HW2 实验报告：基于 DaNN 的图片分类

211240074　余今　匡亚明学院脑科学与人工智能方向

1　实验内容（model.py）

1.1　　模型理论架构

本次实验使用域对抗迁移神经网络（Domain adversarial Neural Network）来构建模型，理论模型架构如图 1 所示。主要包含三部分，分别为特征提取器（Feature Extractor）、类别预测器（Label Predictor）和领域判别器（Domain Classifier）。

其中，特征提取器和类别预测器构成一个完整的前馈神经网络。在训练过程中，会在特征提取器后接一个领域判别器，并通过损失函数实现梯度反转层连接。在训练的过程中，对来自源域（Source Domain）的带标签数据，网络不断最小化类别预测器的损失 loss L_y，从而较好得实现分类任务。对来自源域和目标域（Target Domain）的全部数据，网络不断最大化领域判别器的损失 loss L_d，目标是使得领域判别器无法区分经过了特征提取器的样本数据是来自目标域还是源域，从而保证特征提取器取得的特征与具体的域无关。

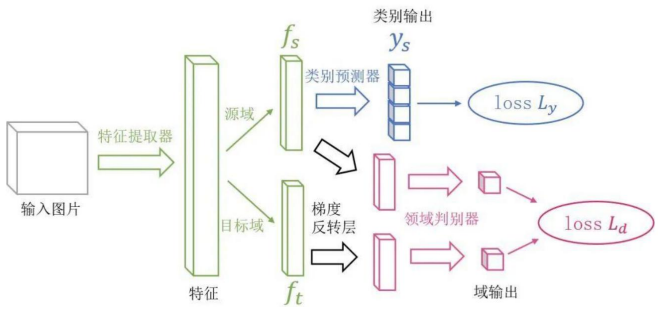


图 1 DaNN 模型理论架构

1.2　　实验中的模型架构

在实验中，我构建了特征提取器（Feature Extractor）、类别预测器（Label Predictor）和领域判别器（Domain Classifier）三个神经网络模型。

1.2.1　　特征提取器（Feature Extractor）

特征提取器的作用为将数据映射到特定特征空间，传入类别预测器和域判别器基本结构为卷积神经网络，由 5 个 cnn_block 组成，cnn_block 结构如下所示，包括 Conv2d、BatchNorm2d、ReLU 和 MaxPool2d。其中，为了防止过拟合，在第 2、3 和 4 个 block 中加了 Dropout。

```
def cnn_block(input_channels, output_channels, kernel_size=3, stride=1, padding=1, inplace=True, pool_size=2, drop=False):
    layer = [
        nn.Conv2d(input_channels, output_channels, kernel_size=kernel_size, stride=stride, padding=padding),
        nn.BatchNorm2d(output_channels),
        nn.ReLU(inplace=inplace),
        nn.MaxPool2d(pool_size)
    ]
    if drop:
        layer.append(nn.Dropout(0.5))
    return nn.Sequential(*layer)
```

图 2 cnn_block 结构

```python
class FeatureExtractor(nn.Module):
    def __init__(self):
        super(FeatureExtractor, self).__init__()
        self.cnn1 = cnn_block(1, 128)  # (1, 32, 32) -> (128, 16, 16)
        self.cnn2 = cnn_block(128, 256, drop=True)  # (256, 8, 8)
        self.cnn3 = cnn_block(256, 512, drop=True)  # (512, 4, 4)
        self.cnn4 = cnn_block(512, 1024, drop=True)  # (1024, 2, 2)
        self.cnn5 = cnn_block(1024, 512)  # (512, 1, 1)
```

图 3 FeatureExtractor 结构

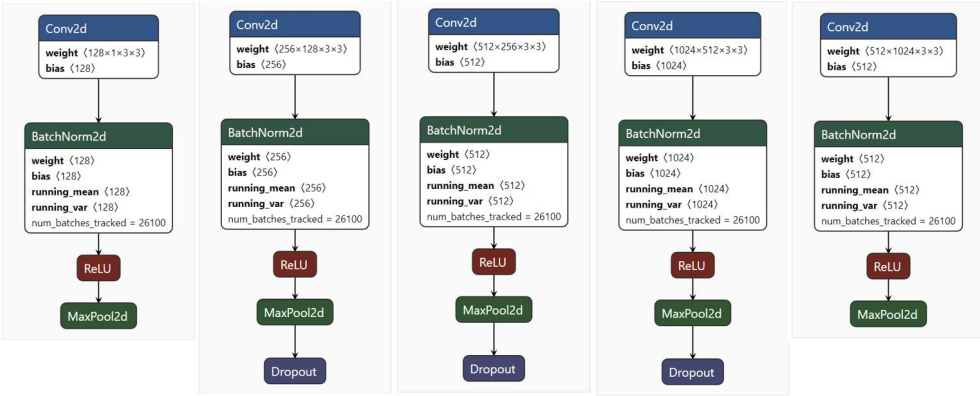


图 4 Neuron 可视化 Feature Extractor

1.2.2 类别预测器（Label Predictor）

类别预测器的作用为对 Feature Extractor 映射后的样本数据进行分类，判断样本类别 label，其结构较为简单，数据在经过 Flatten 后经过两组 Linear + ReLU + Dropout 后再经过一次 Linear，得到类别向量。

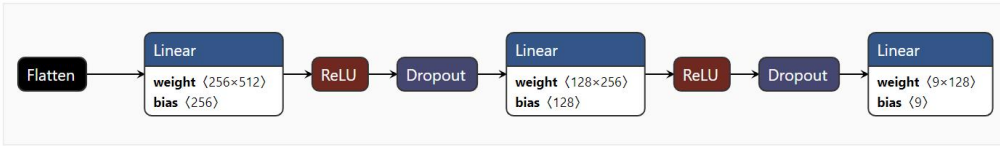
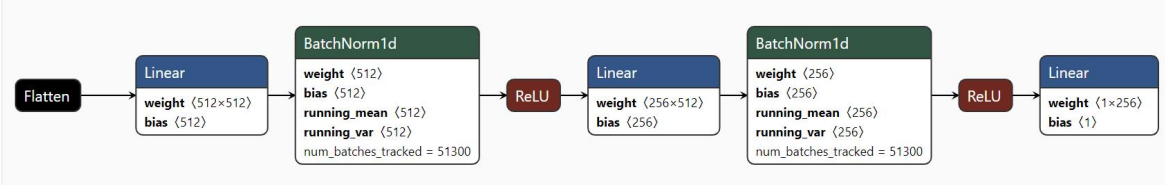


图 5 Neuron 可视化 Label Predictor

1.2.3 领域判别器（Domain Classifier）

领域判别器的作用为对来自源域和目标域的特征空间的数据进行分类，判断数据来自哪个域，结构与类别预测器类似较为简单，三个 Linear 中间插入两组 BatchNorm1d + ReLU，得到领域标签。



1.3 训练过程及优化操作

1.3.1 数据预处理及增强

对来自源域和目标域的数据，将其转化为尺寸相同的灰度图，然后转成 Tensor 格式并进行正则化。除此以外，在 mix 后，源域的数据设为 1，目标域的数据设为 0

```python
train_transform = transforms.Compose([
    transforms.Grayscale(),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.5], std=[0.5])
])
```

```python
test_transform = transforms.Compose([
    transforms.Grayscale(),
    transforms.Resize((image_size, image_size)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.5], std=[0.5])
])
```

图 6 对源域与目标域的数据处理

```python
mixed_data = torch.concat([train_data, target_data], axis=0)
domain_label = torch.zeros([train_data.shape[0] + target_data.shape[0], 1]).to(device)
domain_label[:train_data.shape[0]] = 1  # 源域设置
```

图 7 设置领域标签

1.3.2 优化器与损失函数

对于三个模型，优化器均采用效果稳定且均衡的 Adam，lr = 1e-3，weight_decay = 1e-3。

对于标签预测器，损失函数 class_loss_function 采用交叉熵 CrossEntropyLoss；对于领域判别器，损失函数 domain_loss_function 选择用于二分类的二进制交叉熵 BCEWithLogitsLoss；对于模型总损失，采用二者相减来实现领域判别器的梯度反转，并给予 domain_loss_function 权重 lamb = 0.2。

```python
loss = class_loss_function(label_logits, train_label) - lamb * domain_loss_function(domain_logits, domain_label)
```

图 8 模型总损失设置

1.3.3 训练过程

训练用的主要函数为 train_epoch_with_domain，即每一个 epoch 的训练过程。

首先，训练领域判别器（Domain Classifier），增强领域判别器对源域和目标域的识别能力。整体过程较为常规。由于此时还未训练 Feature Extractor，所以要使用.detach()来防止误差继续反向传播。

```python
feature = ExtractorNet(mixed_data)
domain_logits = ClassifierNet(feature.detach())
loss = domain_loss_function(domain_logits, domain_label)
loss.backward()
optimizer_Cla.step()
classifier_loss += loss.cpu().detach().numpy().item()
```

图 9 训练领域判别器（Domain Classifier）

然后训练特征提取器（Feature Extractor）和类别预测器（Label Predictor），过程与上面类似，损失计算采用图 8 的方式。

```python
# Step 2: 训练标签预测器（Label Predictor）和特征提取器（Feature Extractor）：
label_logits = PredictorNet(feature[:train_data.shape[0]])
domain_logits = ClassifierNet(feature)
loss = class_loss_function(label_logits, train_label) - lamb * domain_loss_function(domain_logits, domain_label)
loss.backward()
optimizer_Ext.step()
optimizer_Pre.step()
total_loss += loss.cpu().detach().numpy().item()
```

图 10 训练特征提取器（Feature Extractor）和类别预测器（Label Predictor）

除此之外，作为对照，用同样的方式进行了无领域对抗的训练，具体过程在 train_epoch_without_domain 函数中。对于二者，取 20%训练集作为验证集，均计算了准确率与损失

1.4 模型训练结果

在训练了 450 轮后，模型基本收敛。无领域对抗模型的分类准确率在训练集上达到了 99.44%，在验证集上达到了 20.89%；有领域对抗模型的分类准确率在训练集上达到了 94%，验证集上达到了 74%。将测试集的预测结果上传到 Kaggle 并尝试了加深 Feature Extractor 和加入 Dropout 层等优化后，分类准确率达到了 65.406%

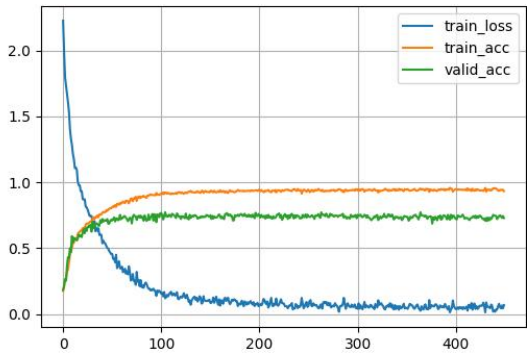


图 11 有领域对抗的模型的损失与准确率变化

res7.csv	Complete (after deadline) · 3h ago	0.6533	0.65406
res6.csv	Complete (after deadline) · 5h ago	0.64021	0.64103
res5.csv	Complete (after deadline) · 6h ago	0.62215	0.62076
res4.csv	Complete (after deadline) · 17h ago	0.60888	0.61236
res3.csv	Complete (after deadline) · 18h ago	0.56407	0.56556
res2.csv	Complete (after deadline) · 18h ago	0.58571	0.5836

图 12 Kaggle 平台部分提交记录

2 可视化 Domain 分布图（eval_model.py）

2.1 无领域对抗训练

读取无领域对抗训练后保存的 extractor_model_without_domain.pth，提取训练集与测试集各 4500 份样本的特征，并将输出结果用 PCA 降维至 2 维，可视化后如图 13 所示。

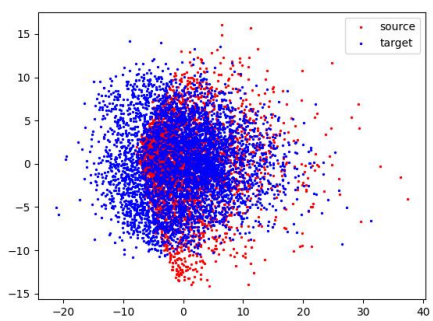


图 13 无领域对抗的特征映射分布

2.2 有领域对抗训练

用同样的方式，读取有领域对抗训练后保存的 extractor_model.pth，提取训练集与测试集各 4500 份样本的特征，并将输出结果用 PCA 降维至 2 维，可视化后如图 14 所示。

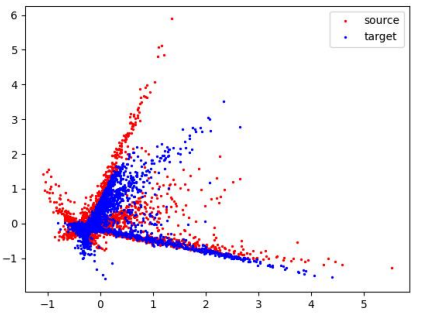


图 14 有领域对抗的特征映射分布

可以看到，增加了领域对抗的训练后，源域与目标域在经过 Feature Extractor 后有了类似的分布。