functions may work

# https://github.com/siyueyu/SCWSSOD

# Structure-ConsistentWeakly Supervised Salient Object Detection with Local Saliency Coherence

# 这是一种新的方法进行损失，因此需要将模型进行改变，最后SM先不上采样，而是先进行结构loss求解，之后再上采样，得到相关loss

**U3net**

**Loss.py**

**# mix up loss to do the deep supervision with d0 as the final Salient Map**

**def multi\_mix\_loss\_fusion(d1, labels\_v): # d7,d0**

**#loss0 = bce\_ssim\_iou\_loss(d0, labels\_v)**

**loss1 = bce\_ssim\_iou\_loss(d1, labels\_v)**

**#loss2 = bce\_ssim\_iou\_loss(d2, labels\_v)**

**#loss3 = bce\_ssim\_iou\_loss(d3, labels\_v)**

**#loss4 = bce\_ssim\_iou\_loss(d4, labels\_v)**

**#loss5 = bce\_ssim\_iou\_loss(d5, labels\_v)**

**#loss6 = bce\_ssim\_iou\_loss(d6, labels\_v)**

**# loss7 = bce\_ssim\_loss(d7, labels\_v)**

**loss=loss1**

**#loss = loss1 + loss2 + loss3 + loss4 + loss5 # + 5.0\*lossa**

**# print("l0: %3f, l1: %3f, l2: %3f, l3: %3f, l4: %3f, l5: %3f, l6: %3f\n" % ( deprecate: 版本兼容性的问题，从[0]改为.item(0)**

**# loss0.data[0], loss1.data[0], loss2.data[0], loss3.data[0], loss4.data[0], loss5.data[0], loss6.data[0]))**

**# print("l0: %3f, l1: %3f, l2: %3f, l3: %3f, l4: %3f, l5: %3f, l6: %3f\n" % (**

**# loss0.item(), loss1.item(), loss2.item(), loss3.item(), loss4.item(), loss5.item(), loss6.item()))**

**return loss1, loss**

**# bce to do the deep supervision with d0 as the final Salient Map**

**def multi\_bce\_loss\_fusion(d1, labels\_v):**

**#loss0 = bce\_loss(d0, labels\_v)**

**loss1 = bce\_loss(d1, labels\_v)**

**#loss2 = bce\_loss(d2, labels\_v)**

**#loss3 = bce\_loss(d3, labels\_v)**

**#loss4 = bce\_loss(d4, labels\_v)**

**#loss5 = bce\_loss(d5, labels\_v)**

**#loss6 = bce\_loss(d6, labels\_v)**

**loss=loss1**

**#loss = loss1 + loss2 + loss3 + loss4 + loss5**

**# print("l0: %3f, l1: %3f, l2: %3f, l3: %3f, l4: %3f, l5: %3f, l6: %3f\n" % ( deprecate: same as above**

**# loss0.data.item(), loss1.data.item(), loss2.data.item(), loss3.data.item(), loss4.data.item(),**

**# loss5.data.item(),**

**# loss6.data.item()))**

**return loss1, loss**

**# todo more loss functions may work**

**# https://github.com/siyueyu/SCWSSOD**

**# Structure-ConsistentWeakly Supervised Salient Object Detection with Local Saliency Coherence**

**# 这是一种新的方法进行损失，因此需要将模型进行改变，最后SM先不上采样，而是先进行结构loss求解，之后再上采样，得到相关loss**

**Train.py (U2net**

# forward + backward + optimize

d0, d1, d2, d3, d4, d5, d6 = net(inputs\_v)

loss2, loss = loss\_fnc(d0, d1, d2, d3, d4, d5, d6, labels\_v)

# 其中loss2保存的是最后一个SMap的loss，而loss保存的是所有SMap的loss

loss.backward()

optimizer.step()

running\_loss += loss.data.item()

running\_tar\_loss += loss2.data.item()

# del temporary outputs and loss

del d0, d1, d2, d3, d4, d5, d6, loss2, loss

Train (U3net

# forward + backward + optimize

d0, d1, d2, d3, d4, d5, d6 = net(inputs\_v)

loss2, loss = loss\_fnc(d0, d1, d2, d3, d4, d5, d6, labels\_v)

# 其中loss2保存的是最后一个SMap的loss，而loss保存的是所有SMap的loss

loss.backward()

optimizer.step()

running\_loss += loss.data.item()

running\_tar\_loss += loss2.data.item()

# del temporary outputs and loss

del d0, d1, d2, d3, d4, d5, d6, loss2, loss

# 내가 추가한거

with torch.no\_grad():

for i, val\_data in enumerate(val\_dataloader):

val\_inputs, val\_labels = val\_data['image'], val\_data['label']

val\_inputs = val\_inputs.type(torch.FloatTensor)

val\_labels = val\_labels.type(torch.FloatTensor)

if torch.cuda.is\_available():

val\_inputs, val\_labels = val\_inputs.cuda(), val\_labels.cuda()

val\_outputs = net(val\_inputs)

val\_loss1, val\_loss = loss\_fnc(val\_outputs, val\_labels)

val\_loss += val\_loss.data.item()

val\_tar\_loss += val\_loss1.data.item()

**test.py**

# --------- 4. inference for each image ---------

for i\_test, data\_test in enumerate(test\_salobj\_dataloader):

print("inferencing:", img\_name\_list[i\_test].split(os.sep)[-1])

inputs\_test = data\_test['image']

inputs\_test = inputs\_test.type(torch.FloatTensor)

if torch.cuda.is\_available():

inputs\_test = Variable(inputs\_test.cuda())

else:

inputs\_test = Variable(inputs\_test)

d1, d2, d3, d4, d5, d6, d7 = net(inputs\_test) # 注意现在是d1为最终的SMap

# normalization

pred = d1[:, 0, :, :]

pred = normPRED(pred)

# save results to test\_results folder

if not os.path.exists(prediction\_dir):

os.makedirs(prediction\_dir, exist\_ok=True)

save\_output(img\_name\_list[i\_test], pred, prediction\_dir)

del d1, d2, d3, d4, d5, d6, d7

if \_\_name\_\_ == "\_\_main\_\_":

main()