神经网络报告 #HW3

吴雨欣 191240060 匡亚明学院

1 代码框架

本实验的代码主要分成三个部分：数据获取及预处理，网络构建，训练及输出。

1.1 数据获取及预处理

在数据预处理方面，我做了以下两个处理方法：

1、由于训练集是彩色的，而测试集是黑白简笔画，因此利用 Grayscale 将训练集转换为黑白图片，并使用 cv 包中的 Canny 函数对其进行边缘检测；

2、由于训练图片和测试图片大小存在不匹配，因此利用 transforms 包中的 Resize 函数将其统一为 32×32。

此前对于训练集，我还采用了 RandomHorizontalFlip 进行随机水平翻转和 RandomRotation 进行随机旋转等图片预处理手段，但效果不如未采取好。一个可能的原因在于训练集本身图片像素较低，原本的数据就很有限，再进行预处理可能导致数据进一步损失。

```python
# load and preprocess data
def loader(train_path, test_path, batch_size):
    trans1 = transforms.Compose([transforms.Grayscale(), transforms.Lambda(lambda x: cv2.Canny(np.array(x), 150, 300)),
                                 transforms.ToPILImage(), transforms.ToTensor()])  ###
    train_imgs = datasets.ImageFolder(train_path, transform=trans1)
    train_iter = torch.utils.data.DataLoader(train_imgs, batch_size=batch_size, shuffle=True)

    trans2 = transforms.Compose([transforms.Grayscale(), transforms.Resize((32, 32)), transforms.ToTensor()])
    test_imgs = datasets.ImageFolder(test_path, transform=trans2)
    test_iter = torch.utils.data.DataLoader(test_imgs, batch_size=batch_size, shuffle=False)

    return train_iter, test_iter
```

图 1: load and prepossess data

1.2 网络构建

网络的构建采用了 DANN(Domain-Adversarial Training of Neural Networks)。如图，整个模型可以分成三个部分，feature extractor, label predictor 和 domain classifier。

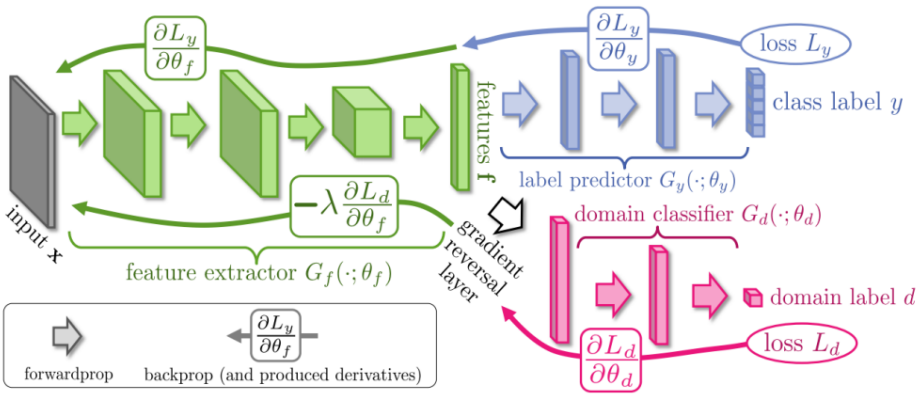

图 2: DANN Structure

对应于上述 DANN 模型的三个部分，分别定义三个网络，feature_extractor, label_predictor 和 domain_classifier:

feature_extractor: 沿用 HW2 中的网络，由 5 个 Stage 组成，每个 Stage 包括 3 个 Conv2d，1 个 MaxPool2d 及 1 个 Dropout。

label_predictor: 由 3 个全连接层组成，激活函数为 ReLU。

```
68  class label_predictor(nn.Module):
69      def __init__(self):
70          super().__init__()
71          self.linear = nn.Sequential(nn.Linear(256, 1024), nn.ReLU(), nn.Linear(1024, 1024), nn.ReLU(),
72                                      nn.Linear(1024, 9))  ###
73
74      def forward(self, src):
75          return self.linear(src)
```

图 3: label_predictor

domain_classifier: 有 3 个全连接层，BatchNormld，激活函数为 ReLU。

```
78  class domain_classifier(nn.Module):
79      def __init__(self):
80          super().__init__()
81          self.linear = nn.Sequential(nn.Linear(256, 512), nn.BatchNorm1d(512), nn.ReLU(),
82                                      nn.Linear(512, 512), nn.BatchNorm1d(512), nn.ReLU(),
83                                      nn.Linear(512, 1))  ###
84
85      def forward(self, x):
86          return self.linear(x)
```

图 4: domain_classifier

1.3 训练及输出

此模型训练分成两个部分，feature extractor & label predictor 和 domain classifier。前者的 loss(losses_pre) 由图片分类错误率 label_loss 和 domain classifier 对图片来源的分类错误率 dis_loss 组成，eta 为系数，即：

$$losses\_pre = label\_loss - eta \times dis\_loss$$

后者 (losses_cla) 则使用 BCEWithLogitsLoss 作为损失函数。

训练 150 轮，将 losses_pre 和 losses_cla 的变化情况绘制再同一张图中比较，易知 label predictor 的 loss 按照期望在不断下降，但 domain classifier 的 loss 在 25 轮次后基本就没有变化。

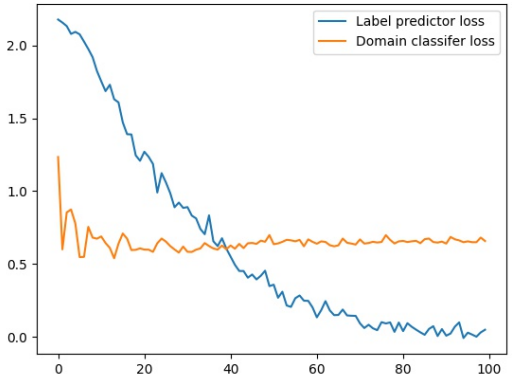

图 5: Result

关于预测的准确率，在目前 kaggle 上的 30% 数据上表现为 46% 左右，效果一般。

2 without/with DaNN 对比图

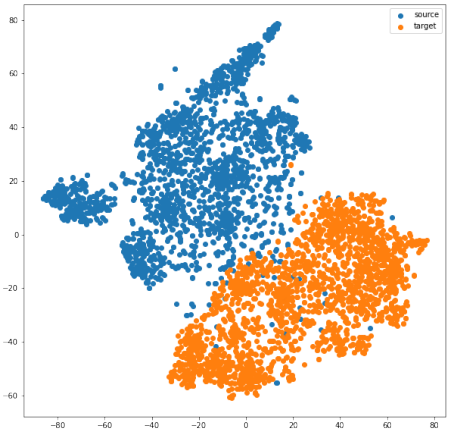

图 6: Without Dann

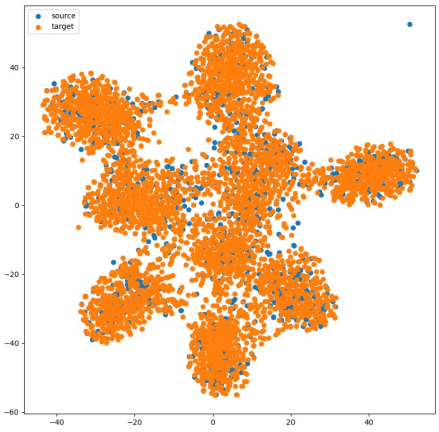

图 7: With Dann

如图，在没有使用 DANN 时，训练图像和测试图像的分布是完全分开的；而加上 DANN 之后，两个部分重合度很高，9 个类别的分类也十分清晰，整体效果不错，这也表明了 DANN 的科学合理性。