目录

[ГЛАВА 2 Обучение модели для отследования людей в нескольких видеопотоках （训练模型） 1](#_Toc134742981)

[ReID模型的训练 1](#_Toc134742982)

[数据集准备 1](#_Toc134742983)

[训练原理 2](#_Toc134742984)

[构建模型 3](#_Toc134742985)

[损失函数和优化算法 3](#_Toc134742986)

[训练步骤 4](#_Toc134742987)

[训练过程 5](#_Toc134742988)

[训练结果 5](#_Toc134742989)

[基于FastReID+YOLOv5搭建DeepSORT模型 7](#_Toc134742990)

[构建模型 7](#_Toc134742991)

[测试数据集MOT-16 8](#_Toc134742992)

[实验1：对比测试经过我们改进后的DeepSORT算法与原DeepSORT算法 9](#_Toc134742993)

[实验2：改进算法与几种主流算法对比 11](#_Toc134742994)

# ГЛАВА 2 Обучение модели для отследования людей в нескольких видеопотоках （训练模型）

## ReID模型的训练

### 数据集准备

#### MSMT17数据集

在CVPR2018会议上，提出了一个新的更接近真实场景的大型数据集**MSMT17**，即**Multi-Scene Multi-Time**，涵盖了多场景多时段。

MSMT17数据集采用了安防在校园内的15个摄像头网络，其中包含12个户外摄像头和3个室内摄像头。为了采集原始监控视频，在一个月里选择了具有不同天气条件的4天。每天采集3个小时的视频，涵盖了早上、中午、下午三个时间段。因此，总共的原始视频时长为180小时。

图片包含 游戏机

描述已自动生成

基于Faster RCNN作为行人检测器，三位人工标注员用了两个月时间查看检测到的包围框和标注行人标签。最终，得到4101个行人的126441个包围框。和其它数据集的对比以及统计信息如下图所示。

图形用户界面, 应用程序, 表格

描述已自动生成

MSMT17与其他数据集相比的优势如下：

（1）数目更多的行人、包围框、摄像头数；

（2）复杂的场景和背景；

（3）涵盖多时段，因此有复杂的光照变化；

（4）更好的行人检测器（faster RCNN），bounding box更精准

**评估协议**

按照训练-测试为1：3的比例对数据集进行随机划分，而不是像其他数据集一样均等划分。这样做的目的是鼓励高效率的训练策略，由于在真实应用中标注数据非常昂贵。

最后，训练集包含1041个行人共32621个包围框，而测试集包括3060个行人共93820个包围框。对于测试集，11659个包围框被随机选出来作为query，而其它82161个包围框作为gallery。

测试指标为CMC曲线和mAP. 对于每个query, 可能存在多个正匹配。

### 训练原理

Reid模型的训练原理如下：

1. 数据预处理：reid首先将数据集中的图片进行0.5概率的裁剪、缩放、对称和数据增强等预处理操作，以便提高模型对不同姿态和光照条件下的人员图像的鲁棒性。
2. 特征提取：Reid使用训练好的ResNet系列卷积神经网络作为特征提取器，从输入图像中提取出512维特征向量。同时fastreid还引入一些新的特征提取方法，例如Scale-Aware Alignment (SAA) 和 random-erasing等，以进一步提高特征表达能力。
3. 度量学习：通过度量学习的方式将特征向量映射到一个低维空间中，并采用triplet loss和Cross Entropy Loss损失函数来优化模型，使得同一身份的样本在特征空间中距离更近，不同身份的样本在特征空间中距离更远。
4. 模型优化：使用Adam优化器对模型参数进行优化，以提供模型的训练效率和精度

### 构建模型

为了加速模型的收敛和提高模型性能，我们使用了在ImageNet数据集上预训练的卷积神经网络模型ResNet50。ImageNet是一个广泛使用的图像分类数据集，其中包含超过1000个类别的140万张图像。ResNet50模型使用了深度残差网络结构，可以有效地处理大规模图像分类任务。在训练完成后，改模型的权重可以被用于其他计算机视觉任务中以提高任务的性能和效率，比如我们的行人重识别。

预训练的ResNet50模型输出特征维度是2048。我们在主干网络后面继续接入了一个平均池化层和一个线性分类器。线性分类器的输出维度是N。N 表示训练数据的 ID 数目。

日程表

描述已自动生成

### 损失函数和优化算法

损失函数：

我们的reid模型会产生两个输出： features f 进行预测 logits p。features f 被用于计算 triplet loss，logits p 用于计算交叉损失熵。

**Cross Entropy Loss**

CrossEntropyLoss是一个常用的分类损失函数，通常用于训练深度神经网络进行图像分类任务。具体地，对于一个输入样本x和其真实标签y，CrossEntropyLoss的计算公式如下：

图形用户界面

描述已自动生成

其中C是类别数，yi表示第i个类别的概率（即真实标签），yi^表示模型预测x属于第i个类别的概率。

CrossEntropyLoss的目标是最小化预测与真实标签之间的差距，使得模型能够更准确地预测出每个样本所属的类别。

**Triplet Loss**

Triplet Loss是在人脸识别领域中广泛应用的损失函数之一，其目的是将同一个人的图像嵌入向量（embedding）拉近，不同人的图像嵌入向量拉远。具体来说，Triplet Loss会为每个样本学习一个嵌入向量，该向量能够使得同一个人的图像距离更近，不同人的图像距离更远。

对于一个triplet包括anchor、positive和negative三张图片，Triplet Loss的计算公式如下：

徽标, 公司名称

描述已自动生成

其中，d(a,p)表示anchor和positive之间的欧几里得距离，d(a,n)表示anchor和negative之间的欧几里得距离，m为margin，是一个预先设定的超参数，通常为正数。这个公式的含义是：如果当前的anchor和positive之间的距离减去anchor和negative之间的距离加上margin小于等于0，则说明当前的嵌入向量已经足够好，不需要再进行优化；否则，需要更新模型参数以获取更好的嵌入向量。

Triplet Loss通常与Batch Hard Triplet Mining结合使用，即在每个batch中选择最难的triplet进行训练，以提高模型的性能。

### 训练步骤

1.使用 ResNet50（初始化权重来自于 ImageNet 的预训练模型），然后改变其全链接层为N。N 表示训练数据的 ID 数目。

2.我们随机采样 P 个身份ID，并且对每个ID采集 K 张，最后一个batch size B = P \* K，在这篇论文中，我们设置P=16,K=4

3.我们改变每张图像的大小为 256 × 128，并且使用 0 值填充 10 个像素，然后使用随机剪切的方式，重新剪切 256 × 128 大小的图像。

4. 每张图像以 0.5 的概率值，随机进行水平反转。

5. 每幅图像都被解码为[0,1]中32位浮点原始像素值，然后我们通过减去0.485,0.456,0.406，再除以来对RGB通道进行归一化分别为0.229、0.224、0.225。

6.模型输出的ReID features f 进行预测 logits p。

7.ReID features f 被用于计算 triplet loss，logits p 用于计算交叉损失熵， triplet loss 的 margin m 设置为0.3

8.采用Adam方法对模型进行优化。初始学习率设为0.00035，在第40个epoch和第70个epoch分别降低0.1。总共有120个epoch

【添加伪代码】 或者【流程图】

### 训练过程

电脑屏幕截图

描述已自动生成

图形用户界面, 文本

描述已自动生成

文本

描述已自动生成

### 训练结果

#### 评价指标

在行人重识别研究中，主要采用累计匹配曲线（ Cumulative Matching Characteristics, CMC）和平均精度均值（Mean Average Precision, mAP）这两种评价指标来评估模型的性能。

**累计匹配曲线（Rank-k）**

累计匹配曲线具体体现为第 *k* 位匹配率，具体指的是在候选图像库 *G* 中选取出与待检索图片𝑝𝑟𝑜𝑏𝑒最相似的 *k* 张图像中找到相同行人图片的概率，用公式表达如下：

文本

描述已自动生成

文本

描述已自动生成

**平均精度均值(mAP)**

在早期的行人重识别数据集中，在图像候选库𝐺中有且仅有一张和检索图片 𝑝𝑟𝑜𝑏𝑒相同的行人图片，但是随着 Market-1501 和 DukeMTMC-reID 等大型数据 集的提出，一张待检索图像𝑝𝑟𝑜𝑏𝑒在候选图像库𝐺中通常可以找到多张相同身份 的匹配图片，仅使用 CMC 评价指标难以评估难检索样本对模型性能的影响。因此，为了更加全面地评价行人重识别模型的性能，行人重识别模型的评价指标中加入了平均精度均值。平均精度均值就是一种能够评价全部正样本的排序结果的指标，只有被检索人在候选库中所有的图片都排在最前面时，mAP的指标才会 高，因此它能更全面地反映行人重识别模型的性能。在计算 mAP 时，首先会计算每个待检索图像𝑝𝑟𝑜𝑏𝑒所对应的平均精度（Average Precision, AP），它是用来衡 量模型在单个查询样本上的识别精度的，计算过程如式（3.19）所示：

图片包含 图形用户界面

描述已自动生成

其中，𝑆表示候选库𝐺中对应检索图像𝑝𝑟𝑜𝑏𝑒的正样本数量，{𝑘1, 𝑘2, . . , 𝑘𝑆 }是𝑆个正 样本在排序结果中的索引位置，𝑘𝑟表示前𝑘个结果中正样本的数量。最后，在计 算包含𝑚张图片的待检索库𝑄中所有行人图像的平均精度后，对所有样本的 AP 值作均值即可得到 mAP，计算过程如式（3.20）所示：

图片包含 图形用户界面

描述已自动生成

#### 在数据集MSMT17上的训练结果

我们使用MSMT17中1/3的数据作为样本去训练reid模型，然后用剩下的2/3数据作为验证集检验模型的能力，得到结果如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Dataset | Rank-1 | Rank-5 | Rank-10 | mAP | mINP | Metric |
| MSMT17 | 84.19 | 90.95 | 95.10 | 62.89 | 14.94 | 64.04 |

图片包含 图示

描述已自动生成

分析结果得知：。。。

横向对比其他reid模型在该数据集上的表现：

|  |  |  |
| --- | --- | --- |
| Methods | MSMT17 | |
| Rank-1 | mAP |
| IANet(IVPR’19) | 75.7 | 45.8 |
| Auto-ReID(ICCV’19) | 78.2 | 52.5 |
| OSNet(ICCV’19) | 78.7 | 52.9 |
| ABDNet(ICCV’19) | 82.3 | 60.8 |
| Circle Loss[](CVPR’20) | 76.9 | 52.1 |
| **ours** | **84.19** | **62.89** |

经过横向对比得知。。。

## 基于FastReID+YOLOv5搭建DeepSORT模型

### 构建模型

1. 使用DeepSORT作为跟踪网络，来跟踪每个人物并将其与之前的轨迹相关联
2. 使用YOLOv5s作为DeepSORT的检测部分，检测每帧图像中的人物，输出检测框
3. 使用FastReID作为特征提取器提取行人的外观特征，使得DeepSORT可以区分不同的人物
4. 确定模型超参数。训练模型的参数如下：

DEEPSORT:

REID\_CKPT: "./fast-reid/checkpoint/model-final.pth"

MAX\_DIST: 0.2

MIN\_CONFIDENCE: 0.3

NMS\_MAX\_OVERLAP: 0.5

MAX\_IOU\_DISTANCE: 0.7

MAX\_AGE: 140

N\_INIT: 3

NN\_BUDGET: 100

### 测试数据集MOT-16

采用行人公共数据集 MOT⁃16 测试我们改进后的DeepSORT，模型。该数据集是用于评价多目标跟踪算法性能的常用数据集， 有多种多样行人场景，包括 14 个视频序列，有 7 个视频序列详细标注了行人身份和边框位置，用于训练多目标行人跟踪算法；另外7个视频序列作为测试集。

猫的照片上写着字

中度可信度描述已自动生成

实验采用评价标准为：跟踪准确度（MOTA）、跟踪精度（MOTP）、目标轨迹正确跟踪 80% 以上的个数（MT）、目标轨迹正确跟踪 20%以下的个数（ML）、目标 ID 变换次数（IDSW），跟踪准确度与跟踪精度计算公式如下：

式中：t表示第t帧，表示t帧漏检目标数目；表示t帧误检目标数目；表示t帧身份切换数目；表示t帧出现目标总数目； 表示第t帧目标i预测位置与真实位置之间的距离；表示第t帧中成功匹配的目标数。

该次实验均在 MOT⁃16 的测试集视频序列上进行， 为了更好地凸显本文算法的性能，设计了两组不同实验进行对照比较。

实验 1：对比测试经过我们改进后的DeepSORT算法（YOLOv5+FastReID+DeepSORT）和原DeepSORT算法。将两者放在测试集视频上测试，分析改进的 DeepSort 算法的优劣性。

实验 2：分析在多样场景下改进算法的鲁棒性。将改进算法与几种主流算法对比，分析改进算法的优劣性。

### 实验1：对比测试经过我们改进后的DeepSORT算法与原DeepSORT算法

#### 实验过程

我们将对MOT16数据集中所有拥有行人身份和边框位置人工标注（gt.txt）的视频序列进行测试，一共7个。分别是：MOT16-02, MOT16-04, MOT16-05, MOT16-09, MOT16-10, MOT16-11, MOT16-13。这里以MOT16-13为例，对应其他视频操作相同。

1. 下载MOT16数据集，获取视频序列和各自对应的gt.txt文件。MOT16-13中gt.txt文件如下(部分)，各参数间用逗号分隔：

1,1,1376,485,37,28,0,11,1

2,1,1379,486,37,28,0,11,1

3,1,1382,487,38,29,0,11,1

4,1,1386,488,38,29,0,11,1

5,1,1389,490,38,29,0,11,1

6,1,1393,491,38,30,0,11,1

7,1,1396,492,39,30,0,11,1

8,1,1399,494,39,30,0,11,1

9,1,1403,495,39,30,0,11,1

10,1,1406,496,40,31,0,11,1

gt.txt文件是CSV文本文件，每行包含一个对象，描述其中一帧中的一个跟踪对象，有9个值，用逗号分隔。TrackEval只用到前6个，帧序号，目标ID，跟踪框4个坐标，后3个（目标置信度，目标类别，可见性）不参与运算，可忽略，如下：

<frame>, <id>, <bb\_left>, <bb\_top>, <bb\_width>, <bb\_height>, <conf>, <class>, <visibility>

1. 运行我们构建的DeepSORT模型中的track.py程序

城市街道与高楼大厦的路上有一辆车

描述已自动生成

图片包含 图形用户界面

描述已自动生成

图形用户界面

描述已自动生成

得到兼容MOT16格式的跟踪数据文件MOT16-13.txt。文件MOT16-13.txt：

3 7 726 247 12 40 -1 -1 -1 0

3 8 269 276 16 45 -1 -1 -1 0

3 9 744 256 13 37 -1 -1 -1 0

4 1 1 341 13 93 -1 -1 -1 0

4 2 773 279 22 57 -1 -1 -1 0

4 3 815 276 30 60 -1 -1 -1 0

4 4 690 253 17 55 -1 -1 -1 0

4 5 249 274 19 46 -1 -1 -1 0

4 6 46 360 23 49 -1 -1 -1 0

MOT16-13.txt格式与gt.txt略有差别，号称与MOT16格式“兼容”。“兼容”格式共10个值，其参数间以空格分隔。参与MOT16运算的前6个参数与gt.txt相同，后4个不参与指标运算，均为-1。格式如下：

<frame>, <id>, <bb\_left>, <bb\_top>, <bb\_width>, <bb\_height>, <conf>, <x>, <y>, <z>

1. 运行评估程序：python run\_mot\_challenge.py --BENCHMARK MOT16 --METRICS CLEAR HOTA，得到结果：

电脑萤幕画面

描述已自动生成

1. 分别对MOT16-02, MOT16-04, MOT16-05, MOT16-09, MOT16-10, MOT16-11进行同样的操作，得到这个MOT16数据集上最终的结果。

#### 实验结果

对比原DeepSORT算法的表现，见下表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Algorithm | MOTA ↑ | MOTP ↑ | MT / % ↑ | ML / % ↓ | IDs / % ↓ |
| Original DeepSORT | 61.4 | 79.1 | 32.8 | 18.2 | 781 |
| **Ours** | **66.2** | **80.8** | **35.3** | **17.6** | **760** |

可以看出，改进了DeepSORT算法后，各项指标都略有提升。跟踪准确度（MOTA）提升了5.2%，跟踪精度（MOTP）提升了1.7%,目标轨迹正确跟踪80%以上的个数（MT）提升了3.5%，目标轨迹正确跟踪20%以下的个数（ML）降低了0.6%，目标身份切换总数（IDs）减少了21次，可见改进DeepSORT能减少行人切换次数。

### 实验2：改进算法与几种主流算法对比

实验2结果见表。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Algorithm | MOTA ↑ | MOTP ↑ | MT / % ↑ | ML / % ↓ | IDs / % ↓ | FPS / Hz ↑ |
| SORT | 59.8 | 79.6 | 25.4 | 22.7 | 1423 | 8.6 |
| Original DeepSORT | 61.4 | 79.1 | 32.8 | 18.2 | 781 | 6.4 |
| JDE | 64.4 | - | 35.4 | 20 | 1544 | 18.5 |
| **Ours** | **66.2** | **80.8** | **35.3** | **17.6** | **760** | **5.8** |

我们在 DeepSORT 的基础上改良了行人重识别的算法（fastREID），因此模型的准确度、精度和维持行人 ID 的能力都有提升。我们的模型与其他模型相比具有全面的优势，跟踪准确度（MOTA）、跟踪精度（MOTP）都是最高，目标轨迹正确跟踪80%以上的个数（MT）与JDE几乎持平并列第一，目标轨迹正确跟踪20%以下的个数（ML）和目标身份切换总数（IDs）都是最低的。

此外注意到与 JDE 算法相比实时性略差，这是因为 JDE 属于一阶段的跟踪算法，实时性跟踪相对较高，但是 ID 切换相对频繁，这是由于目标相互遮挡导致的。在零售的环境下经常出现大量密集行人的场景，所以此时我们的算法是一个合适的选择。