1、迁移 transfer ，的方法训练低分辨率人脸识别

在主干网络中添加一个数据变量，作为数据输入接口，接收，高分辨率网络的人脸识别512 维度特征，与当前网络的接口输出，直接求L2，作为损失函数,

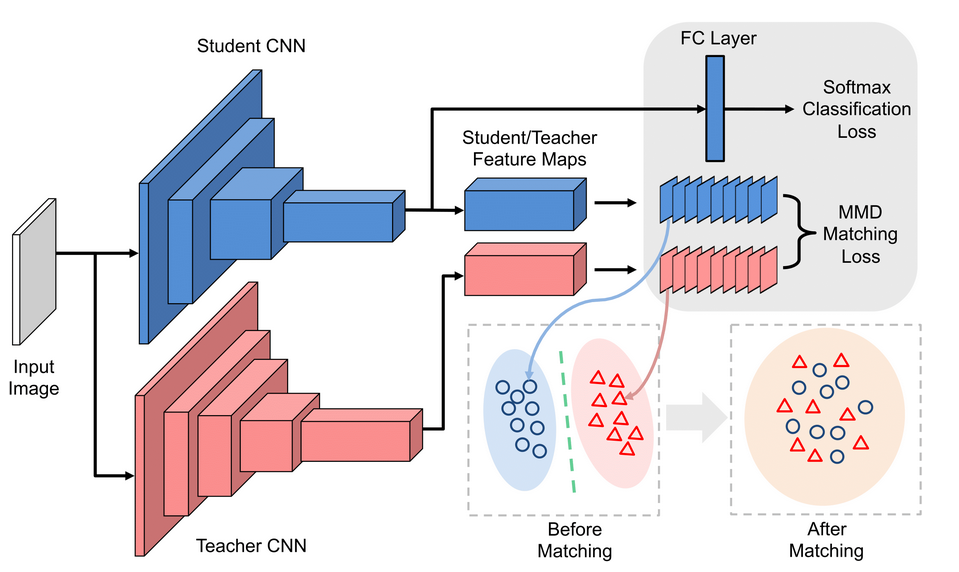
可使用的参考代码，mxnet

1、neuron-selectivity-transfer

• 两个模型512 feature， 先归一化然后求 求欧式距离 ，反向传播权重5

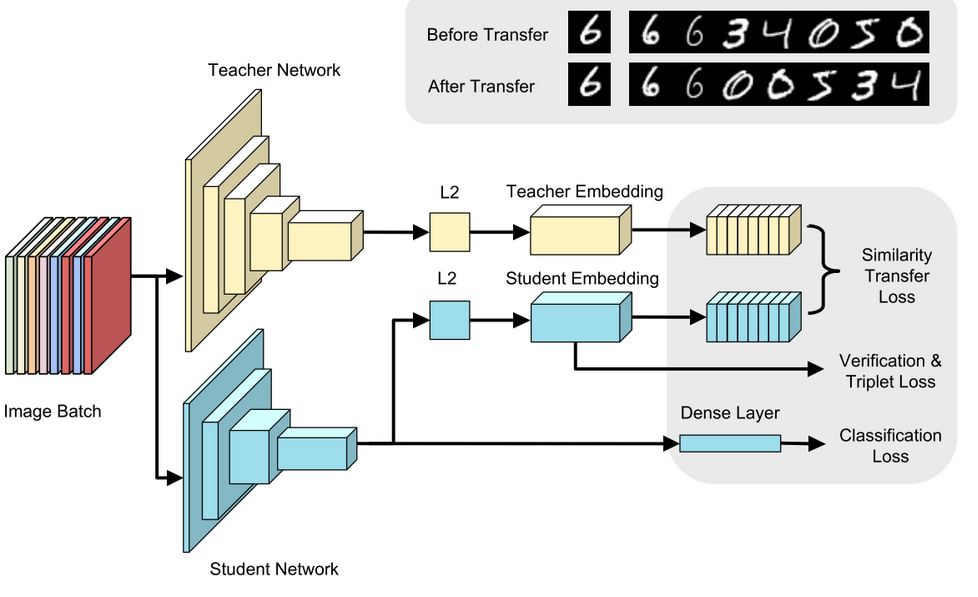
域适应损失，（MMD）最大均值差异损失，迁移学习中常用

https://arxiv.org/pdf/1707.01219.pdf mxnet mmd 实现，（https://gitee.com/huangchaosusu/neuron-selectivity-transfer，代码）



2、

https://blog.csdn.net/yimiyangguang1994/article/details/80833607 ，双通道蒸馏学习，<https://github.com/TuSimple/DarkRank>

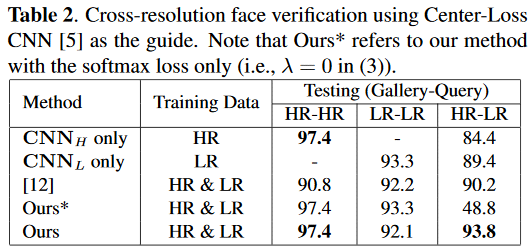


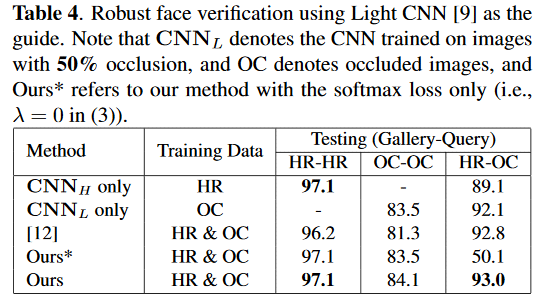
* 3、2017（guided CNN）

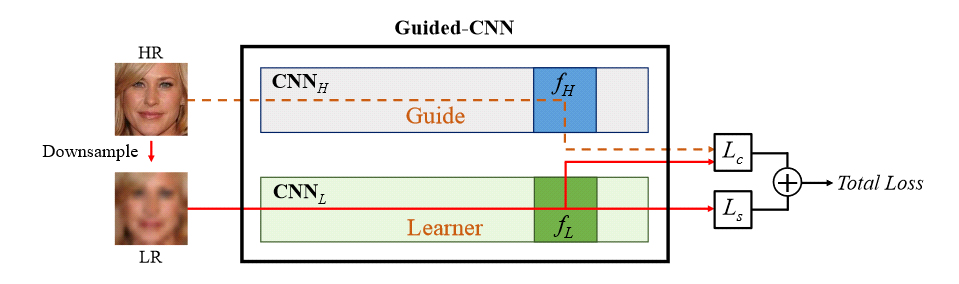
LEARNING GUIDED CONVOLUTIONAL NEURAL NETWORKS FORCROSS-RESOLUTION FACE RECOGNITION

<https://people.cs.nctu.edu.tw/~walon/publications/fu17mlsp.pdf>

低分辨率人脸验证，有遮挡的人脸验证

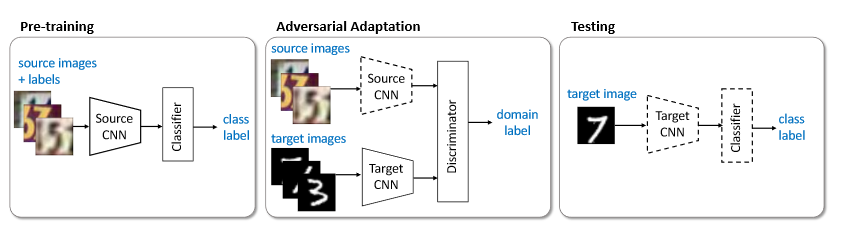






4、ADDA，对抗域损失迁移训练，没有mxnet,有pytorch tensorflow版

2017CVPR--Adversarial Discriminative Domain Adaptation



5、代码有参考，没有使用，因为里面的GRL 梯度反转层没有mxnet 实现, ADDA,有一个说明，直接梯度反转层的缺点

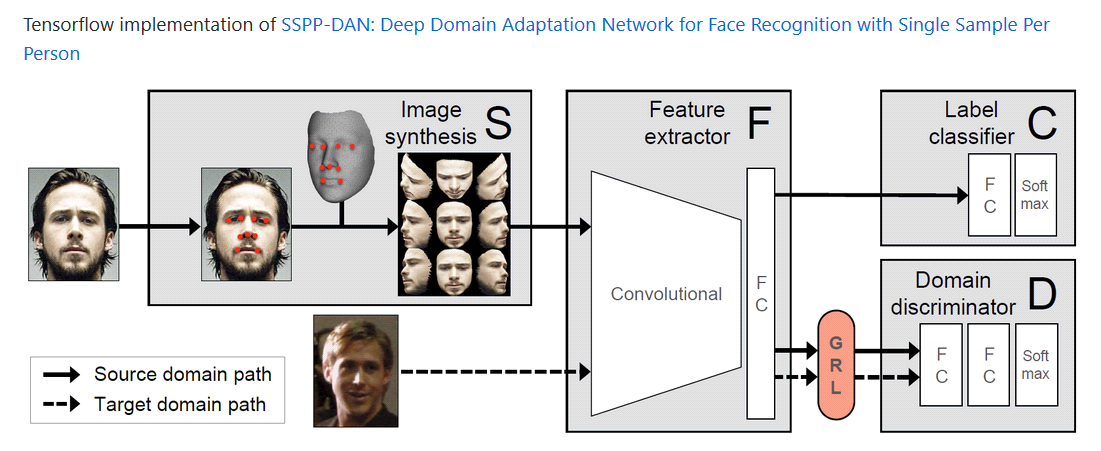
4，5都是类似对抗损失训练，多了一个鉴别器网络，输入是原网络和目标网络的特征，规定标签是0 和 1，然后二分类，得到域损失，达到迁移的目的，两者都是源网络训练好，直接得到特征，只训练目标网络，以及域鉴别器

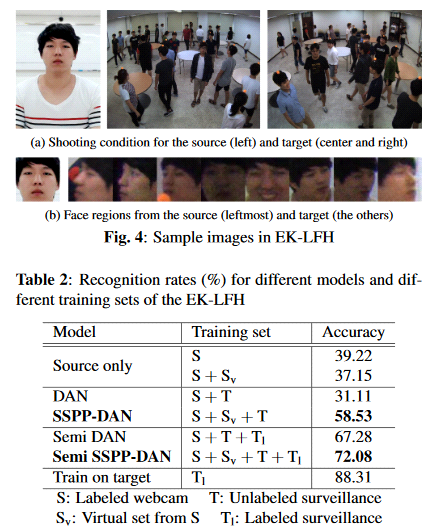
2018 SSPP-DAN: Deep Domain Adaptation Network for Face Recognition with Single Sample Per Person，单人脸例如只有证件照，图片的人脸识别场景，domloss 用到了梯度反转层，gradient reversal layer (GRL)

EK-LFH 是自制和scface 类似的数据，并多了多姿态

<https://arxiv.org/pdf/1702.04069.pdf>

<https://github.com/csehong/SSPP-DAN/blob/master/train_model.py>





6、mxnet mnist对抗学习<https://www.cnblogs.com/heguanyou/p/7642608.html> 代码解析，梯度更新

如何处理读取两组数据训练，如何，梯度更新两个网络

ADDA，的对抗学习提取特征，参考 pytorch 的实现代码，以及mxnet mnist gan 训练代码，梯度更新，主要过程，读取两组数据，然后鉴别器训练得到的梯度，给生成器网络训练

modD.forward(mx.io.DataBatch(outG, [label]), is\_train=True)

modD.backward()

diffD = modD.get\_input\_grads()

# diffD就是modG的loss产生的梯度，用它来向后传播并更新参数。

modG.backward(diffD)

modG.update()

1. adda 训练注意事项，bind, inputs\_need\_grad=True,设定true,为了进行输入特征的梯度计算（不设置默认是不计算输入的梯度的），鉴别器网络的更新，输入不需要，所以数据需要加detach,不计算输入的梯度，

第一次鉴别器网络运行更新后，第二次再鉴别器继续再进行一次，是为了得到输入的梯度，传递给目标网络，这里是个连接，

d\_model.bind(data\_shapes=data\_shapes,label\_shapes = label\_shapes,inputs\_need\_grad=True)，默认是false,不计算输入梯度，这里先设置计算梯度，在鉴别器网络本身更新的时候detach阻止输入的梯度更新，然后再第二次输入图片的时候，不阻止，就得到了目标网络的输出了。

教师网路可以是很复杂的网络，学生网络不一样都可以尝试

然后就是gan对抗训练中间label 0 1 的变化，

<https://www.cnblogs.com/heguanyou/p/7642608.html>说的也很详细

1. lr for discriminator: 1e-3  
   lr for target encoder: 1e-5  
   trainin epoch : 6 # 训练效果不好
2. 训练设置

default.kvstore = 'device'#'local' #'device' #MXNET\_ENABLE\_GPU\_P2P=1，，local P2P=0

export MXNET\_ENABLE\_GPU\_P2P=1

export MXNET\_GPU\_WORKER\_NTHREADS=4

MXNET\_BACKWARD\_DO\_MIRROR=1

1. 训练收敛的结果是借鉴作者的工程参数

https://github.com/corenel/pytorch-adda/blob/master/core/adapt.py

采用该作者提供的代码的超参数，鉴别器网络和目标网络用Adam 损失，初始学习率都是1e-4,不用SGD，（实际训练，训练验证集直接降采样的数据不行）

鉴别器网络采用，两个全连接输出一样512，然后二分类输出

参数设置

mxnet.optimizer.Adam(learning\_rate=0.0001, beta1=0.5, beta2=0.9, epsilon=1e-08)

鉴别器网络

fc1 = mx.sym.FullyConnected(data=data, num\_hidden=512,name="fc1") #512 this is me add ,can set any

act1=mx.sym.LeakyReLU(data=fc1, act\_type='prelu', name="prelu1")

fc2 = mx.sym.FullyConnected(data=act1, num\_hidden=512, name="fc2") # 512 this is me add ,can set any

act2 = mx.sym.LeakyReLU(data=fc2, act\_type='prelu', name="prelu2")

fc3 = mx.sym.FullyConnected(data=act2, num\_hidden=2, name="fc3") # class loss