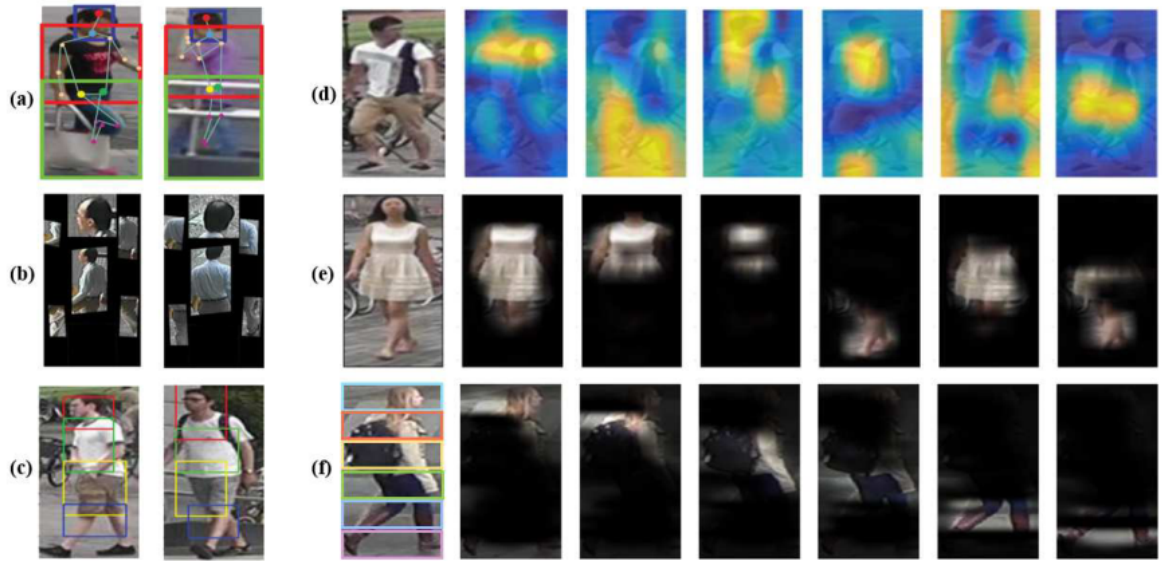
Beyond Part Models: Person Retrieval with Refined Part Pooling(and A Strong Convolutional Baseline)

提出问题：

通过将整体目标分为多个部分进行特征提取，在行人重识别领域已经被证实能够有效提升ReID性能。然而怎么分割更为合理、需不需要额外的标注工作仍然有待探索。

当前的划分策略大致分为两类



1. 基于额外先验信息：利用行人动作估计等额外信息进行分割，但是这需要额外的标注工作，如上图(a)、(b)均为此类工作；
2. 不依赖额外信息：仅靠聚类（c图工作）或注意力机制（d、e图工作）进行目标分割；

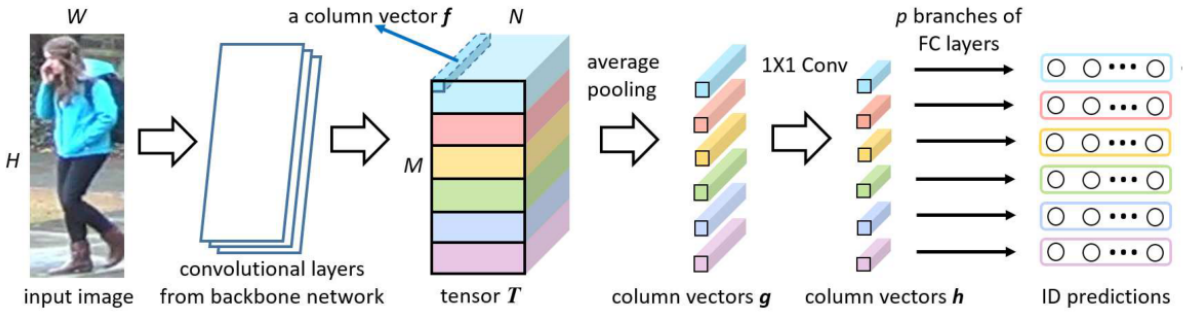
摘要

本文主要由两部分工作构成：

1. 提出了一种简单的分部特征提取网络：PCB(Part-based Conv Baseline)，将输入目标等分后分别进行特征提取
2. 提出了一种细化分割区域的模块：RPP(Refined Part Pooling)，在PCB的基础上通过半监督学习的方式精细调整各区域的划分，从而更加适应行人不同区域的特征

模型结构

1. PCB



骨干网络：

保留ResNet50全局平均池化（GAP）之前的网络结构。

算法步骤：

1.输入图片经过骨干网络，得到张量T，定义沿着通道方向的激活向量为列向量$f(1\times1\times c)$

2.将张量T平均分为p个水平方向，使用平均池化，将p个水平部分在空间上进行下采样，得到p个列向量g_i

3.p个列向量$g_i(i=1,2,\ldots,p)$通过一个$1\times1 Conv$,在通道上降维到256，得到p个列向量$h_i(1\times1\times256)$

4.将每个h_i输入到一个分类器中，分类器由一个全连接层和一个$softmax$构成

5.训练阶段最小化p个id预测的交叉熵损失函数

6.测试阶段，将g或h的p个片段拼接起来形成最终描述符，即$G=[g_1,g_2,\ldots,g_p]$或$H=[h_1,h_2,\ldots,h_p]$

特点：减少ResNet50中conv5_x[0]层中的降采样（经过conv4_x维度为24×8×1024）

	layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
	conv1	112×112	7×7, 64, stride 2				
			3×3 max pool, stride 2				
layer1	conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
layer2	conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
layer3	conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
layer4	conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
		1×1	average pool, 1000-d fc, softmax				
	FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

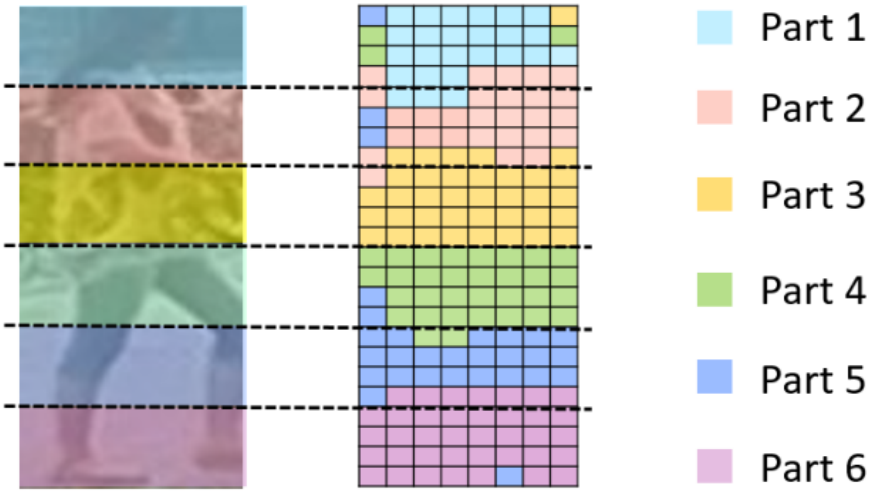
训练：loss为交叉熵损失函数的sum(p个分类器，p个loss)

测试：串联向量g和h作为特征表示

参数设置：
输入图片 384 × 128 × 3
p = 6
T = 24 × 8 × 2048
g = 1 × 1 ×2048
f = 1 × 1 ×256

2. RPP

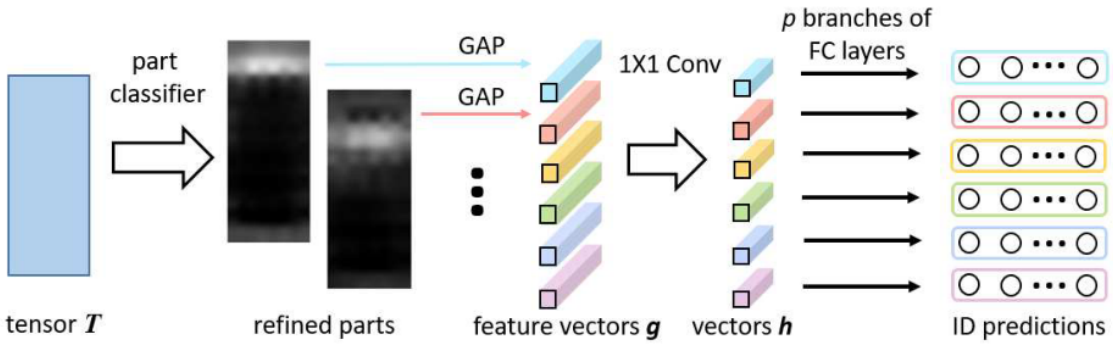
Within-Part 不一致性



计算每个 f 和 g_i 的余弦距离，将 f 归类到最近的部分，得到上图的表格，每个列向量 f 由一个小矩形表示，并以最近部分的颜色绘制，观察到两个现象：

1. 同一个水平部分的大部分列向量 f 聚集在一起
2. 一个水平部分中存在异常值与其他水平部分更相似。

重新定位异常值



RPP根据所有列向量和每个部分的相似性来重新定位异常值。

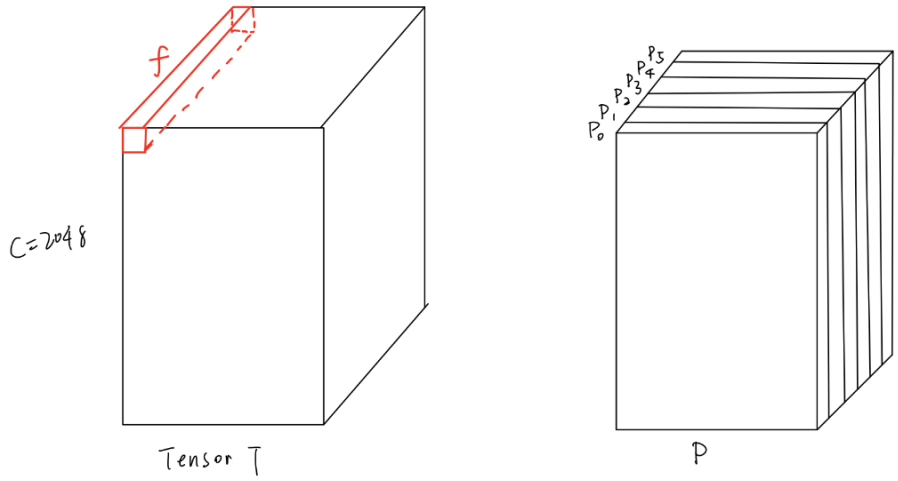
对张量T中的每个列向量f添加一个部件分类器，由一个线性层和softmax函数构成

$$P\left(P_i|f\right)=soft\max\left(W_i^Tf\right)=\frac{\exp\left(W_i^Tf\right)}{\sum_{j=1}^{p}\exp\left(W_j^Tf\right)}$$

$P(P_i|f)$表示f属于P_i部分的可能性。根据$P(P_i|f)$，将f分配给P_i，每一部分$P(P_i|f)$作为采样权重，对所有的列向量f进行采样

$$P_i=\{P(P_i|f)\times f,\forall f\in F\}$$

代码实现



Pi每个元素表示对应位置上的P(Pi|f)，将各层Pi取出与Tensor T相乘

即为 $P_i = \{P(P_i|f) \times f, \forall f \in F\}$

```
# Define the RPP layers
class RPP(nn.Module):
    def __init__(self):
        super(RPP, self).__init__()
        self.part = 6
        add_block = []
        add_block += [nn.Conv2d(2048, 6, kernel_size=1, bias=False)]
        add_block = nn.Sequential(*add_block)
        add_block.apply(weights_init_kaiming)

        norm_block = []
        norm_block += [nn.BatchNorm2d(2048)]
        norm_block += [nn.ReLU(inplace=True)]

        norm_block = nn.Sequential(*norm_block)
        norm_block.apply(weights_init_kaiming)

        self.add_block = add_block
        self.norm_block = norm_block
        self.softmax = nn.Softmax(dim=1)
        self.avgpool = nn.AdaptiveAvgPool2d((1, 1))


    def forward(self, x):  # [batchsize,2048,24,8]
        w = self.add_block(x)   #[batchsize,6,24,8]
        p = self.softmax(w)
        y = []
        for i in range(self.part):
            p_i = p[:, i, :, :]
            p_i = torch.unsqueeze(p_i, 1)  #[batchsize,1,24,8]
            y_i = torch.mul(x, p_i)
            y_i = self.norm_block(y_i)
            y_i = self.avgpool(y_i)     #[batchsize,2048,1,1]
            y.append(y_i)

        f = torch.cat(y, 2)  #[batchsize,2048,6,1]
        return f
```

部分分类的诱导训练

Algorithm 1: Induced training for part classifier

Step 1. A standard PCB is trained to convergence with uniform partition.
Step 2. A p-category part classifier is appended on the tensor $\boldsymbol{T}$.
Step 3. All the pre-trained layers of PCB are fixed. Only the part classifier is trainable. The model is trained until convergence again.
Step 4. The whole net is fine-tuned to convergence for overall optimization.

1. 首先将一个张量 T 均匀分区的PCB训练到收敛。
2. 然后去掉张量 T 后原始的平均池化层，添加一个 p 类部件分类器，根据部件分类器的预测从张量 T 中选取新的部分。
3. 固定已经训练过的PCB的层参数，在训练集上仅仅训练部件分类器，直到达到一个平衡状态。训练中，惩罚部件分类器，直到它进行的划分接近原始的均匀划分，同时部件分类器将相似列向量归类到相同的部件中。
4. 最后，PCB和部件分类器同时更新，通过微调实现整体优化。

基于PCB训练的分类器诱导训练，会收敛的非常快，总共需要10次以上。假如去除第一步的诱导，RPP结构更接近与注意力机制，虽然模型同样会收敛，但是模型性能会显著下降，表明诱导训练优于注意力机制。

实验设置

Datasets and Settings:

- **Datasets**：Market-1501，DuckMTMC-reID，CUHK03
- **setting**：single-query，without re-ranking（因为re-ranking会对mAP的提升效果显著）

Batch size : 64

学习率：前60次0.1，后40次0.01，预训练部分的学习率设为0.1 x 学习率

训练RPP时，训练10次，学习率为0.1

实验结果

变体：

1. 将 p 个 hi 合并平均为一个向量 h ，即 p 个ID损失变为一个。
2. 不改变PCB的结构，最后的分类器共享全连接层参数。

Models	Feature	dim	Market-1501		DukeMTMC-reID		CUHK03	
			R-1	mAP	R-1	mAP	R-1	mAP
IDE	pool5	2048	85.3	68.5	73.2	52.8	43.8	38.9
IDE	FC	256	83.8	67.7	72.4	51.6	43.3	38.3
Variant 1	$\mathcal{G}$	12288	86.7	69.4	73.9	53.2	43.6	38.8
Variant 1	$\mathcal{H}$	1536	85.6	68.3	72.8	52.5	44.1	39.1
Variant 2	$\mathcal{G}$	12288	91.2	75.0	80.2	62.8	52.6	45.8
Variant 2	$\mathcal{H}$	1536	91.0	75.3	80.0	62.6	54.0	47.2
PCB	$\mathcal{G}$	12288	92.3	77.4	81.7	66.1	59.7	53.2
PCB	$\mathcal{H}$	1536	92.4	77.3	81.9	65.3	61.3	54.2
PCB+RPP	$\mathcal{G}$	12288	**93.8**	**81.6**	**83.3**	**69.2**	62.8	56.7
PCB+RPP	$\mathcal{H}$	1536	93.1	81.0	82.9	68.5	**63.7**	**57.5**

1. PCB相较于IDE的提升，表明整合部分信息，可以提高特征的区分能力。
2. RPP可以进一步提升PCB的性能，map相较于rank-1的提升更大，表明RPP在寻找困难匹配上更有用。
3. PCB优于变体1表明，多个loss对于学习有区别力的部分特征很重要。
4. PCB优于变体2表明，共享全连接参数会导致性能下降。

与其他方法进行对比

Group 1:手工设计模型（hand-crafted methods）

Group 2:利用全局特征的深度学习方法（deep learning methods employing global feature）

Group 3:利用局部特征的深度学习方法（deep learning methods employing part features）

*号表示需要额外的辅助part labels

Methods	R-1	R-5	R-10	mAP
BoW+kissme [43]	44.4	63.9	72.2	20.8
WARCA[17]	45.2	68.1	76.0	-
KLFDA[18]	46.5	71.1	79.9	-
SOMAnet[1]	73.9	-	-	47.9
SVDNet[32]	82.3	92.3	95.2	62.1
Triplet Loss [15]	84.9	94.2	-	69.1
DML [40]	87.7	-	-	68.8
Cam-GAN [50]	88.1	-	-	68.7
MultiRegion [34]	66.4	85.0	90.2	41.2
PAR [41]	81.0	92.0	94.7	63.4
MultiLoss [20]	83.9	-	-	64.4
PDC* [31]	84.4	92.7	94.9	63.4
MultiScale [3]	88.9	-	-	73.1
GLAD* [35]	89.9	-	-	73.9
HA-CNN [21]	91.2	-	-	75.7
PCB	92.3	97.2	98.2	77.4
PCB+RPP	**93.8**	**97.5**	**98.5**	**81.6**

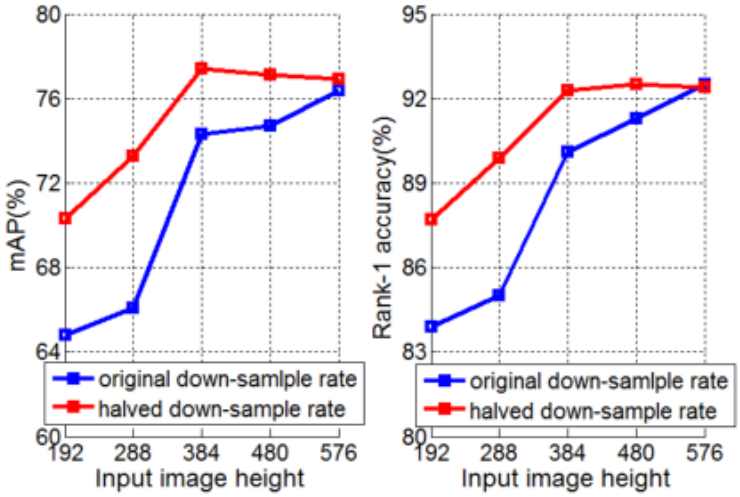
Methods	DukeMTMC-reID		CUHK03	
	rank-1	mAP	rank-1	mAP
BoW+kissme [43]	25.1	12.2	6.4	6.4
LOMO+XQDA [23]	30.8	17.0	12.8	11.5
GAN [47]	67.7	47.1	-	-
SVDNet [32]	76.7	56.8	41.5	37.3
MultiScale [3]	79.2	60.6	40.7	37.0
SVDNet+Era [49]	79.3	62.4	48.7	43.5
Cam-GAN [50]	75.3	53.5	-	-
HA-CNN [21]	80.5	63.8	41.7	38.6
PCB (UP)	81.8	66.1	61.3	54.2
PCB (RPP)	**83.3**	**69.2**	**63.7**	**57.5**

参数

1、输入与采样率

image size 在 192 × 64 到 576 × 192之间的范围, 使用 96 × 32 作为不同组实验之间的间隔；

设置了2个不同的降采样率进行对比，蓝色曲线为原始的降采样率，红色为对半的降采样率。

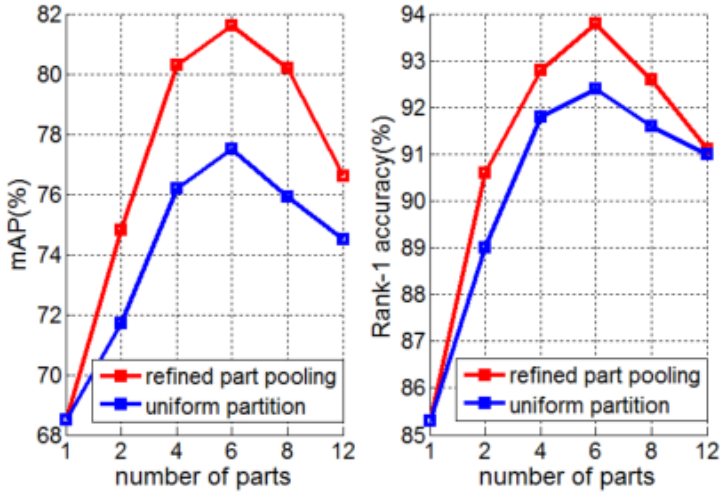


(a) Impact of the size

结论：

- 大的图像尺寸有利于学习到局部特征
- 小一些的 down-sampling rate （ larger spatial size of tensor T）更有效，特别是对于输入的图片尺寸较小时，提升很明显。
- 我们可以看到，使用384×128的图像尺寸并使用对半的down-sampling rate，与使用 576 × 192图像尺寸并使用原始的down-sampling rate达到几乎一样的表现效果

2、分区数



(b) Impact of p

当 p 增加到8或12时，有些refined parts与其它part非常相似，有些可能会坍塌为空part。因此，过度增加的 p 实际上损害了part 特征的识别能力。在实际应用中，我们建议使用 p = 6个parts。

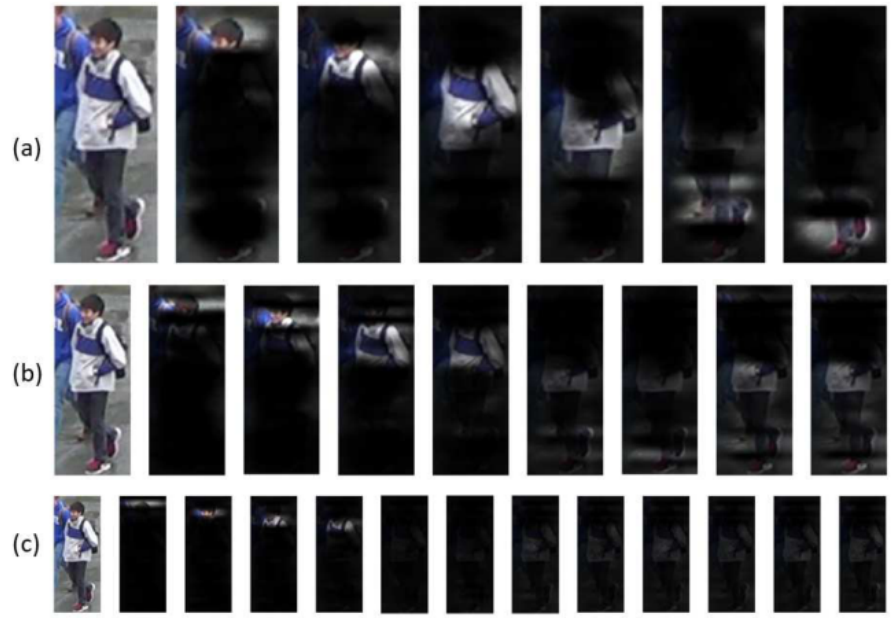


Fig. 6. Visualization of the refined parts under different p values. When $p = 8$ or 12, some parts repeat with others or become empty.

3、诱导训练与注意力机制

Methods	Market-1501		DukeMTMC-reID	
	rank-1	mAP	rank-1	mAP
PAR [41]	81.0	63.4	-	-
IDE	85.3	68.5	73.2	52.8
RPP (w/o induction)	88.7	74.6	78.8	60.9
PCB	92.3	77.4	81.7	66.1
PCB+RPP	93.8	81.6	83.3	69.2

没有PCB预训练的RPP更接近于注意力机制，诱导训练优于注意力机制。

Learning Discriminative Features with Multiple Granularities for Person Re-Identification

摘要

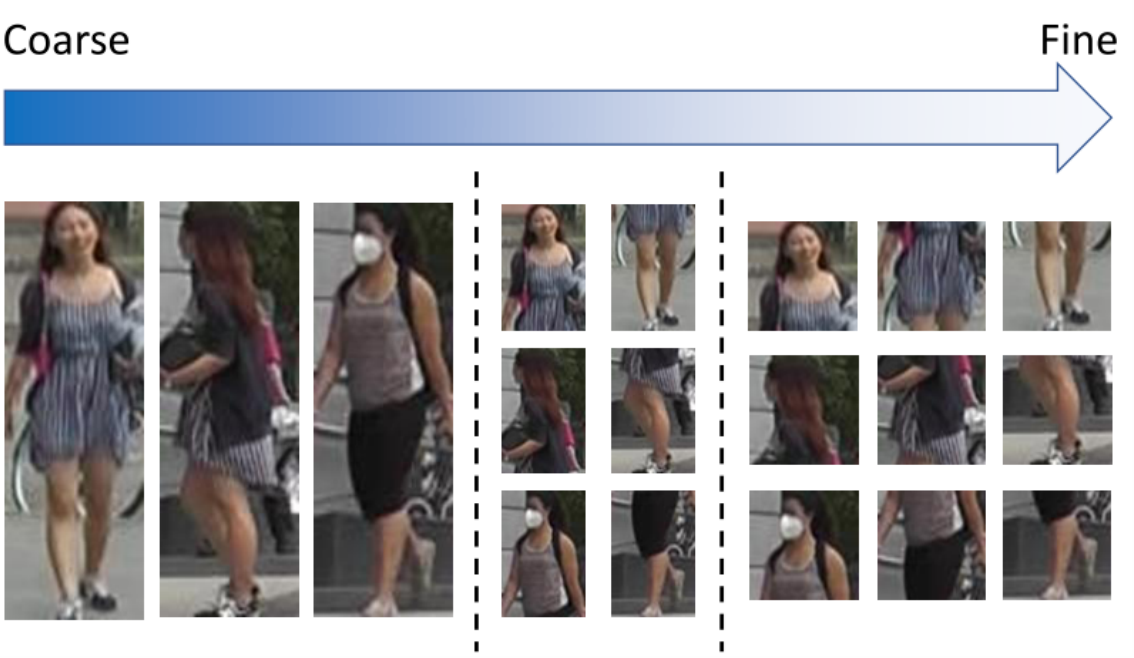
将整体特征与局部特征相结合是提高行人再识别(Re-ID)能力的重要解决方案。以往的基于局部的方法主要是通过定位具有特定语义的区域来学习局部表示，这增加了学习难度，但对方差较大的场景缺乏有效性和鲁棒性。

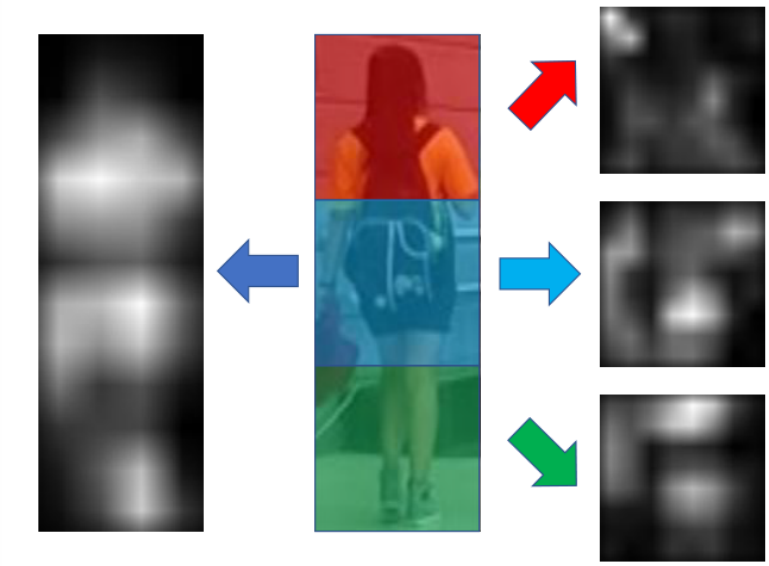
贡献：

本文中提出了一种融合不同粒度的判别信息的端到端特征学习策略。

设计了多分支深度网络体系结构，其中一个分支用于全局特征表示，两个分支用于局部特征表示。

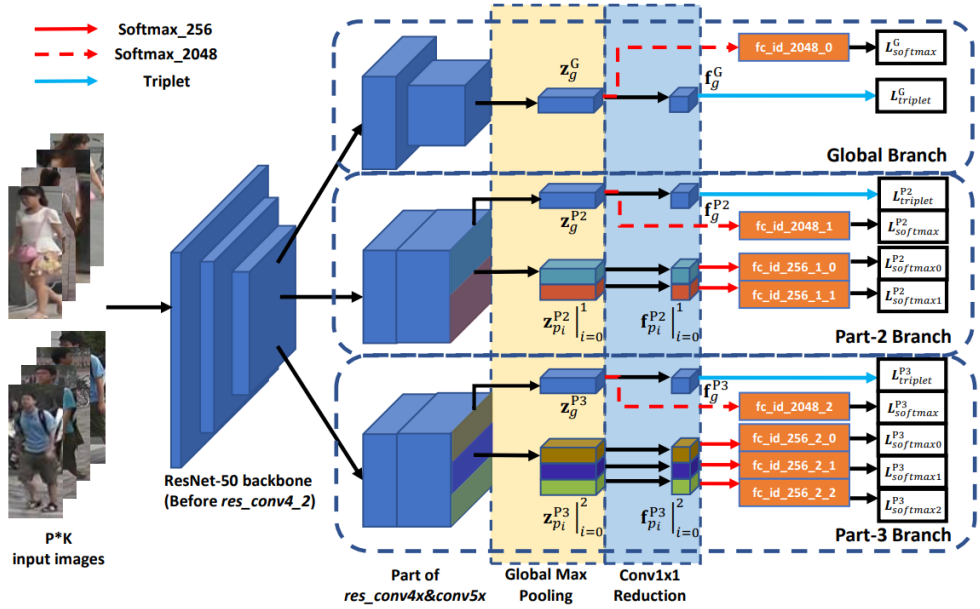
本文没有学习语义区域，而是将图像统一划分为若干条，并改变局部分支的部分数量，得到具有多个粒度的局部特征表示。





MGN

网络结构



骨干网络：ResNet50

	layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
	conv1	112×112	7×7, 64, stride 2				
			3×3 max pool, stride 2				
layer1	conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
layer2	conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
layer3	conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
layer4	conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
		1×1	average pool, 1000-d fc, softmax				
	FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

在res_conv4_1的residual block之后，网络结构被分成了3个分支，每个分支和原始的ResNet的后续结构类似。

在global中第五层比第四层的缩小一倍，因为做了下采样（12×4×2048），而下面两个分支（part-2，part-3 24×8×2048）没有下采样，因此其第四五层特征图是相同的。

黄色部分：part-2 、part-3 的操作和 global不一样。

以part-2为例：part-2 有两个 pooling，第一个pooling对应Zg_p2（蓝色的长条），通过24×8大小的卷积核，将第五层特征图直接最大池化生成1×1×2048的向量。第二个pooling的卷积核大小和第一个不一样，为12×8，生成2×1×2048的向量，我们将其拆成2个1×1×2048的向量，对应图中part-2 中的2个接在一起的长条形。

淡蓝色使用1×1Conv降维，大小为1×1×256。
测试的时会在淡蓝色的地方，小方块从上到下应该是 8 个，我们把这 8 个 256 维的特征串连一个 2048 的特征，用这个特征替代前面输入的图片。

注意：用于降维的1×1Conv和FC不共享权重

Loss

softmax + batch-hard triplet （P×K）

global分支：图中第一块的Loss 设计，这个地方对2048维的特征做了SoftmaxLoss 损失，对256维的特征做了一个TripletLoss 损失。

part2&part3分支：中间part2部分和下部part3部分有一个全局信息，有global特征，做SoftmaxLoss + TripletLoss 。

下面两个局部特征只用了SoftmaxLoss,未用TripletLoss

原因：一张图片分成从上到下两部分的时候，最完美的情况当然是上面部分是上半身，下面部分是下半身，但是在实际的图片中，有可能整个人都在上半部分，下半部分全是背景，这种情况用上、下部分来区分，假设下半部分都是背景，把这个背景放到 TripletLoss 三元损失里去算这个 Loss，就会使得这个模型学到莫名其妙的特征。

实验

参数设置：

图片size：384×128×3

triplet loss ： P = 16，K = 4

与其他方法进行比较

Methods	Single Query		Multiple Query	
	Rank-1	mAP	Rank-1	mAP
TriNet[14]	84.9	69.1	90.5	76.4
JLML[21]	85.1	65.5	89.7	74.5
AACN[40]	85.9	66.9	89.8	75.1
AOS[16]	86.5	70.4	91.3	78.3
DPFL[7]	88.6	72.6	92.2	80.4
MLFN[4]	90.0	74.3	92.3	82.4
KPM+RSA+HG[30]	90.1	75.3	-	-
PSE+ECN[27]	90.4	80.5	-	-
HA-CNN[22]	91.2	75.7	93.8	82.8
DuATM[31]	91.4	76.6	-	-
GSRW[29]	92.7	82.5	-	-
DNN_CRF[5]	93.5	81.6	-	-
PCB+RPP[36]	93.8	81.6	-	-
MGN(Ours)	**95.7**	**86.9**	**96.9**	**90.7**
TriNet(RK)[14]	86.7	81.1	91.8	87.2
AOS(RK)[16]	88.7	83.3	92.5	88.6
AACN(RK)[40]	88.7	83.0	92.2	87.3
PSE+ECN(RK)[27]	90.3	84.0	-	-
MGN(Ours, RK)	**96.6**	**94.2**	**97.1**	**95.9**

Table 2: Comparison of results on Market-1501 with Single Query setting (SQ) and Multiple Query setting (MQ). "RK" refers to implementing re-ranking operation.

Methods	Rank-1	mAP
SVDNet[35]	76.7	56.8
AOS[16]	79.2	62.1
HA-CNN[22]	80.5	63.8
GSRW[29]	80.7	66.4
DuATM[31]	81.8	64.6
PCB+RPP[36]	83.3	69.2
PSE+ECN[27]	84.5	75.7
DNN_CRF[5]	84.9	69.5
GP-reid[2]	85.2	72.8
MGN(Ours)	**88.7**	**78.4**

Table 3: Comparison of results on DukeMTMC-reID.

Methods	Labeled		Detected	
	Rank-1	mAP	Rank-1	mAP
BOW+XQDA[46]	7.9	7.3	6.4	6.4
LOMO+XQDA[23]	14.8	13.6	12.8	11.5
IDE[47]	22.2	21.0	21.3	19.7
PAN[48]	36.9	35.0	36.3	34.0
SVDNet[35]	40.9	37.8	41.5	37.3
HA-CNN[22]	44.4	41.0	41.7	38.6
MLFN[4]	54.7	49.2	52.8	47.8
PCB+RPP[36]	-	-	63.7	57.5
MGN(Ours)	**68.0**	**67.4**	**66.8**	**66.0**

Table 4: Comparison of results on CUHK03 with evaluation protocols in [50].

消融实验

Model	Rank-1	Rank-5	Rank-10	mAP
ResNet-50	87.5	94.9	96.7	71.4
ResNet-101	90.4	95.7	97.2	78.0
ResNet-50+TP	88.7	96.0	97.2	75.0
Global (Branch)	89.8	95.8	97.5	78.5
Part-2 (Single)	92.6	97.1	98.0	80.2
Part-2 (Branch)	94.4	97.9	98.8	83.9
Part-3 (Single)	93.1	97.6	98.7	82.1
Part-3 (Branch)	94.4	98.2	98.8	84.1
G+P2+P3 (Single)	94.4	97.6	98.5	85.2
MGN w/o Part-3	94.4	97.9	98.7	85.7
MGN w/ Part-4	95.1	98.3	98.9	86.1
MGN (Part2+4)	94.8	98.2	98.9	85.6
MGN (Part3+4)	95.0	98.1	98.8	86.1
MGN w/o TP	95.3	97.9	98.7	86.2
MGN	**95.7**	**98.3**	**99.0**	**86.9**

TP：Triplet loss

Branch：MGN的一个子分支

Single：与MGN中名称对应的分支设置相同的单个网络

1. **与骨干网络对比？**对比 MGN 跟 Resnet50：MGN w/o TP和第一行对比，发现比Resnet50 水平，Rank1 提高了 7.8%，mAP 提高了 14.8%，整体效果是不错的。
2. **多参数？**因为MGN网络有三个分支，参数量肯定会增加，增加的幅度跟 Resnet101的水平差不多，是否网络成果来自于参数增加？我们做了一组实验，第二行有一个 Resnet101，它的 rank1 是 90.4%，mAP 是 78%，这个比 Resnet50 确实好了很多，但是跟我们的工作成果有差距，说明MGN网络不是纯粹堆参数堆出来的结果，应该是有网络设计的合理性在。
3. **多分支多网络？**表格第二个大块有三个分支，把这三个分支做成三个独立的网络，同时独立训练，然后把结果结合在一起，是不是效果跟我们差不多，或者比我们好？实验最后的结果是"G+P2+P3（single）"，Rank1 有 94.4%，mAP85.2%，效果也不错，但差于三个网络联合的网络结构。解释是不同分支在学习的时候，会互相去督促或者互相共享有价值的信息，使得大家即使在独立运作时也会更好。
4. **多分支架构设置？**增加/减少分支数量：去除Part-3分支与添加Part-4分支的模型进行比较，发现去除Part-3分支(不含Part-3的MGN)的性能明显下降，而添加Part-4分支(不含Part-4的MGN)的性能没有明显提升。这证实了我们提出的3个分支设置的必要性和有效性。另一方面，对于Part-N局部分支，分区的数量是影响学习表示粒度的超参数。在每个局部分支中将特征图分为2,3,4，并且仍然发现建议的Part-2/Part-3设置是最优的。

不同分支学到了不同侧重点的东西可以有互补作用，这可能是本模型表现提升的原因。