算分 Project 实验报告

2000017795 陈骏清　200013108 王天宇　2000012967 陆怡杰

June 17th 2022

目录

摘要

行人重识别，即跨不同摄像机的视图匹配行人图像的过程，是视觉监控中的一项重要任务。我们参考的原论文中，作者认为大多数现有模型在很大程度上依赖于颜色外观，并假设行人不会在摄像机视图中换衣服。但是，如果该人（例如犯罪嫌疑人）改变了他的衣服，则在不同地点和不同时间跟踪一个人时，此限制可能会成为重新识别的问题，导致大多数现有方法失败，因为它们是严重依赖颜色外观，因此他们倾向于将一个人与另一个穿着相似衣服的人相匹配。

原论文的工作中，其考虑的是一个人只适度更换衣服的情况，即假设一个人穿的衣服厚度相近，因此在短时间内天气没有发生实质性变化的情况下，人的体形不会发生明显变化。从而基于人物图像的轮廓草图执行跨衣服人员重新识别，以利用人体的形状而不是颜色信息来提取对适度的衣服变化具有鲁棒性的特征。为了在身体轮廓草图上选择更可靠和有区别的曲线图案，我们在深度神经网络中引入基于学习的空间极坐标变换（SPT）层来变换轮廓草图图像，以提取可靠和判别卷积神经网络特征在极坐标空间中。在以下层中应用特定角度提取器 (ASE) 以提取更细粒度的判别角度特定特征。通过改变 SPT 的采样范围，开发了一个多流网络，用于聚合多粒度特征以更好地识别一个人。且论文作者提供其创建的 PRCC 数据集。我们的实验基于该论文的实现，并且做一定的改进，以探究能否通过此方式能获得更好的行人重识别效果。

1 介绍

Re-id 是关联在不相交的摄像机视图中移动的单个人员的过程。随着智能视频监控变得越来越重要，Re-id 变得越来越流行。原论文注意到之前的模型中较好的结果都是基于没有更换衣服的人的图像。因此作者提出如果图像中的人更换衣服，颜色信息具有误导性，运用 Res-Net50 在 PRCC 数据集上的测试结果，显示在一个人穿着不同的衣服后性能的急剧下降。

由此，原论文基于天气在短时间内没有发生实质性变化时，人的体形不会发生显着变化这一假设，在此种情况下，轮廓草图提供了可靠和有效的视觉线索，然而，不同人体的轮廓草图在整体上看起来相似，更重要的是，并非轮廓草图上的所有曲线图案都是有区别的。因此，原论文开发了一种基于学习的空间极坐标变换 (SPT) 来自动选择相对不变、可靠和有区别的局部曲线模式。此外，引入了角度特定提取器（ASE）来模拟每个角度条纹的特征图通道之间的相互依赖关系，以探索细粒度的角度特定特征。最后，针对一组特征学习多流网络，以提取多粒度（即全局粗粒度和局部细粒度）特征，并在他穿着不同时重新识别一个人。

2 论文中的模型

2.1 SPT 极坐标变换模型

此步骤中我们将原图 $U \in R^{H\times W}$ 通过一个可学习的线性变换 $V = \Gamma(U)$ 转化为 $V \in R^{N\times M}$ 具体而言，极坐标转换过程为

$$r = \sqrt{x^2 + y^2} \quad \varphi = \arcsin\frac{y}{x}$$

之后定义各个角度：使 θ_i 为轮廓草图图像的第 i 个采样角度，像素的角度在-π 到 π 之间均匀分布，$i = 0, 1, ..., N$

定义各个极径：$r_j = j \times R/M$

整个转换过程即为：$x^s_{i,j} = r_j \cos\theta_i$, $y^s_{i,j} = r_j \sin\theta_i$,

$u_{h,w}$ 代表原图中（h,w）处的像素值，$v_{i,j}$ 代表转换后的图（i,j）处的像素值，二者之间的转换过程即为：

$$v_{i,j} = \sum_{h=1}^{H}\sum_{w=1}^{W} u_{h,w} \times \lfloor 1 - |x_{i,j}^s - \omega| \rfloor_+ \times \lfloor 1 - |y_{i,j}^s - h| \rfloor_+$$

$i = 0, 1, ..., N$ 以及 $i = 0, 1, ..., M$

之后我们将上述基础模型改为可以学习的模型：将 Γ 转化为基于 $\theta = [\theta_1, \theta_2, ..., \theta_N]$ 的 $V = \Gamma(U; \theta)$，由于采样角度的顺序保留了人的语义结构，这对于人的轮廓建模很重要。为了将采样角度 i 保持在特定范围内和采样角度的顺序，我们将 i 参数化为

$$\theta_i = f(z_i), z_i = \frac{\sum_{k=0}^{i} \lfloor \lambda_k \rfloor_+}{\sum_{k=0}^{N} \lfloor \lambda_k \rfloor_+}$$

$$f(z_i) = (b - a) \times z_i + a$$

这样我们就可以将 θ_i 限制在 $[a, b]$, 以及保证 $\theta_i < \theta_{i+1}$, 之后在学习的过程中不断更新 λ_k 以更新各个 θ_i。下面列出各个 gradient 的表达式：

$$\frac{\partial v_{i,j}}{\partial x_{i,j}^s} = \sum_{h}^{H}\sum_{w}^{W} u_{h,w} max(0, 1 - |y_{i,j}^s - h|) \times grad,$$

$$grad = \begin{cases} 0 & |x_{i,j}^s - \omega| \geq 1 \\ 1 & x\omega \geq x_{i,j}^s and |x_{i,j}^s - \omega| < 1 \\ -1 & x\omega < x_{i,j}^s and |x_{i,j}^s - \omega| < 1 \end{cases}$$

$$\frac{\partial z_i}{\partial \lambda_k} = \begin{cases} \frac{\sum_{c=i+1}^{n} \lfloor \lambda_c \rfloor_+}{(\sum_{c=0}^{n} \lfloor \lambda_c \rfloor_+)^2} & if\ \lambda_k > 0 \quad and \quad k \leq i \\ \frac{-\sum_{c=i+1}^{n} \lfloor \lambda_c \rfloor_+}{(\sum_{c=0}^{n} \lfloor \lambda_c \rfloor_+)^2} & if\ \lambda_k > 0 \quad and \quad k > i \\ 0 & otherwise \end{cases}$$

其余梯度公式均较为明显，在此不列出。由此，如果 θ_i 是可学习的，则转换后的图像将从包含更多身份信息的特定区域中采样更多。

2.2 ASE 特定角度提取器模型

此步希望进行利用细粒度的角度特定特征来应对服装变化。在图 1.a 中，利用图像中相应角度范围将特征图分为 B 个角度块，然后对每个进行平均池化。然后不同的特征被输入到不同的 CNN 分支，以学习细化和特定角度的特征。在此过程中，由于不同通道对识别贡献不同，因此通过对通道间依赖关系建模来重新加权通道。由于不同角度块之间通道间依赖性不同，可以使用非共享层，来模拟不同角度块的依赖性。这种方式下有助于更多关注相对不变的曲线模式。

ASE 实现上为首先将特征向量 a_j 通过两层全连接层得到新的特征向量 d_j，同时为了降低本地噪声对于新特征向量的影响，我们将原特征向量和其与新特征向量的点乘向量直接相加得到特征向量 o_j，再将此向量通过卷积层获得图片的细粒度特征，则 ASE 可以表示如下：

$$d_j = \sigma(Q_j \zeta(W_j a_j))$$

$$o_j = a_j + a_j \odot d_j$$

其中，σ，ζ 表示激活函数，$j = 1, 2, ...B$，$W_j \in R^{\frac{L}{r} \times L}$, $Q_j \in R^{L \times \frac{L}{r}}$,$d_j \in R^L$

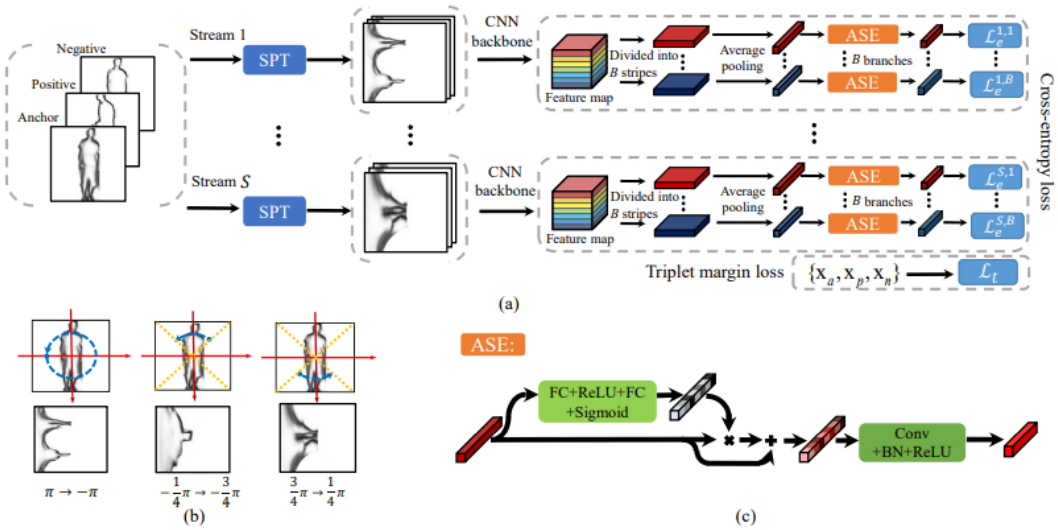

图 1: SPT 与 ASE 以及网络模型

3 网络代码实现细节

3.1 利用 SPT 变换轮廓图

如下左图，我们设计了四个 Stream，分别对应 θ 的取样区间为 $[-\pi, \pi]$，$[-\frac{\pi}{4}, \frac{\pi}{4}]$，$[\frac{3\pi}{4}, \frac{\pi}{4}]$，$[\frac{5\pi}{8}, \frac{3\pi}{8}]$，其中，Stream$[-\pi, \pi]$ 用于提取全局特征，其余三个 Stream 用于提取局部特征，在每个 Stream 中，由 z_i 计算得到 θ_i。如下右图，利用计算得到的 θ_i 和 r_i 确定 SPT 取样点的坐标。

```
for i in range(self.N):
    zi = self.lambdak[:i].sum() / self.lambdak.sum()
    fi = (self.b0 - self.a0) * zi + self.a0
    theta_list0.append(fi)
for i in range(self.N):
    zi = self.lambdak[:i].sum() / self.lambdak.sum()
    fi = (self.b1 - self.a1) * zi + self.a1
    theta_list1.append(fi)
for i in range(self.N):
    zi = self.lambdak[:i].sum() / self.lambdak.sum()
    fi = (self.b2 - self.a2) * zi + self.a2
    theta_list2.append(fi)
for i in range(self.N):
    zi = self.lambdak[:i].sum() / self.lambdak.sum()
    fi = (self.b3 - self.a3) * zi + self.a3
    theta_list3.append(fi)
```

```
for k in range(4):
    for i in range(self.N):
        for j in range(self.M):
            self.grid[k,0,i,j,0]=r_list[j]*torch.sin(theta_lists[k,i])
            self.grid[k,0,i,j,1]=r_list[j]*torch.cos(theta_lists[k,i])
self.grid = self.grid.cuda()
```

图 2: SPT 代码实现

3.2 利用预训练的 CNN 框架提取特征图

这一步我们利用卷积神经网络从 SPT 得到的采样图中提取图片的特征图，我们直接使用了 geffnet 和 torchvisions.model 中的预训练卷积神经网络模型，即 resnet50,efficientnet_b0 和 efficientnet_b1, 经过测试我们发现 efficientnet_b0 效果最好，于是最终训练网络模型时我们使用 efficient_b0 提取图片特征图。

```
if model_name=='resnet50':
    model = resnet50(True)
    stride = 1
    model.layer4[0].downsample[0].stride = stride
    model.layer4[0].conv2.stride = stride
    base = nn.Sequential(
        model.conv1,
        model.bn1,
        model.maxpool,
        model.layer1,
        model.layer2,
        model.layer3,
        model.layer4
    )
    feat=2048
```

```
elif model_name == 'eff_b0':
    backbone = efficientnet_b0(True)
    backbone = nn.Sequential(
        backbone.conv_stem,
        backbone.bn1,
        backbone.act1,
        backbone.blocks,
        backbone.conv_head,
        backbone.bn2,
        backbone.act2,
        nn.Conv2d(1280, 2048, 1)
    )
    base = backbone
    feat = 2048
```

```
elif model_name == 'eff_b1':
    backbone = efficientnet_b1(True)
    backbone = nn.Sequential(
        backbone.conv_stem,
        backbone.bn1,
        backbone.act1,
        backbone.blocks,
        backbone.conv_head,
        backbone.bn2,
        backbone.act2,
        nn.Conv2d(1280, 2048, 1)
    )
    base = backbone
    feat = 2048
```

图 3: 可用于提取特征图的三个预训练卷积神经网络模型

3.3 利用 ASE 提取网络细粒度特征

对特征图中的每一块特征调用 ASE 可以提取图片的细粒度特征，ASE 具体实现细节为首先将特征向量 a_j 通过两层全连接层得到新的特征向量 d_j，同时为了降低本地噪声对于新特征向量的影响，我们将原特征向量和其与新特征向量的点乘向量直接相加得到特征向量 o_j，再将此向量通过卷积层获得图片的细粒度特征，则 ASE 可以表示如下：

$$d_j = \sigma(Q_j \zeta(W_j a_j))$$

$$o_j = a_j + a_j \odot d_j$$

其中，σ，ζ 表示激活函数。

3.4 利用 TripletLoss 计算损失函数

训练网络时我们的输入为 anchor,negative,positive 三张图片，anchor 为基准图片，negative 图片对应的人和 anchor 不同，positive 图片对应的人和 anchor 相同，所以我们的损失函数相应采用 triplet loss，通过最大化 anchor 图片和 negative 图片对应特征向量间距离，最小化 anchor 图片和 positive 图片对应特征向量间距离使得网络具有行人重识别的能力。

4 网络改进措施

4.1 将 SPT 中的 r 也设为可学习参数

原论文中，SPT 只将 θ 设为可学习参数。但显而已见，在利用 SPT 对原图片取样时我们希望尽可能取样点集中在人的身体或边缘上，而不希望取样点位于背景噪声上。并且如果取样

```python
self.ASE_FC_list=nn.ModuleList()
self.ASE_CONV_list=nn.ModuleList()
for i in range(4):
    for j in range(4):
        fc_block=nn.Sequential(
            nn.Linear(2048,1024),
            nn.ReLU(),
            nn.Linear(1024,2048),
        )
        conv_b=nn.Sequential(
            nn.Conv2d(2048,1024,kernel_size=1),
            nn.BatchNorm2d(1024),
            nn.ReLU()
        )
        self.ASE_FC_list.append(fc_block)
        self.ASE_CONV_list.append(conv_b)
self.fc_list=nn.ModuleList()
for i in range(4):
    for j in range(4):
        fc=nn.Linear(1024,num_class)
        self.fc_list.append(fc)
```

```python
def ASE(self,x,index):
    feat=self.ASE_FC_list[index](x.view(x.shape[0],-1))
    x=x+torch.mul(x,F.sigmoid(feat).view(x.shape[0],-1,1,1))
    x=self.ASE_CONV_list[index](x)
    return x
```

图 4: ASE 代码实现

```python
class TripletLoss(object):
    """Modified from Tong Xiao's open-reid (https://github.com/Cysu/open-reid).
    Related Triplet Loss theory can be found in paper 'In Defense of the Triplet
    Loss for Person Re-Identification'."""

    def __init__(self, margin=None):
        self.margin = margin
        if margin is not None:
            self.ranking_loss = nn.MarginRankingLoss(margin=margin)
        else:
            self.ranking_loss = nn.SoftMarginLoss()

    def __call__(self, global_feat, labels, normalize_feature=True):
        if normalize_feature:
            global_feat = normalize(global_feat, axis=-1)
        dist_mat = euclidean_dist(global_feat, global_feat)
        dist_ap, dist_an = hard_example_mining(
            dist_mat, labels)
        y = dist_an.new().resize_as_(dist_an).fill_(1)
        if self.margin is not None:
            loss = self.ranking_loss(dist_an, dist_ap, y)
        else:
            loss = self.ranking_loss(dist_an - dist_ap, y)
        return loss
```

图 5: TripletLoss 代码实现

点过多位于背景噪声上，SPT 得到的结果可能是无意义的，对于网络训练也会产生负面效果。如下图，我们取样时不仅希望取样点的角度集中在人在图片中占比较大的部分，同时对于不同角度我们采样的 r 也应该灵活调整。比如在角度接近 $\frac{\pi}{2}$ 的地方，我们的 r 可以均匀分步在整个图片上部区域，但在角度接近 $\frac{5\pi}{4}$ 的地方，我们的 r 应该集中在图片靠近图片中心的部分，如果 r 取得过大，则此时取样点会落在背景噪声上。

图 6: PRCC 中图片示例

因此，简单地将 r 设为超参数并不是一个明智的选择，将 r 设为可学习参数，使得不同的 θ 对应不同的 r 的分布会改善 SPT 的取样效果。具体代码实现上，我们借鉴了原论文对于 θ 的处理，r 的定义如下：

$$r_i = (r_{max} - r_{min}) * \frac{\sum_{d=1}^{i} \lambda_d}{\sum_{d=1}^{n} \lambda_d} + r_{min}$$

代码实现上，我们只需将 λ_d 设为可学习参数即可。

```
r_list = []
for i in range(self.M):
    zi = self.lambdad[:i].sum()/self.lambdad.sum()
    fi = (self.R_max - self.R_min) * zi + self.R_min
    r_list.append(fi)
```

图 7: 将 r 设为可学习参数

4.2 增加 SPT 的 Stream 数

原论文中只使用了 $[-\pi, \pi]$，$[-\frac{\pi}{4}, \frac{\pi}{4}]$，$[\frac{3\pi}{4}, \frac{\pi}{4}]$ 三个 Stream 进行采样，但我们发现对于一些图片特例，如被拍摄者背书包时，从正面拍摄和从侧面拍摄对于人物轮廓会有巨大的影响，此时如果仍然从人的躯干部分采样，两张图片采样结果会有巨大的差别。

同时我们观察到人物头部的轮廓相对较为固定，因此我们增加了一条 SPT 的 Stream 用于专门在人物头部区域取样，对应 θ 范围为 $[\frac{5\pi}{8}, \frac{3\pi}{8}]$。

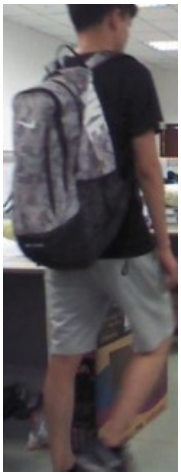

图 8: 躯干部分轮廓有时对于拍摄角度非常敏感

```
for i in range(self.N):
    zi = self.lambdak[:i].sum() / self.lambdak.sum()
    fi = (self.b0 - self.a0) * zi + self.a0
    theta_list0.append(fi)
for i in range(self.N):
    zi = self.lambdak[:i].sum() / self.lambdak.sum()
    fi = (self.b1 - self.a1) * zi + self.a1
    theta_list1.append(fi)
for i in range(self.N):
    zi = self.lambdak[:i].sum() / self.lambdak.sum()
    fi = (self.b2 - self.a2) * zi + self.a2
    theta_list2.append(fi)
for i in range(self.N):
    zi = self.lambdak[:i].sum() / self.lambdak.sum()
    fi = (self.b3 - self.a3) * zi + self.a3
    theta_list3.append(fi)
```

图 9: SPT 增加一条在头部取样的 Stream

4.3 代码实现的细节提升

4.3.1 改进 Average Pooling

原论文中将特征图中每一个 stripe 通过 Average Pooling 得到特征向量，我们在代码实现中采用了 Power-Average Adaptive Pooling，这种自适应的 Average Pooling 效果更好。具体公式为:

$$out = pow(Adaptive - Average(x^p), \frac{1}{p})$$

```python
class GeneralizedMeanPooling(nn.Module):
    r"""Applies a 2D power-average adaptive pooling over an input signal composed of
    The function computed is: :math:`f(X) = pow(sum(pow(X, p)), 1/p)`
        - At p = infinity, one gets Max Pooling
        - At p = 1, one gets Average Pooling
    The output is of size H x W, for any input size.
    The number of output features is equal to the number of input planes.
    Args:
        output_size: the target output size of the image of the form H x W.
                     Can be a tuple (H, W) or a single H for a square image H x H
                     H and W can be either a ``int``, or ``None`` which means the si
                     be the same as that of the input.
    """

    def __init__(self, norm, output_size=1, eps=1e-6):
        super(GeneralizedMeanPooling, self).__init__()
        assert norm > 0
        self.p = float(norm)
        self.output_size = output_size
        self.eps = eps

    def forward(self, x):
        x = x.clamp(min=self.eps).pow(self.p)
        return torch.nn.functional.adaptive_avg_pool2d(
            x, self.output_size).pow(1. / self.p)

    def __repr__(self):
        return self.__class__.__name__ + '(' \
                + str(self.p) + ', ' \
                + 'output_size=' + str(self.output_size) + ')'
```

图 10: 改进 Average Pooling

4.3.2 改进交叉熵

原论文中使用的是正常的交叉熵函数作为损失函数，我们在代码实现时采用了平滑的交叉熵函数，效果更好。平滑化操作即将输入的 one-hot 标签向量中为 0 的分量加上一个小扰动，为 1 的分量减去扰动之和，再计算交叉熵。

4.3.3 改进 Triplet Loss

原论文使用的是 Triplet Loss 计算最终的 loss，在我们的代码实现过程中，我们参考了论文《Circle Loss: A Unified Perspective of Pair Similarity Optimization》，将 Triplet Loss 优化为 Circle Loss，使得最终 s_p 和 s_n 之间区分度更大，网络的重识别能力更强。

```python
class CrossEntropyLabelSmooth(nn.Module):
    """Cross entropy loss with label smoothing regularizer.
    Reference:
    Szegedy et al. Rethinking the Inception Architecture for Computer Vision. CVPR 2
    Equation: y = (1 - epsilon) * y + epsilon / K.
    Args:
        num_classes (int): number of classes.
        epsilon (float): weight.
    """
    def __init__(self, num_classes, epsilon=0.1, use_gpu=True):
        super(CrossEntropyLabelSmooth, self).__init__()
        self.num_classes = num_classes
        self.epsilon = epsilon
        self.use_gpu = use_gpu
        self.logsoftmax = nn.LogSoftmax(dim=1)

    def forward(self, inputs, targets):
        """
        Args:
            inputs: prediction matrix (before softmax) with shape (batch_size, num_c
            targets: ground truth labels with shape (num_classes)
        """
        log_probs = self.logsoftmax(inputs)
        targets = torch.zeros(log_probs.size()).scatter_(
            1, targets.unsqueeze(1).data.cpu(), 1)
        if self.use_gpu: targets = targets.cuda()
        targets = (1 - self.epsilon) * targets + self.epsilon / self.num_classes
        loss = (- targets * log_probs).mean(0).sum()
        return loss
```

图 11: 平滑化交叉熵函数代码实现

```python
class CircleLoss(nn.Module):
    def __init__(self, m: float, gamma: float) -> None:
        super(CircleLoss, self).__init__()
        self.m = m
        self.gamma = gamma
        self.soft_plus = nn.Softplus()


    def forward(self,feat,label) -> Tensor:

        feat=normalize(feat,axis=-1)
        sp, sn = convert_label_to_similarity(feat, label)
        ap = torch.clamp_min(- sp.detach() + 1 + self.m, min=0.)
        an = torch.clamp_min(sn.detach() + self.m, min=0.)

        delta_p = 1 - self.m
        delta_n = self.m

        logit_p = - ap * (sp - delta_p) * self.gamma
        logit_n = an * (sn - delta_n) * self.gamma

        loss = self.soft_plus(torch.logsumexp(logit_n, dim=0) + torch.logsumexp(
            logit_p, dim=0))

        return loss
```

图 12: Circle Loss 代码实现

5 实验结果

论文中的模型、我们的初始模型、最终模型在 prcc 数据集上的结果如下表所示。论文中的模型的准确率数据直接摘自论文。

Methods	Camera A and B (Same clothes)			Camera A and C (Cross-clothes)		
	Rank 1	Rank 10	Rank 20	Rank 1	Rank 10	Rank 20
model in the paper	64.2	92.62	96.65	34.38	77.3	88.05
our original model						
final model	0.33	0.56	0.65	0.19	0.37	0.48

表 1: 在验证集上的测试结果

5.1 结果分析

可以看到，我们的最终模型相较于原论文有所下降。我们分析模型效果下降的原因：在所有改动中，比较欠考虑的是增加取样范围这一项。增加的取样范围可能并不能很好地达到我们希望它做到的效果。我们主观上认为头部是一个在拍摄角度变化时可以保持良好特征的部分，但事实上可能并非如此。

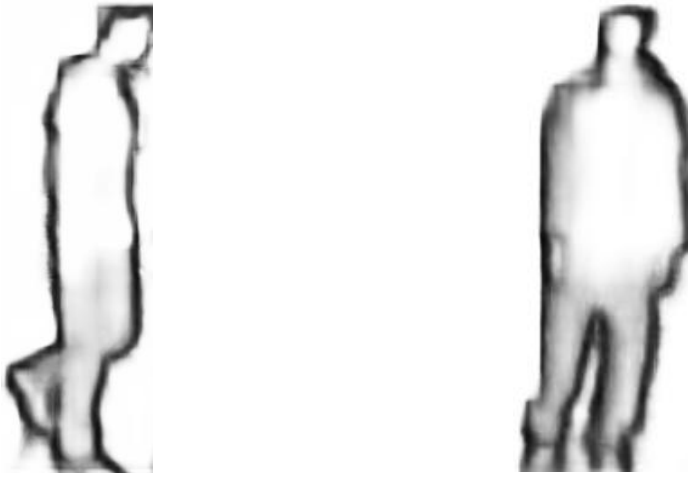

图 13: 头部状态不稳定的例子

当拍摄角度发生改变时，头部虽然在原图变化不大，但投射到极坐标下后的位置会产生比较大的偏移。

增加这一取样范围对模型的扰乱可能大于对局部特征的放大。一种可能的分析是，我们增加的取样范围被包括在了原来的三个取样流之中，这造成的重复取样会在头部特征不显著时产生严重扰乱。而在头部特征足够显著时，原范围可能已经“够用了”，而不需要再额外取样一遍。

5.2 实验方法拓展

除此之外，我们此前还以另一种数据划分方式训练并测试了此模型。具体为，我们不再限制训练集和测试集在人物 id 上的正交性，允许人的一部分照片出现在训练集上，另一部分出现在和测试集上。

这样做的想法是进一步找寻不同因素对实验结果的影响。我们想知道如果一个人此前在训练中出现过，在验证时，这个人以另一个摄像机拍摄的全新样貌出现时，我们的模型能否成功识别。尽管如助教所言，这样“违背了 re-id 问题的初衷”，我们仍然认为这样做不是全无意义的。

在如上的处理下，我们得到如下的结果：

Methods	Camera A and B (Same clothes)			Camera A and C (Cross-clothes)		
	Rank 1	Rank 10	Rank 20	Rank 1	Rank 10	Rank 20
our original model	0.66	0.94	0.97	0.76	0.94	0.98
final model	0.71	0.97	0.98	0.78	0.97	0.99

表 2: 在新验证集上的测试结果

从以上结果我们看到：

首先，在更换了宽松的数据划分方式后，模型的准确率有非常大的提升，这是意料中的结果。

其次，我们发现一个显著的事实：验证集在摄像机 A 和 C（对应衣着变化）上的准确率高于在摄像机 A 和 B（对应衣着不变）上的准确率。对应到原数据集，我们发现，摄像头 A 和 C 的拍摄角度和拍摄背景相比较于摄像头 A 和 B 更接近。此外，相机 B 的背景相比 A、C 更复杂，巨大的背景噪声可能也是造成此结果的原因。

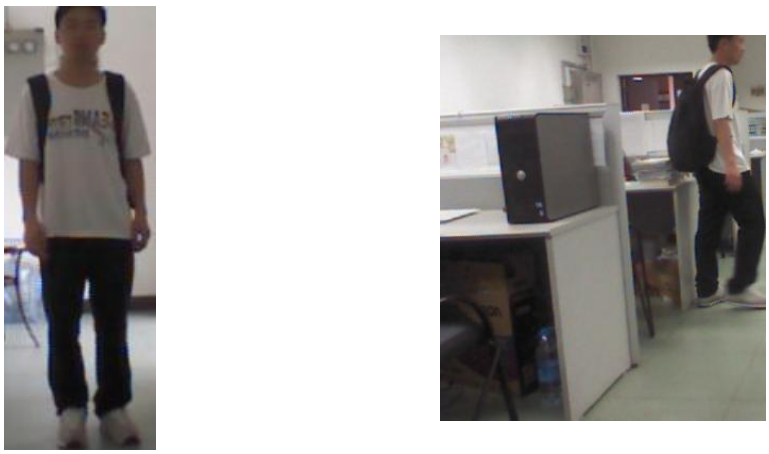

图 14: 背景噪声影响较大的图片（左为摄像机 A，右为摄像机 B）

我们从中发现，尽管我们添加了可学习的极坐标变换，并把极坐标距离（而不仅仅是角度）添加进了可学习参数，但背景噪声的影响并没有因此得到很大的改善。我们更改的模型在对背景噪声的处理上可能仍然有很大的改进之处。

6 未来的工作

6.1 网络超参数的调整

原论文中并没有给出官方代码。此外，由于训练条件有限，我们没有充足的时间和资源逐个调整网络的超参数。一些超参数的设置可能不合理，需要在多次训练中逐步调整。

6.2 增加结果数据

仍然受限制于训练资源，我们没有完成像原论文那样详尽的结果对比。对于我们的各个改进项，应该添加对应论文中各部分工作对结果的影响对比，在此结果对比下分析各个部分的提升效果。

TABLE 6
The rank-1 accuracy (%) of our approach in the ablation study Note that when removing the SPT layers, the model only remains one stream.

Methods	Camera A and C (Cross-clothes)		
	Rank 1	Rank 10	Rank 20
Fixed $\boldsymbol{\theta}$ of SPT	31.05	72.68	86.79
Removing SPT	25.74	70.66	83.08
Removing ASE	28.00	70.89	84.11
Removing $\mathcal{L}_t$	31.39	72.62	85.00
Removing ASE, SPT, $\mathcal{L}_t$	21.95	56.81	73.77
Our full model	**34.38**	77.30	88.05

图 15: 原论文关于各部分工作的影响分析

再如对应各种不同程度的换装，我们模型的最终表现如何。这受限制于 prcc 数据集没有给出换装类型等的标签，我们除了人工没有别的手段获得这些信息。

TABLE 2
The statistics on the test set of the PRCC dataset and the corresponding testing result (%). "Others" means "the changes of bags, shoes, hats, haircut"

Type of clothing change			The number	PCB [19] (RGB)			Our model		
Upper body	Lower body	Others	of identities	Rank 1	Rank 10	Rank 20	Rank 1	Rank 10	Rank 20
√	×	×	23	24.47	78.65	89.83	36.04	81.96	92.67
√	√	×	24	20.14	56.63	76.05	32.67	77.60	89.80
√	√	√	12	9.58	36.39	56.66	19.28	58.03	72.62

图 16: 原论文关于换装类型的影响分析

以上两个分析我们认为较为重要，可以很大程度上体现模型的特化和泛化能力，描述我们模型的特长和缺陷。

7 总结

本次算法设计与分析小班课课程的专题研究，我们利用 prcc 数据集调研并运行了基于轮廓草图的人物重识别算法，并在此基础上进行了自己的改动，以期望达到更好的效果。本次实验给我们积累了研究问题的一般经验，获得了深度学习领域的宝贵知识，锻炼了寻找问题、优化问题、解决问题的能力，并找到了自己知识和技能上的不足，为我们未来的学习和研究铺垫了良好基础。