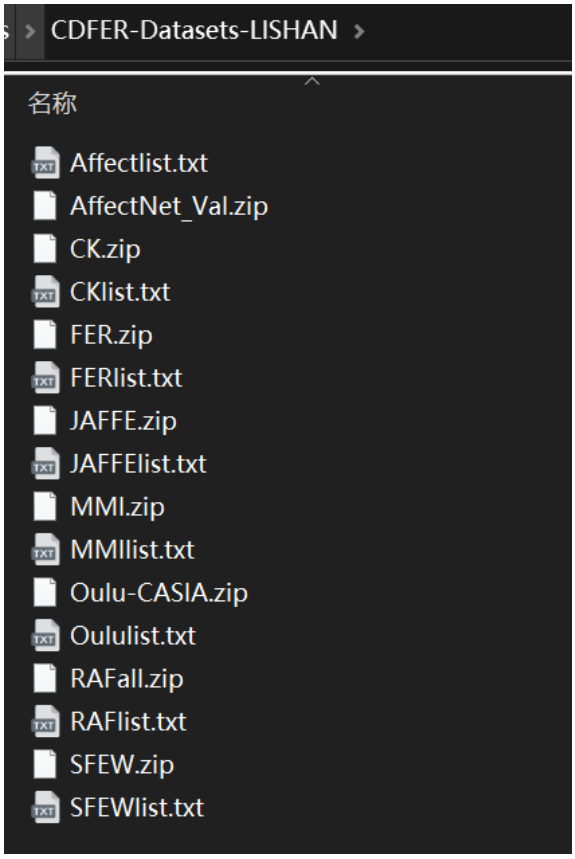
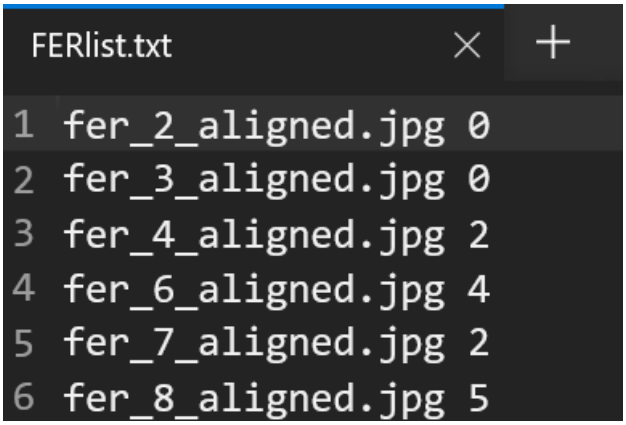
JDMAN 论文复现报告

数据集

网上找不到RAF-DB 2.0数据，通过松岭师兄要到了英建师兄的微信，他将使用的数据集全部发给了我，且**建议我使用vgg网络以获得更稳定的结果**



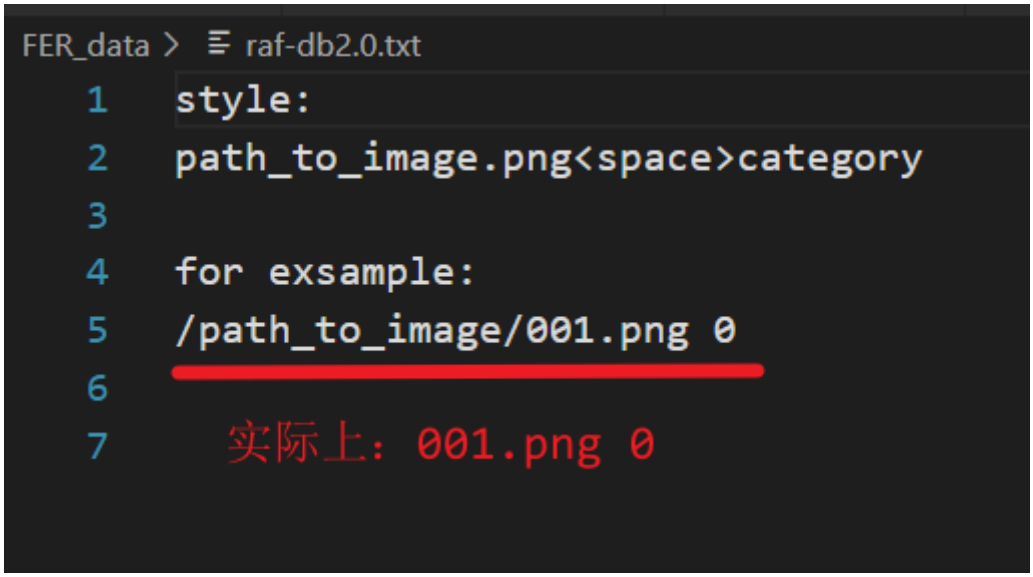


训练情况

- 资源：1卡GPU，A30，20GB（from学校高性能服务中心）
- 和论文实验同样的配置，训练vgg网络：用时8h
- epoch减半，训练resnet网络：用时4h

代码问题

- 项目给出的 数据集示例 和 实际数据集 格式不一致。更改了dataset.py的代码。



- ResNet50和VGG16的output的维度不一致，但是代码层面没有设置相关的命令行参数入口，只能从model入参那边改数字（resnet是3072,vgg是4096）

```python
def create_model(args):
    state = None

    model_creators = get_catalogue() # 获取可用模型目录

    assert args.model in model_creators # 确认参数中的模型在可用目录里

    model = model_creators[args.model](args) # model结构

    if args.MI == True:
        # optimize mutual information by adversarial learning according to
        # Self-supervised representation learning from multi-domain data
        adv_model = AdversarialNetwork(3072, 512, args.n_epochs * 112) # for resnet50
        # adv_model = AdversarialNetwork(4096, 512, args.n_epochs * 112) # for vgg16

    if args.resume: # 模型恢复
        save_path = os.path.join(args.save_path)  # 模型保存目录
        checkpoint = torch.load(save_path)

        model.load_state_dict(checkpoint['model']) # 模型参数
        state = checkpoint['state'] # state参数
```

- vgg架构没有使用adaptation layer，跟resnet不一致

- test_only阶段，没有对应的导入模型参数的代码

```python
def resnet50_with_adlayer(args):
    """Constructs a ResNet-50 model.
    Args:
        pretrained (bool): If True, returns a model pre-trained on ImageNet
    """
    if args.pretrained:
        # zhw added this part for test_only
        model = ResNet_with_ADlayer(Bottleneck, [3, 4, 6, 3], args)
        if args.test_only:
            trained_dict = torch.load(args.pretrained)
            # del trained_dict["state"]
            trained_dict_model = trained_dict["model"]
            model.load_state_dict(trained_dict_model)
            return model

        pretrained_dict = torch.load(args.pretrained)
        model_dict = model.state_dict()

        keys = deepcopy(pretrained_dict).keys()

        for key in keys:
            if key not in model_dict:
                print(key)
                del pretrained_dict[key]

        model_dict.update(pretrained_dict)
        model.load_state_dict(model_dict)

        return model
```

- 当仅测试阶段(test_only)的时候，还是跑了三次实验，但是test_only阶段只跑一次就够了。可优化。

困惑

- 论文3.4.1（SML with common centers）提出的损失函数，在代码()中好像没有使用上
- 在代码中，图中这部分的求法是使用Entropy(softmax)来近似P(h,d)？这个逻辑并不理解

$$\min_{\theta}\max_{\theta_m} L_{MI} = \mathbb{E}_{P_\theta(\mathbf{h},\mathbf{d})}\left[logQ_{\theta_m}(\mathbf{d}|\mathbf{h}) - logQ(\mathbf{d})\right].$$

> 以上疑惑均已发送给英建师兄，由于师兄出差，下周返深才能答复，故同样以报告形式整理于此

实验结果分析

论文实验结果

Table 1: Comparison of our method with state-of-the-arts (%). (bold: best, underline: second best)

Methods	Source	JAFFE	MMI	Oulu-CASIA	CK+	AffectNet	FER2013	Average Accuracy
Da et al. [6]	BOSPHORUS	36.20	-	-	57.60	-	-	-
ICID [12]	RAF-DB	-	64.70	-	84.50	-	-	-
DETN [15]	RAF-DB	57.75	66.05	-	78.83	-	52.37	-
SPWFA-SE [19]	RAF-DB	-	65.63	-	81.72	-	48.68	-
gACNN [20]	RAF-DB	-	59.51	50.31	81.07	-	-	-
CNN+MMD [18]	RAF-DB2.0	58.64	65.80	60.14	82.44	48.76	56.54	62.05
SAFN [35]	RAF-DB2.0	60.56	70.31	60.21	85.19	48.46	58.10	63.81
ECAN [18]	RAF-DB2.0	61.94	69.89	63.97	86.49	51.84	58.21	65.39
AGRA [34]	RAF-DB2.0	62.44	70.03	63.02	87.95	**52.69**	58.57	65.78
SWD [14]	RAF-DB2.0	64.32	69.89	62.44	85.28	50.34	58.48	65.13
CDANs [21]	RAF-DB2.0	64.79	66.05	58.65	81.55	52.31	58.05	63.57
Baseline	RAF-DB2.0	53.52	67.76	57.40	82.28	49.20	51.24	60.23
JDMAN	RAF-DB2.0	**68.54**	**71.88**	**64.38**	**88.51**	52.54	**58.63**	**67.41**

本次实验结果

Method	JAFFE	MMI	CK+	FER2013
Original JDMAN (resnet with adlayer)	68.54	71.88	88.51	58.63
my vgg (without adlayer)	52.582	67.045	88.188	52.076
my_resnet (half epoch)	46.009	63.636	89.887	55.017

原因分析

1. 源码中，CK+ 和 JAFFE 在训练和测试的时候是不同的，但由于英建师兄发来的数据集中并未对 CK+ 和 JAFFE 有更具体的分类，所以这里我都用了同一个数据集。可能训出来的模型会因训练数据的区别而有所差异。

YingjianLi add sh files

Code Blame 34 lines (34 loc) · 1.23 KB

```
python -u main.py \
    -shuffle \
    -aug \
    -model resnet50_with_adlayer_all \
    -pretrained your_path_to_pretrained_models/resnet50.pth \
    -train_list0 path_to_the_training_data/raf-db2.0.txt \
    -train_list1 path_to_the_training_data/raf-db.txt \
    -test_list1 path_to_the_testing_data/raf-db.txt \
    -test_list2 path_to_the_testing_data/affectnet.txt \
    -test_list3  path_to_the_testing_data/fer2013.txt \
    -test_list4 path_to_the_testing_data/ck+_testing_only.txt \
    -test_list5 path_to_the_testing_data/mmi.txt \
    -test_list6 path_to_the_testing_data/jaffe.txt \
    -test_list7 path_to_the_testing_data/oul-CASIA.txt \
    -test_list8 path_to_the_testing_data/sfew.txt \
    -test_list9 path_to_the_testing_data/balanced_ck+_training_only.txt \
    -test_list10 path_to_the_testing_data/jaffe_add1.txt \
    -save_path path_to_save_your_trained_models \
    -criterion mc_loss_center \
    -MI \
```

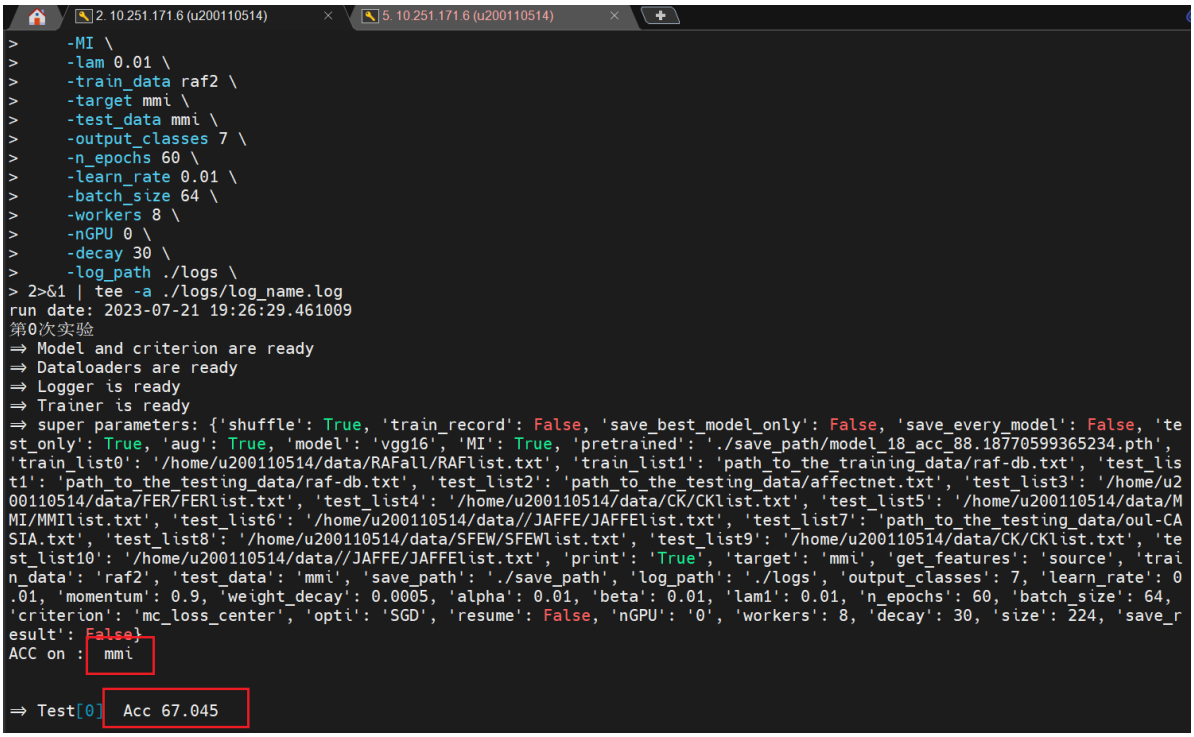
```
def get_test_loader(args): # testing data used in the training stage(different ck+ and jaffe)
    test_domain = args.test_data.split(',')
    test_data = [ ]

    if 'raf' in test_domain:…
    if 'aff' in test_domain:…
    if 'fer' in test_domain:…
    if 'ck+' in test_domain:
        dataset4 = RAFTestSet(args,  args.test_list9)
        data_loader4 = DataLoader(
```

2. 可以看到，vgg在近乎和原文一样的配置下，只在CK+（目标域）作为测试集的情况下取得近似的结果，而在其他数据集取得的结果甚至不如Baseline。也许是adlayer没有使用上?
3. 由于训练成本较高，resnet只用了一半的epoch（30）训，出来的效果只是在CK+（目标域）作为测试集的情况下取得近似甚至超出的结果（有点奇怪，该不会是[分析1]里提到的原因，导致某种程度的过拟合？），其他在jaffe上的效果尤其差，按照论文4.3.2的解释，模型应该学到了constrained和unconstrained数据集之间的domain shift。这个性能有点反常。

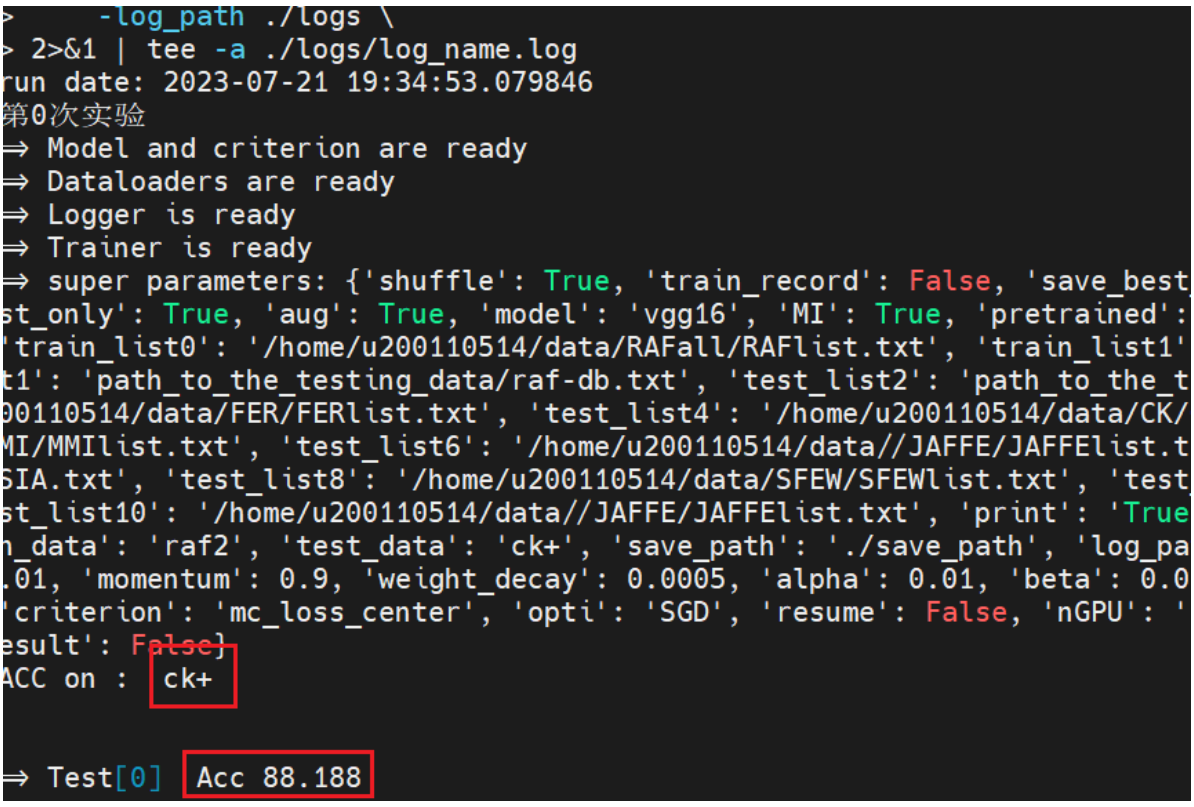
⭐因为师兄一开始推荐我用vgg，所以我vgg是按原文配置训练的。24小时GPU使用时间即将到期，等我下一次申请到时，我会按照相同配置训一个resnet看看问题出在哪。

(部分实验过程截图)

- vgg, test on MMI



- vgg, test on CK+



总结

- 排队申请GPU资源（15h）等待比较长时间。
- 在自己windows系统上试跑，出现了跟num_workers相关的多线程问题，排查后发现，是因为win和linux创建子进程的方式有所不同。所以这只是win上特有的错误。
- resnet和vgg切换时，模型的维度接口需要手动改数字。因为这个问题，几乎把整个项目的代码彻底理解了一遍。
- 代码中很多变量的命名跟论文不一致，有些缩写也没有一些注释辅助理解，只能硬看，较为吃力