HW2

171098160 王慧敏 商学院金融工程专业

一、 模型简述

1、方法：

采用 Domain Adversarial Training，解决 Source Data（真实图片）与 Target Data（手绘图片）任务相同（分类）但数据分布不同的问题。

网络结构有三大部分：特征提取器 Featrue Extractor、标签分类器 Label Predictor、域分类器 Domain Classifier。Feature Extractor 采用卷积网络，输入图像来提取特征，输出特征既要提取有利于分类，同时要将 Source Data 和 Target Data 的域特征去掉，使得 Source Data 和 Target Data 经过 Feature Extractor 的输出分布相似。Label Predictor 输入 Feature Extractor 得到的 Feature 来进行分类。Domain Classifier 输入 Feature Extractor 得到的 Feature 来判断 Feature 是来自 Source Data（真实图片）还是 Target Data（手绘图片）。

具体实现方法是在每一个 batch 中，

- 首先固定 Feature Extractor，训练 Domain Classifier。Domain Classifier 希望正确分类（真实 or 手绘），最小化域分类的 Loss（二分类问题采用 BCEWithLogitsLoss），因此在反向传播的时候采用梯度下降。
- 其次，固定 Domain Classifier，训练 Feature Extractor 和 Label Predictor。Feature Extractor + Label Predictor 希望正确分类（哪一类物品），最小化 Label 的 Loss（多分类问题采用较差熵 Loss），因此在反向传播的时候，对 Feature Extractor 和 Label Predictor 采用梯度下降。同时，Feature Extractor 还希望消除 Source Data 和 Target Data 的域特征，骗过 Domain Classifier 让其无法正确分类，因此要最大化上一个步骤中 Domain Classifier 的 Loss，在反向传播的时候，对 Feature Extractor 采用梯度上升。具体方法是，定义第二步的 Loss 为（相减相当于梯度上升）：

$$Loss = Label\ Predictor\ Loss - \lambda * Domain\ Classifier\ Loss$$

重复，采用 Adam 优化，训练 150 个 epoch。

在开始训练之前，对数据进行预处理。
如图 1 所示，Target Data 手绘图片基本为黑白简笔画，只勾勒物品的边缘，因此 Source Data 真实图片中的颜色、具体内部细节没有用处。为了方便训练，可以对 Source Data 用 cv2 的 Canny 进行边缘检测，保留其轮廓，让 Source Data 和 Target data 尽量接近一点。Canny 只能传入灰度图，因此需要将 Target Data 转换为灰度图（为了统一网络输入，Target Data 也要转成灰度图），颜色信息对于类别判断（特别是 Target Data）不起到重要作用，所以这么操作并不会对准确率造成大的影响。Canny 变化的结果如图 2 所示。

图 1：Source Data 和 Target Data

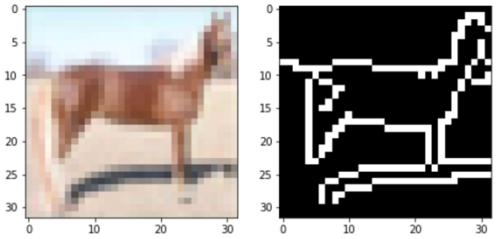

图 2：Source Data 原图和 Canny 变化之后的图片

由于对 Source Data 采用 Canny 变化，Source Data 输入的大小变为 1*32*32。
由于许多 Pre-train 网络要求输入为 RGB 图像，且输入像素要求较高、卷积层输出维度较高（例如 VGG16 要求输入为 3*224*224，卷积层最终输出为 512*7*7），再接入 Domain Classifier 和 Label Predictor 需要 Fine-Tune 的参数仍然很多，且本题目 Source 和 Target Data 的像素都比较低（分别为 32*32、28*28），因此不考虑采用接入 Pre-train Model 进行 Fine-Tune 的方法，而是直接搭建网络。
由于卷积层需要多次池化，将输入放大，resize 为 1*64*64。同时通过旋转、翻转对输入进行数据增强。

2、网络构架如下：

- Feature Extractor 采用卷积网络，输入图像来提取特征，既要提取有利于分类的特征，同时要将 Source Data 和 Target Data 的域特征去掉，使得二者经过 Feature Extractor 的输出分布相似；
- Label Predictor 采用全连接网络，输入 Feature Extractor 得到的 Feature 来判断类别；
- Domain Classifier 采用全连接网络，输入 Feature Extractor 得到的 Feature 来判断 Feature 是来自 Source Data（真实图片）还是 Target Data（手绘图片）。

Feature Extractor：

```
FeatureExtractor(
  (conv): Sequential(
    (0): Conv2d(1, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (5): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): ReLU()
    (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (8): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (10): ReLU()
    (11): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (14): ReLU()
    (15): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (16): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (17): ReLU()
    (18): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (19): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (20): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (21): ReLU()
    (22): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (23): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (24): ReLU()
    (25): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
)
```

Label Predictor：

```
LabelPredictor(
  (classifier): Sequential(
    (0): Linear(in_features=2048, out_features=512, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.2, inplace=False)
    (3): Linear(in_features=512, out_features=256, bias=True)
    (4): ReLU()
    (5): Dropout(p=0.2, inplace=False)
    (6): Linear(in_features=256, out_features=9, bias=True)
  )
)
```

Domain Classifier：

```
DomainClassifier(
  (classifier): Sequential(
    (0): Linear(in_features=2048, out_features=512, bias=True)
    (1): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): Linear(in_features=512, out_features=256, bias=True)
    (4): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU()
    (6): Linear(in_features=256, out_features=256, bias=True)
    (7): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU()
    (9): Linear(in_features=256, out_features=128, bias=True)
    (10): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (11): ReLU()
    (12): Linear(in_features=128, out_features=1, bias=True)
  )
)
```

3、训练结果

从图 3 中可以看出，随着训练 epoch 的增加，Label Predictor 对于 Source Data 的分类准确率逐步上升，在结束 150 个 epochs 后，最终准确率达到 98.73%。Feature Extractor & Label Predictor 的 Loss 以及 Domain Classifier 的 Loss 均随着 epochs 的增加而逐步下降。

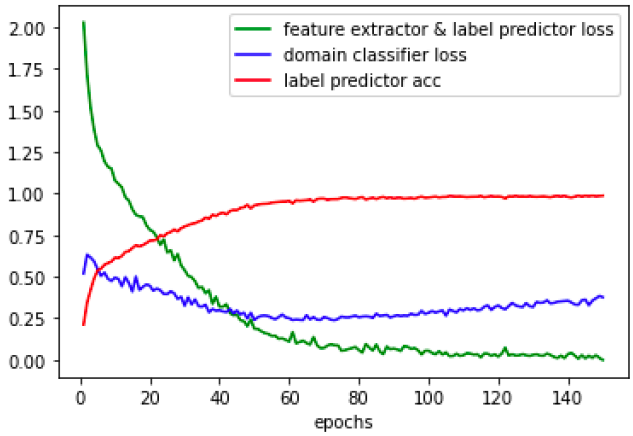

图 3: Domain Adversarial Neural Network 训练

二、 真实图片以及手绘图片通过没有使用 Domain Adversarial Training 的 Feature Extractor 的 Domain 分布图

重新训练一个没有使用 Domain Adversarial Training 的网络，网络结构去掉上述 Domain Adversarial Training 的 Domain Classifier 层，保留 Feature Extractor 和 Label Predictor。训练只采用带标签的 Source Data。输入采用相同的变换，同样训练 150 个 epochs。
可视化 Source Data 和 Target Data 在经过上述模型训练后的 Feature Extractor 时的分布。根据网络结构，Feature Extractor 输出的特征是 2048 维，为了可视化，采用 TSNE 方法降维到 2 维（TSNE 首先运用 PCA 的方法，然后再映射到 2 维）。为了节省运算时间，Source Data 和 Target Data 各随机取 1024 个数据点进行展示（取 16 个 batch，batch_size = 64）。
从图 4 中可以明显看出，Source Data 和 Target Data 在经过 Feature Extractor 后得到的特征分布差异很大。

三、 可视化真实图片以及手绘图片通过使用 Domain Adversarial Training 的 Feature Extractor 的 Domain 分布图。

采用之前训练的 Domain Adversarial 模型。
可视化 Source Data 和 Target Data 在经过 Feature Extractor 时的分布。Feature Extractor 输出的特征同样是 2048 维，为了可视化，同样采用 TSNE 方法降维到 2 维（TSNE 首先运用 PCA 的方法，然后再映射到 2 维）。为了节省运算时间，Source Data 和 Target Data 各随机取 1024 个数据点进行展示（16 个 batch，batch_size = 64）。
从图 5 中可以看出，Source Data 和 Target Data 在经过 Feature Extractor 后得到的特征分布没有明显差异。

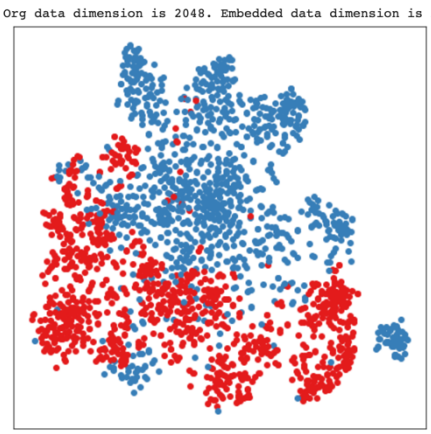

图 4: 非 Domain Adversarial Training Source 和 Target data 的特征分布（Source：红色；Target：蓝色）

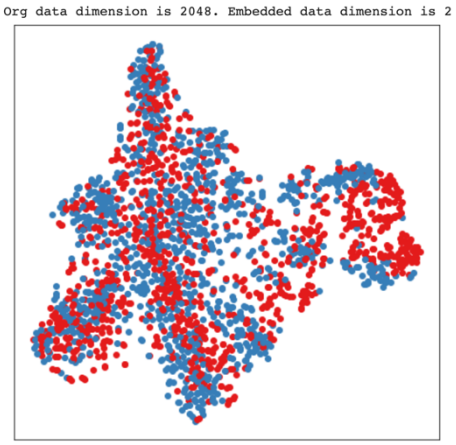

图 5: Domain Adversarial Training Source 和 Target data 的特征分布（Source：红色；Target：蓝色）