Towards Real-Time Multi-Object Tracking

Abstract

现在的 MOT 系统通常使用 tracking-by-detection 的方式。这种方式有：1）用于目标定位的检测模型，2）用于数据关联(data association)的外观嵌入模型(appearance embedding model)。两个模型分开执行可能会导致效率问题，其运行时间只是简单的将两个结构的运行时间加起来，而没有去探究它们之间可能某些结构可以共享的潜在可能性。现有的实时 MOT 的方法主要研究 association step，所以这些方法事实上只是实时 association model 而不是实时 MOT 系统。在本文中，我们提出了一个将目标检测和外观嵌入共享结构学习的 MOT 模型。具体而言，我们将外观嵌入模型合并到 single-shot detector 中，以便该模型可以同时输出检测结果和相应的嵌入。因此，该系统被转化为一个多任务的学习问题：存在多个目标，即，anchor classification，bbox regression，embedding learning,以及各个损失的自动权重。据我们所知，我们这个工作是第一个(近)实时 MOT 系统，它的运行速度取决于输入图像分辨率，大致在 18.8 到 24.1 帧间波动。同时，其跟踪精度可与采用独立检测与嵌入学习的 SOTA 模型相媲美(64.4%MOTA vs 66.1%MOTA 在 MOT-16 challenge)。代码和模型开源在 https://github.com/Zhongdao/Towards-Realtime-MOT。

Introduction

MOT 旨在预测视频序列中多个目标的轨迹，对无论是自动驾驶还是智能视频分析的应用都意义重大。

该问题的主流方法，即 tracking-by-detection ，将 MOT 问题分解成两步：1）detection step，该步骤对视频单帧中的目标进行定位。2）association step，将检测出的目标指派、链接给已经存在的轨迹。这意味着系统至少需要两个计算密集型组件：detector model、embedding(re-ID) model。为了方便起见，我们将这种方法称为 Separate Detection and Embedding (SDE)方法。该方法的总体推测时间大致为两个部件之和，且随着检测数量的增加而增加。这一特性会在 SDE 方法构建一个实时的 MOT 系统时带来严重挑战，而实时是实践中的基本要求。

为了节省计算量，一个可行的想法是将检测器和嵌入模型集成到单个网络中。因此，这两个任务可以共享同一组低级功能，并且避免了重新计算。联合检测器和嵌入学习的一种选择是采用 Faster R-CNN 框架（Ren 等人，2015），这是一种 two-stage 检测器。具体来说，第一阶段，区域提议网络（RPN）与 Faster R-CNN 保持相同，并输出检测到的边界框。第二阶段，Fast R-CNN（Girshick，2015 年）可以通过用度量学习监督代替分类监督来转化为嵌入学习模型（Xiao 等，2017； Voigtlaender 等，2019）。尽管节省了一些计算，但由于采用了两阶段设计，该方法的速度仍然受到限制，并且通常以低于 10 帧每秒（FPS）的速度运行，这远远超出了实时要求。此外，Two-stage 方法的运行时间也像 SDE 方法一样随着目标数量的增加而增加。

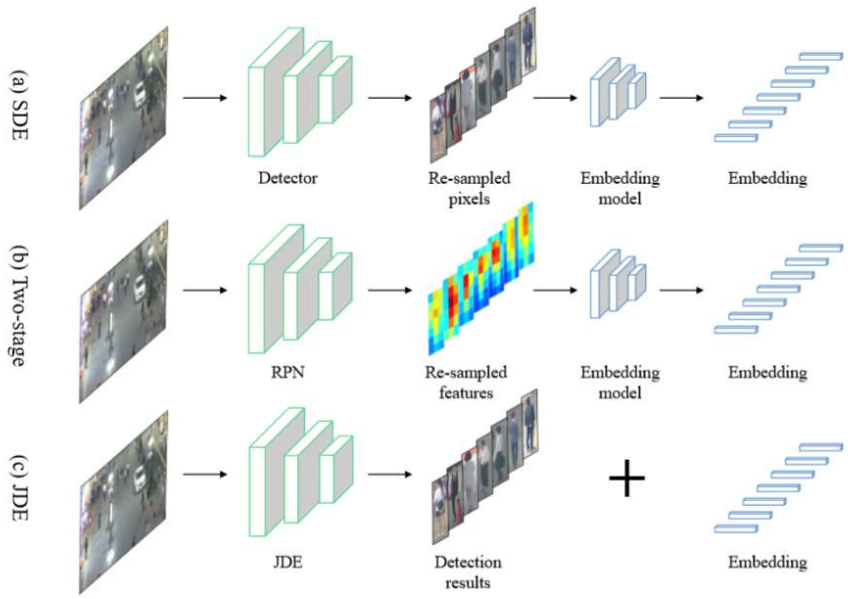

Figure 1: Comparison between (a) the Separate Detection and Embedding (SDE) model, (b) the two-stage model and (c) the proposed Joint Detection and Embedding (JDE).

本论文专门提高 MOT 系统的效率。我们引入了一个早期的尝试，即通过单个深度网络联合学习 Detector 和 Embedding 模型（JDE）。换句话说，提出的 JDE 使用单个网络来同时输出检测结果和检测盒的相应外观嵌入。相比之下，SDE 方法和 two-stage 方法分别以重新采样的像素（边界框）和特征图为特征。边界框和特征图都被馈入单独的 re-ID 模型中以提取外观特征。图 1 简要说明了 SDE 方法、两阶段方法和建议的 JDE 之间的区别。我们的方法几乎是实时的，且准确率几乎与 SDE 方法一样。例如，我们在 MOT-16 test 数据集上获得 MOTA=64.4%的精度下，帧率达到 18.8 帧，作为比较，Faster R-CNN+QAN embedding 在获得 MOTA=66.1%的精度下，帧率只有不到 6 帧。

为了构建一个高效，准确的联合学习框架，我们探索并有意设计以下几个基础方面：训练数据，网络结构，学习目标，优化策略和验证指标。首先，我们收集了六个关于行人检测和人员搜索的公开可用数据集，以形成统一的大规模多标签数据集。在这个统一的数据集中，所有行人边界框都被标记，并且一部分行人身份被标记。其次，选择特征金字塔网络（FPN）作为基础框架，并讨论哪种损失函数是网络学习最佳的嵌入方法。然后，我们将训练过程建模为具有 anchor 分类，框回归和嵌入学习的多任务学习问题。为了平衡每个任务的重要性，我们采用了与任务相关的不确定性（Kendall，Gal 和 Cipolla 2018）来动态加权异构损失。最后，我们采用以下评估指标。平均精度（AP）用于评估检测器的性能。采用一定误报率（FAR）下的正样本通过率（TAR）来评估嵌入的质量。总体 MOT 准确性由 CLEAR 指标（Bernardin 和 Stiefel-hagen 2008）评估，尤其是 MOTA 指标。本文还为联合学习检测和嵌入任务提供了一系列新的设置和基准，我们认为这将促进对实时 MOT 的研究。

我们的工作贡献总结如下：

1） 我们介绍了 JDE，这是用于**联合检测和嵌入学习的单一框架**。作为在线 MOT 系统，它可以（近）实时运行，并且在准确度上与单独的检测+嵌入（SDE）SOTA 结果相当。
2） 我们从训练数据、网络结构、学习目标和优化策略等多个方面对如何构建这样一个联合学习框架进行了深入的分析和实验。

3） 用相同的训练数据进行的实验表明, 提出的 JDE 性能和一系列优秀的 SDE 模型组合一样好，并且达到了最快的速度。
4） 在 MOT-16 上的实验表明，考虑到训练数据量、准确性和速度，我们的方法优于最先进的 MOT 系统。

Related Work

多目标跟踪的最新进展可以大致分为以下几个方面：

1） 将关联问题建模为图优化问题的模型 (Wen 等, 2014； Zamir, Dehghan 和 Shah 2012; Kim 等，2015)。
2） 通过端到端神经网络对关联过程进行建模（Sun 等人 2019; Zhu 等人 2018)。
3） 寻求除 tracking by detection 之外的新颖跟踪范式的人（Bergmann，Meinhardt 和 LealTaixe 2019)。

其中，前两类是过去十年来 MOT 的主流解决方案。 在这些按检测跟踪的方法中，将检测结果和外观嵌入作为输入，唯一要解决的问题是数据关联。 尽管某些方法声称能够达到实时速度，但排除了检测器的运行时间和外观特征提取，因此整个系统仍与要求保持一定距离。 相反，在这项工作中，我们只考虑整个 MOT 系统的运行时，而不是仅考虑关联步骤。在整个系统上实现效率实际上更为重要。

第三类尝试探索新颖的 MOT 范例，例如，通过预测空间偏移将单个对象跟踪器合并到检测器中（Bergmann，Meinhardt 和 Leal-Taixe 2019)。 这些方法的简单性吸引人，但是除非引入了额外的嵌入模型，否则跟踪精度将无法令人满意。 因此，性能和速度之间的权衡仍然需要改进。

我们的方法还与人员搜索任务有关, 该任务旨在从大量数据库框架中定位和识别查询人员。 该任务的一些解决方案是共同学习人体检测器和嵌入模型（Xiao 等人 2017)。 然而，MOT 和人员搜索系统之间的主要区别在于 MOT 对运行时的要求更加严格，因此不能直接借用人员搜索方法。

另一系列相关作品是用于人体姿势估计（Newell，Huang 和 Deng 2017）和检测（Law and Deng 2018）的关联嵌入。 学习了称为关联嵌入的低维密集矢量图，以对人的关节或框角进行分组。 但是，关联仅适用于单个图像，而在 MOT 中，需要跨不同帧进行关联，因此要求嵌入更具区分性。

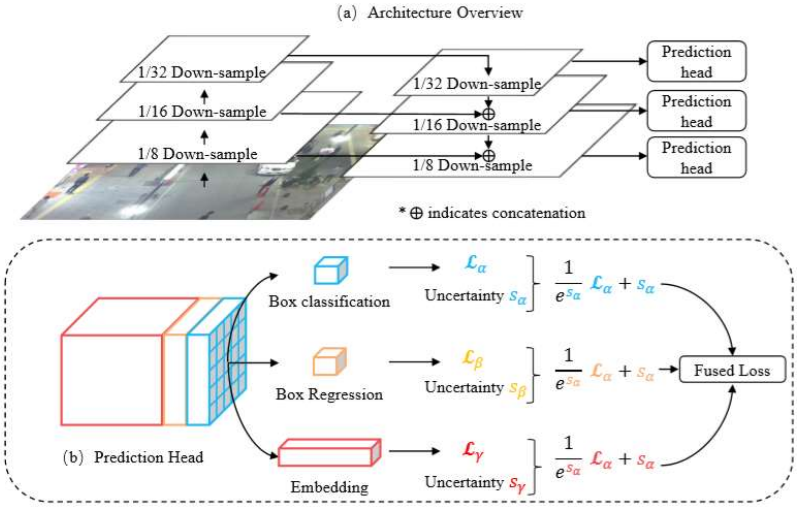

Figure 2: Illustration of (a) the network architecture and (b) the prediction head. Prediction heads are added upon multiple FPN scales. In each prediction head the learning of JDE is modeled as a multi-task learning problem. We automatically weight the heterogeneous losses by learning a set of auxiliary parameters, *i.e.*, the task-dependent uncertainty.

Joint Learning of Detection and Embedding Problem Settings

JDE 的目的是在一次前向传递中同时输出目标的 location 和 appearance embedding。形式上，假设我们有一个 training dataset $\{I, B, y\}_{i=1}^{N}$. 这里的$I \in \mathbb{R}^{c\times h\times w}$表明一个图像帧，$B \in \mathbb{R}^{k\times 4}$表示该帧中 K 个目标的 bbox 的标注信息，$y \in \mathbb{Z}^{k}$表示部分注释的身份标签，其中 -1 表示没有身份标签的目标。JDE 旨在输出 bbox $\hat{B} \in \mathbb{R}^{\hat{k}\times 4}$和 appearance embedding $\hat{F} \in \mathbb{R}^{\hat{k}\times D}$，其中 D 是嵌入维数。同时应满足以下条件：

1、 B^{*}尽可能接近B(原文是B^{*}我认为应该是$\hat{B}$)

2、 给定一个距离度量$d(\cdot)$, $\forall(k_t, k_{t+\Delta t}, k'_{t+\Delta t})$满足$y_{k_{t+\Delta t}} = y_{k_t}$ 和 $y_{k'_{t+\Delta t}} \neq y_{k_t}$，同时我们有$d(f_{k_t}, f_{k_{t+\Delta t}}) < d(f_{k_t}, f_{k'_{t+\Delta t}})$,其中$f_{k_t}$是$\hat{F}_t$的行向量，$f_{k_{t+\Delta t}}$、$f_{k'_{t+\Delta t}}$是$\hat{F}_{t+\Delta t}$的行向量，即目标的嵌入分别在$t$帧和$t+\Delta t$帧。

第一个目的需要模型准确地检测目标。第二个目的是要求外观嵌入具有以下特性。连续帧中相同身份的观察之间的距离应小于不同身份之间的距离。距离度量$d(\cdot)$可以是欧氏距离或余弦距离。 从技术上讲，如果两个目标都得到满足，那么即使是简单的关联策略，例如匈牙利算法，也会产生良好的跟踪结果。

Architecture Overview

我们使用的架构是特征金字塔（FPN）。FPN 使得可以从多尺度预测，从而改善了目标尺度变化很大的行人检测效果。Figure 2 简单的展示了神经网络框架在 JDE 中的使用。首

先，输入视频帧会经过 backbone 进行正向传递，以获得三个比例的特征图，即分别具有 1 / 32、1 / 16 和 1/8 下采样率的比例。然后，将最小尺寸的特征图（语义上的最强特征图）上采样（变得和上一级一样大），与第二小的特征图通过 skip connection 进行融合，对其他比例的特征图也是同样操作。最后，在所有三个尺度上将预测头加到融合特征图上。预测头由几个堆叠的卷积层组成，并输出大小为（6A + D）×H×W 的密集预测图，其中 A 是分配给该尺度的 anchor 模板的数量，D 是嵌入的维数。密集预测图分为三个部分（任务）：

1） the box classification results of size 2A×H ×W;

2） the box regression coefficients of size 4A×H ×W;

3） the dense embedding map of size D×H ×W;

在以下各节中，我们将详细介绍如何训练这些任务。

Learning to Detect

大体上检测分支与标准的 RPN 是相似的，仅有两处修改。第一，我们重新设计了 anchor 的 numbers、scales、aspect ratios 来适应被检测目标（例如我们案例中的行人）。根据普遍的先验，将所有 anchor 的纵横比设置为 1：3。anchor 模板的数量设置为 12，以使每个比例 A = 4，并且 anchor 的比例（宽度）从$11 \approx 8 \times 2^{1/2}$到$512 = 8 \times 2^{12/2}$。其次，我们注意到为前景/背景分配使用的双重阈值选择合适的值很重要。通过可视化，我们确定 IOU> 0.5 基本事实大致确保了前景，这与通用目标检测中的通用设置一致。另一方面，IOU <0.4 groundtruth 的那些 bbox 在我们的案例中应被视为背景，而不是一般情况下的 0.3。 我们的初步实验表明，这些阈值可有效抑制虚假警报(这种警报通常发生在重度遮挡下)。

检测的学习目标具有两个损失函数，即前景/背景分类损失 Lα 和边界框回归损失 Lβ。Lα 表示为交叉熵损失，Lβ 表示为 smooth-L1 损失。回归目标的编码方式与（Ren et al.2015）相同。

Learning Appearance Embeddings

第二个目标是度量学习问题，即学习一个嵌入空间，其中相同身份的实例彼此靠近，而不同身份的实例相距遥远。为了达到这一目标，一种有效的解决方案就是使用 triplet loss (Schroff, Kalenichenko, and Philbin 2015)。 triplet loss 在前人的 MOT 工作中也被使用过 Formally (Voigtlaender et al. 2019)。我们使用 triplet loss $\mathcal{L}_{triplet} = \max(0, f^T f^- - f^T f^+)$, f^T是 mini-batch 中被选为 anchor 的一个实例，f^+代表正样本，f^-代表负样本。为了方便起见，忽略了微小项。Triplet loss 的这种简单公式会面临以下几种挑战。第一，是训练集中的巨大采样空间。在这项工作中，我们通过查看一个 mini-batch 并挖掘该 mini-batch 中的所有负样本和最困难的正样本来解决此问题,如

$$\mathcal{L}_{triplet} = \sum_i \max(0, f^T f_i^- - f^T f^+) \tag{1}$$

其中f^+是一个 mini-batch 中的最困难的正样本。

第二个挑战是，triplet loss 的训练可能会出现不稳定和收敛缓慢的情况。为了让训练过程更加稳定、收敛速度更快，（Sohn 2016）建议在 triplet loss 的平滑上边界上进行优化，

$$\mathcal{L}_{upper} = \log(1 + \sum_i \exp(f^T f_i^- - f^T f^+)) \tag{2}$$

注意，这种平滑上边界的 triplet loss 也可以被写作，

$$\mathcal{L}_{upper} = -\log \frac{\exp(f^T f^+)}{\exp(f^T f^+) + \sum_i \exp(f^T f_i^-)} \tag{3}$$

这种形式和交叉熵损失函数很相似，

$$\mathcal{L}_{CE} = -\log\frac{\exp(f^T g^+)}{\exp(f^T g^+) + \sum_i \exp(f^T g_i^-)} \tag{4}$$

其中，我们将正类别（anchor 实例所属的类别）的类权重表示为g^+，将负类的权重表示为g_i^-。$\mathcal{L}_{upper}$和$\mathcal{L}_{CE}$之间的主要区别是双方面的。首先，交叉熵损失采用可学习的类别权重作为类实例的代理，而不是直接使用实例的嵌入。其次，所有负样本类都参与$\mathcal{L}_{CE}$中的损失计算，从而使 anchor 实例脱离嵌入空间中的所有负样本类。 相反，在$\mathcal{L}_{upper}$中，anchor 实例仅被拉离采样的负样本。

鉴于以上分析，我们推测在我们的情况下这三个损失的表现应该是$\mathcal{L}_{CE} > \mathcal{L}_{upper} > \mathcal{L}_{triplet}$。 实验部分的实验结果证实了这一点。 这样，我们选择交叉熵损失作为嵌入学习的目标（以下简称$\mathcal{L}_\gamma$）。

具体来说，如果将 anchor box 标记为前景，则从密集嵌入图中提取相应的嵌入矢量。提取的嵌入内容被馈送到共享的全连接层中以输出 class-wise logits，然后将交叉熵损失应用于 Logits。 以这种方式，来自多个尺度的嵌入共享相同的空间，并且使得跨尺度的关联变得可行。 在计算嵌入损耗时，将忽略带有标签-1 的嵌入，即带有框注释但没有身份注释的前景。

Automatic Loss Balancing

JDE 中每个预测头的学习目标可以建模为一个多任务学习问题。 联合目标可以表示为每个尺度和每个组成部分的加权线性损失总和，

$$\mathcal{L}_{total} = \sum_i^M \sum_{j=\alpha,\beta,\gamma} \omega_j^i \mathcal{L}_j^i \tag{5}$$

M 是预测头的数量，ω_j^i中的 i=1,⋯,M，$j = \alpha, \beta, \gamma$是 loss weights。有一种简单的方法来确定 loss weights，如下所述：

1） 让$\omega_\alpha^i = \omega_\beta^i$，在已有的目标检测工作(Ren et al. 2015)中是这样建议的。

2） 让$\omega_\alpha^1 = \ldots = \omega_\alpha^M, \omega_\beta^1 = \ldots = \omega_\beta^M, \omega_\gamma^1 = \cdots = \omega_\gamma^M$。

3） 搜索其余两个独立的 loss weights 以获得最佳性能。

使用这种策略搜索 loss weights 可以在几次尝试中产生不错的结果。 但是，搜索空间的减少也给损失权重带来了很大的限制，从而导致产生的损失权重可能远非最佳。 相反，我们采用（Kendall，Gal 和 Cipolla 2018）提出的针对 loss weights 的自动学习方案，采用的是与任务无关的不确定性概念(task-independent uncertainty)。 形式上，具有自动损失平衡的学习目标写为：

$$\mathcal{L}_{total} = \sum_i^M \sum_{j=\alpha,\beta,\gamma} \frac{1}{2}\left(\frac{1}{e^{s_j^i}}\mathcal{L}_j^i + s_j^i\right) \tag{6}$$

其中s_j^i是每个个体损失的任务相关不确定性，并被建模为可学习的参数。 我们向读者推荐（Kendall，Gal 和 Cipolla2018）的工作，里面进行更详细的推导和讨论。

Online Association

尽管关联算法不是本文的重点，但在此我们介绍一种简单快速的在线关联策略，以与 JDE 结合使用。

对于给定的视频，JDE 模型处理每个帧并输出 bboxes 和相应的 appearance embeddings。

因此，我们在观测值的嵌入和先前已有 tracklets 池的嵌入之间计算 affinity 矩阵。 使用匈牙利算法将观测值分配给 tracklets。 卡尔曼滤波器用于平滑轨迹并预测先前轨迹在当前帧中的位置。 如果分配的观测值在空间上与预测位置相距太远，则该分配将被拒绝。然后，tracklest 的嵌入如下更新：

$$f_t = \eta f_{t-1} + (1-\eta)\tilde{f} \quad (7)$$

其中$\tilde{f}$表示分配的观察值的嵌入，而f_t表示在时间戳 t 处的小轨迹的嵌入。 η 是用于平滑的动量项，我们将 η= 0.9。 如果没有任何观察值分配给 Tracklet，则将该 Tracklet 标记为丢失。 如果丢失的时间大于给定的阈值，则标记为已丢失的 tracklets 将从当前的 tracklets 池中删除，或者将在分配步骤中重新找到。

Experiments

Datasets and Evaluation Metrics

在小型数据集上进行实验可能会导致有偏差的结果，并且在将相同算法应用于大规模数据集时可能无法得出同样的结论。 因此，我们通过将六个关于行人检测，MOT 和人员搜索的公开可用数据集放在一起，构建了大规模的培训集。

Dataset	ETH	CP	CT	M16	CS	PRW	Total
# img	2K	3K	27K	53K	11K	6K	54K
# box	17K	21K	46K	112K	55K	18K	270K
# ID	-	-	0.6K	0.5K	7K	0.5K	8.7K

Table 1: Statistics of the joint training set.

这些数据集可分为两种类型：仅包含边界框注释的数据集，以及同时具有边界框和身份注释的数据集。 第一个类别包括 ETH 数据集（Ess et al。2008）和 CityPersons（CP）数据集（Zhang，Benenson 和 Schiele 2017）。 第二类包括 CalTech（CT）数据集（Doll´ar 等人 2009），MOT-16（M16）数据集（Milan 等人 2016），CUHK-SYSU（CS）数据集（Xiao 等人 2017）和 PRW 数据集（Zheng et al.2017）。 收集所有这些数据集的训练子集以形成联合训练集，并排除 ETH 数据集中与 MOT-16 测试集重叠的视频以进行公平评估。Table 1 显示了联合训练集的统计数据。

为了进行验证/评估，需要对性能的三个方面进行评估：检测准确性，嵌入的判别能力以及整个 MOT 系统的跟踪性能。 为了评估检测精度，我们在 Caltech 验证集上以 0.5 的 IOU 阈值计算平均精度（AP）。 为了评估外观嵌入，我们在 Caltech 数据集，CUHK-SYSU 数据集和 PRW 数据集的验证集上提取所有地面真值框的嵌入，在这些实例中应用 1: N 检索，并在错误接受时报告 ture positive rate 0.1（TPR@FAR=0.1）。 为了评估整个 MOT 系统的跟踪精度，我们采用 CLEAR 度量（Bernardin 和 Stiefelhagen 2008），特别是最适合人类感知的 MOTA 度量。 在验证中，我们将 MOT-15 训练集使用重复的序列，并将训练集删除。 在测试过程中，我们使用 MOT-16 测试集与现有方法进行比较。

Implementation Details

我们采用 DarkNet-53（Redmon 和 Farhadi 2018）作为 JDE 中的骨干网络。 该网络使用标准 SGD 训练了 30 个 epoch。学习率初始化为 0.01，在第 15 和第 23 个时期降低 10%。

几种数据增强技术（例如随机旋转，随机缩放和颜色抖动）可用于减少过度拟合。 最后，将增强图像调整为固定分辨率。 如果未指定，则输入分辨率为 1088×608。

Experimental Results

对外观嵌入学习的三种损失函数的比较。 我们首先比较了用交叉熵损失和 triplet loss 以及它的变体训练下，外观嵌入的描述能力，如上一节所述。 对于使用$\mathcal{L}_{triplet}$和$\mathcal{L}_{upper}$训练的模型，对 B / 2 对时间连续帧进行采样以形成大小为 B 的 mini-batch。这确保了始终存在正样本。 对于使用$\mathcal{L}_{CE}$训练的模型，此采样策略不是必需的，并且可以对图像进行随机采样以形成 mini-batch。 表 2 列出了三种损失函数的比较。

Embed. Loss	Weighting Strategy	Det	Emb	MOT	
		AP↑	TPR↑	MOTA↑	IDs↓
$\mathcal{L}_{triplet}$	App.Opt	81.6	42.2	59.5	375
$\mathcal{L}_{upper}$	App.Opt	81.7	44.3	59.8	346
$\mathcal{L}_{CE}$	App.Opt	<u>82.0</u>	88.2	<u>64.3</u>	<u>223</u>
$\mathcal{L}_{CE}$	Uniform	6.8	**94.8**	36.9	366
$\mathcal{L}_{CE}$	MGDA-UB	8.3	<u>93.5</u>	38.3	357
$\mathcal{L}_{CE}$	Loss.Norm	80.6	82.1	57.9	321
$\mathcal{L}_{CE}$	Uncertainty	**83.0**	90.4	**65.8**	**207**

Table 2: Comparing different embedding losses and loss weighting strategies. TPR is short for TPR@FAR=0.1 on the embedding validation set, and IDs means times of ID switches on the tracking validation set. ↓ means the smaller the better; ↑ means the larger the better. In each column, the best result is in **bold**, and the second best is <u>underlined</u>.

不出所料，$\mathcal{L}_{CE}$优于$\mathcal{L}_{triplet}$和$\mathcal{L}_{upper}$。 令人惊讶的是，性能差距很大（+ 46.0 / + 43.9 TAR@FAR=0.1）。 性能差距较大的可能原因是，交叉熵损失要求一个实例与其正类别之间的相似度高于此实例与所有负类别之间的相似度。 该目标比 triplet loss 家族更为严格，后者仅在采样的 mini-batch 中施加约束。 考虑到它的有效性和简单性，我们**将交叉熵损失用于在 JDE 中嵌入学习**。

不同损失加权策略的比较。 损失加权策略对于学习 JDE 的良好联合表示至关重要。 本文采用了三种损失加权策略。 第一种是损失归一化方法（称为“Loss.Norm”），其中，损失以其移动平均幅度的倒数加权。 第二个是（Sener and Koltun 2018）中提出的“ MGDA-UB”算法，最后一个是上一节中介绍的权重不确定性策略(weight-by uncertainty strategy)。此外，我们有两个基准。 首先训练所有具有相同损失权重的任务，称为“Uniform”。 第二种方法称为“ App.Opt”，通过在上一节所述的两个独立变量假设下进行搜索，使用了一组近似的最佳损失权重。表 2 总结了这些策略的比较。有两个发现。

首先，统一基线产生较差的检测结果，因此跟踪精度不好。 这是因为嵌入损失的规模比其他两个损失要大得多，并且支配了训练过程。 一旦我们设置了适当的损失权重以使所

有任务都能以“ App.Opt”基准中的相似速率学习，则检测任务和嵌入任务都会产生良好的性能。

其次，结果表明，“Loss.Norm”策略优于“Uniform”基线，但不如“ App.Opt”基线。MGDA-UB 算法尽管在理论上是最合理的方法，但在我们的案例中失败了，因为它为嵌入损耗分配了太大的权重，因此其性能类似于均匀基准。 唯一优于 App.Opt 基准的方法是不确定性权重策略。在图 3 中，我们使用不确定性方法来可视化损失曲线和学习的权重。 我们观察到，尽管损失权重被统一初始化，但是不确定性方法迅速将嵌入损失的权重降低到 10^-1 左右，并将其他两个任务的损失权重提高到 10^2 的数量级。 可以在 App.Opt 基准（64：0.1）中确定权重，但是由于可以自动学习损失权重，因此“不确定性”策略可以提高跟踪精度。

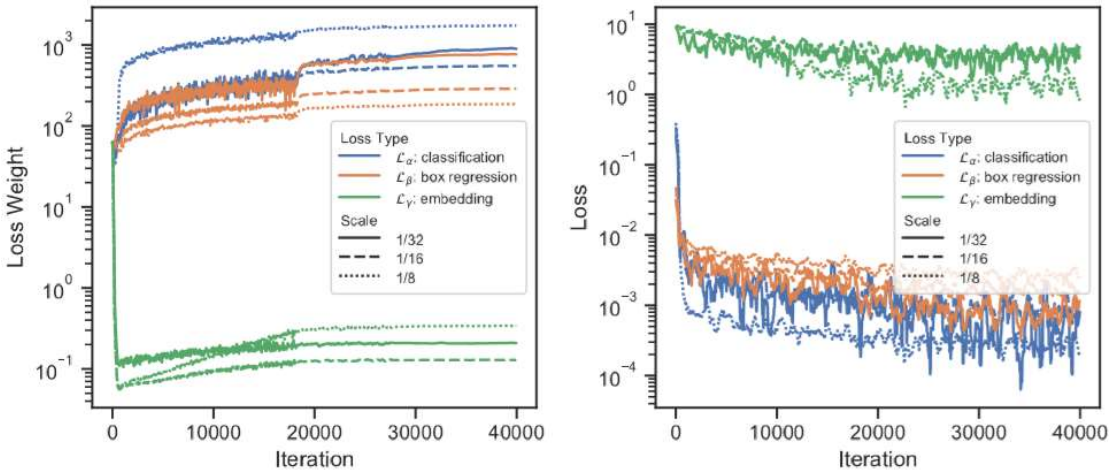

Figure 3: The change of **loss weights** (left) and **losses** (right) in the Uncertainty strategy during training. The profiles of three scales are shown. Note that both the loss weights and the losses are in log-scale. The Uncertainty strategy automatically learns reasonable loss weights and benefits the multi-task learning process.

与 SDE 方法的比较为了证明 JDE 相对于单独检测和嵌入（SDE）方法的优越性，我们实施了几种最先进的检测器和人员身份证模型，并在准确性（MOTA）和运行时间（FPS）将其组合与 JDE 进行了比较。 检测器包括以 ResNet50 和 ResNet-101（He et al.2016）为骨干的 JDE，以 ResNet-50 和 ResNet-101 为骨干的 Faster R-CNN（Ren et al.2015）和 Cascade R-CNN（Cai and Vasconcelos 2018）以 ResNet-50 和 ResNet-101 为骨干。 人员重新识别模型包括 IDE（Zheng，Yang 和 Hauptmann 2016），Triplet（Hermans，Beyer 和 Leibe 2017）和 PCB（Sun 等人 2018）。 在关联步骤中，我们对所有 SDE 模型使用上一节中介绍的相同的在线关联方法。 为了公平比较，这些 SDE 模型使用的训练数据与 JDE 相同。

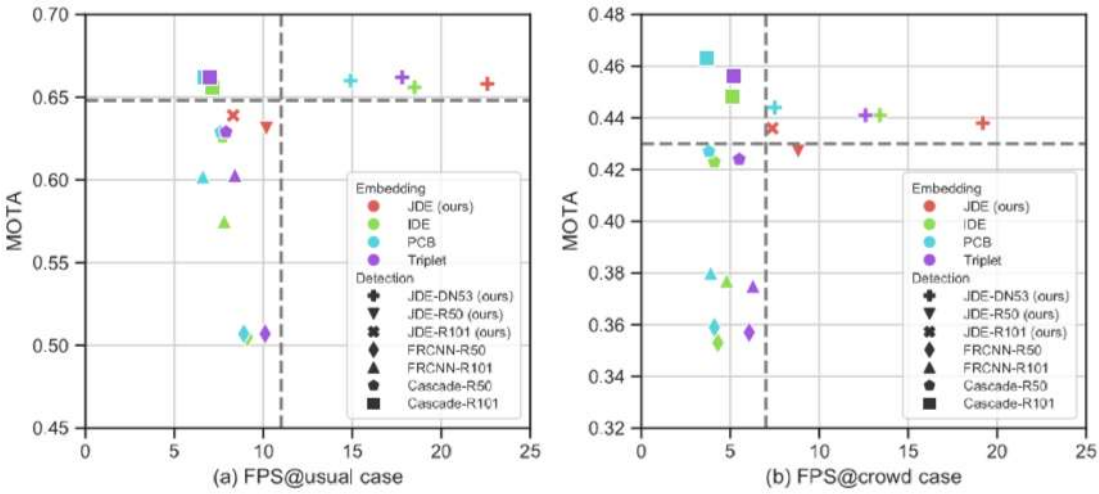

Figure 4: Comparing JDE and various SDE combinations in terms of tracking accuracy (MOTA) and speed (FPS). (a) shows comparisons under the case where the pedestrian density is low (MOT-15 `train` set), and (b) shows comparisons under the crowded scenario (MOT-CVPR19-01). Different colors represent different embedding models, and different shapes denote different detectors. We clearly observe that the proposed JDE method (JDE Embedding + JDE-DN53) has the best time-accuracy trade-off. Best viewed in color.

在图 4 中，我们针对上述检测器和人员身份模型的 SDE 组合的运行时（每幅图像）绘制了 MOTA 度量。 所有模型的运行时都在单个 Nvidia Titan xp GPU 上进行了测试。 图 4（a）显示的是 MOT-15 训练集在行人密度较低的情况下的对比，而图 4（b）显示的是包含高密度人群的视频序列的比较（CVPR19 中的 CVPR19-01 MOTchallenge dataset）。 可以得出几个观察结果。

首先，提出的 JDE 运行速度非常快，同时产生了具有竞争力的跟踪精度，从而在精度和速度之间达到了最佳平衡。 具体来说，带有 DarkNet-53 的 JDE（JDE-DN53）的运行速度为 22 FPS，产生的跟踪精度几乎与 Cascade RCNN 检测器与 ResNet-101（Cascade-R101）+ PCB 嵌入的组合一样好，而后者速度仅以约 6 FPS。

其次，JDE 的跟踪精度非常接近 JDE + IDE，JDE + Triplet 和 JDE + PCB 的组合（参见图 4 中的交叉标记），这表明共同学习的嵌入几乎与单独学习的嵌入一样具有区别性。

最后，通过比较图 4（a）和（b）中相同模型的运行时间，可以观察到，在拥挤的情况下，所有 SDE 模型的速度都会显着下降。 这是因为嵌入模型的运行时间随检测到的目标数量的增加而增加。 在 JDE 中不存在此缺点，因为计算嵌入是为了获得检测结果。 因此，通常情况下和拥挤情况下的 JDE 运行时差异要小得多（请参见红色标记）。**实际上，JDE 的速度下降是由于关联步骤中时间的增加，与目标数正相关。**

与 SOTA 的 MOT 系统进行比较。由于我们使用其他数据而不是 MOT-16 训练集来训练 JDE，因此我们在 MOT-16 基准测试的“私有数据”协议下比较 JDE。比较了私有协议下的最新在线 MOT 方法，包括 DeepSORT 2（Wojke，Bewley 和 Paulus 2017），RAR16wVGG（Fang

等人 2018），TAP（Zhou 等人 2018），CNNMTT（ Mahmoudi，Ahadi 和 Rahmati（2019）和 POI（Yu 等人，2016）。所有这些方法都使用相同的检测器，即以 VGG-16 为骨干的 Faster-RCNN，在大型私人行人检测数据集上进行训练。这些方法之间的主要区别在于它们的嵌入模型和关联策略。例如，DeepSORT 2 使用广域残差网络（WRN）（Zagoruyko 和 Komodakis 2016）作为嵌入模型，并使用 MARS（Zheng et al.2016）数据集训练外观嵌入。具有 VGG，TAP，CNNMTT 和 POI 的 RAR16 使用 Inception（Szegedy 等人，2015），Mask-RCNN（He 等人，2017），5 层 CNN 和 QAN（Liu，Yan 和 Ouyang，2017）作为嵌入模型，分别。这些嵌入模型的训练数据也互不相同。为了便于比较，我们在表 3 中列出了所有这些方法的检测器，嵌入模型和训练数据的数量，并给出了精度和速度指标。

Method	Det	Emb	#box	#id	MOTA	IDF1	MT	ML	IDs	FPSD	FPSA	FPS
DeepSORT_2	FRCNN	WRN	429K	1.2k	61.4	62.2	32.8	18.2	781	<15*	17.4	<8.1
RAR16wVGG	FRCNN	Inception	429K	-	63.0	63.8	39.9	22.1	**482**	<15*	1.6	<1.5
TAP	FRCNN	MRCNN	429K	-	64.8	**73.5**	**40.6**	22.0	794	<15*	18.2	<8.2
CNNMTT	FRCNN	5-Layer	429K	0.2K	65.2	62.2	32.4	21.3	946	<15*	11.2	<6.4
POI	FRCNN	QAN	429K	16K	**66.1**	65.1	34.0	21.3	805	<15*	9.9	<6
JDE-864(ours)	JDE	-	270K	8.7K	62.1	56.9	34.4	**16.7**	1,608	**34.3**	81.0	**24.1**
JDE-1088(ours)	JDE	-	270K	8.7K	64.4	55.8	35.4	20.0	1,544	24.5	**81.5**	18.8

Table 3: Comparison with the state-of-the-art online MOT systems under the private data protocol on the MOT-16 benchmark. The performance is evaluated with the CLEAR metrics, and runtime is evaluated with three metrics: frames per second of the detector (FPSD), frame per second of the association step (FPSA), and frame per second of the overall system (FPS). * indicates estimated timing. We clearly observe our method has the best efficiency and a comparable accuracy.

考虑到总体跟踪精度，例如 MOTA 度量，JDE 通常是可比较的。我们的结果比 DeepSort 2 高 3.0%，比 POI 低 1.7%。就运行速度而言，直接比较这些方法是不可行的，因为它们的运行时间并未全部报告。因此，我们重新实现了基于 VGG-16 的 Faster R-CNN 检测器，并对其运行速度进行了基准测试，然后针对这些方法估算了整个 MOT 系统的运行速度上限。请注意，对于某些方法，没有考虑嵌入模型的运行时，因此速度上限远远不够严格。即使在这样宽松的上限下，所提出的 JDE 仍比现有方法快至少 2 到 3 倍，以接近 1088×608 的图像分辨率达到接近实时的速度，即 18.8 FPS。当我们将输入帧降采样为 864×408 的较低分辨率时，JDE 的运行时间可以进一步提高到 24.1 FPS，而性能只有很小的下降（Δ = -2.6% MOTA）。

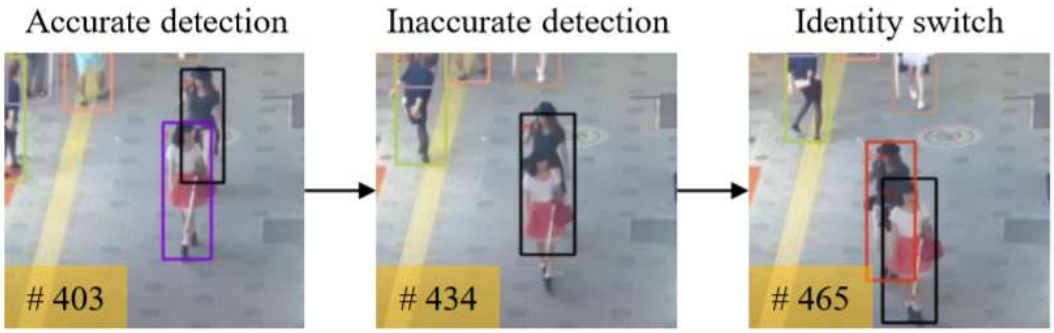

Figure 5: Failure case analysis. Inaccurate detection results when pedestrian have large overlappings give rise to ID switches. Best viewed in color.

分析和讨论。 可能会注意到，与现有方法相比，JDE 的 IDF1 得分更低，ID 切换更多。首先，我们怀疑原因是共同学习的嵌入可能比单独学习的嵌入要弱。 但是，当我们用单独

学习的嵌入替换共同学习的嵌入时，IDF1 分数和 ID 切换的数量几乎保持不变。 最后，我们发现，主要原因是当多名行人彼此重叠时，检测不准确。 图 5 显示了这种 JDE 失败案例。这样不准确的 boxes 会引入很多 ID 切换，不幸的是，这样的 ID 切换经常出现在轨迹的中间，因此 IDF1 得分较低。 在我们未来的工作中，当行人重叠很明显时，**如何改进 JDE 以做出更准确的 boxes 预测仍有待解决。**

Conclusion

在本文中，我们介绍了 MDE 系统 JDE，该系统允许在共享模型中学习目标检测和外观特征。 我们的设计显着减少了 MOT 系统的运行时间，从而可以（接近）实时速度运行。同时，我们系统的跟踪精度可与最新的在线 MOT 方法相媲美。 此外，我们在建立这样的联合学习框架方面提供了有关良好实践和见解的详尽分析，讨论和实验。将来，我们将更深入地研究时间准确性的权衡问题。