CVPR2020 《Learning a Neural Solver for Multiple Object Tracking》

Abstract

图论提供了一种自然的方法，可以在“tracking-by-detection”范式内定义多目标跟踪（MOT，multiple object tracking）。 但是，这也给其学习方法带来了重大挑战，因为定义在这种结构化域运行的模型并非易事。 因此，大多数基于学习的工作都致力于学习更好的 MOT 特征，然后将其与完善的优化框架一起使用。 这项工作采用 MOT 的经典网络流（network flow ）公式来定义基于消息传递网络（MPN，Message Passing Networks）的完全可微分框架。 通过直接在图域上操作，该方法可以在整个检测范围内进行全局推理并预测最终解决方案。 因此，MOT 的学习并不仅限于特征提取阶段，还可以应用于数据关联步骤。 在三个公开基准数据上，其实验都显示出 MOTA（Multiple Object Tracking Accuracy）和 ID F1 分数的显着改善。

1.Introduction

多对象跟踪（MOT）是确定视频中所有对象实例的轨迹的任务。 这是计算机视觉中的一个基本问题，在自动驾驶，机器人技术，生物学和监视等领域中都有应用。 尽管具有相关性，但在深度学习的背景下，它仍然是一项艰巨的任务，并且是一个尚未开发的领域。

近年来，检测跟踪已成为 MOT 中最先进方法中的主要范例。 这两个步骤的方法包括首先获取逐帧对象检测，然后将它们链接形成轨迹。 虽然第一个任务可以使用基于学习的检测器来解决[52，1]，但后者，即数据关联，通常被表述为图分区问题[64、70、72、42、8]。 在此 MOT 的图形视图中，节点表示对象检测，而边缘表示两个节点之间的连接。 活动边缘表示这两次检测属于同一轨迹。 解决图划分任务，即找到活动边或轨迹的集合，也可以分解为两个阶段。 首先，将成本分配给图中的每个边缘，以编码属于同一轨迹的两次检测的可能性。 之后，将这些成本用于图优化框架中以获得最佳图分区。

以前基于图论的 MOT 的工作大致可分为两类：那些专注于图公式化的研究，以及那些专注于学习更好成本的研究。 第一类方法的许多研究致力于建立复杂的图优化框架，该框架结合了多种信息源，目的是对检测之间的高阶相关性进行编码。 这样的方法通常使用在一定程度上手工制造的成本。第二类方法的几项工作采用了更简单、更容易的优化图结构，并着重于通过深度学习技术改进边缘成本的定义。 通过 siamese 卷积神经网络（CNN），这些方法可以编码目标之间可靠的成对交互，但无法说明场景中的高阶信息。 总之，这两类工作面临一个难题：MOT 方法应集中在改进图优化框架，还是特征提取？

我们建议将这两个任务组合成一个基于学习的统一求解器，该求解器可以：(i)学习 MOT 的功能，以及（ii）通过对整个图形进行推理来学习提供解决方案。为此，我们利用 MOT 的经典网络流量公式[73]来定义我们的模型。我们的方法不是学习成对的成本，而是在可用的求解器中使用成对的成本，而是学习直接将图的最终划分预测为轨迹。为此，我们使用消息传递网络（MPN）直接在自然 MOT 域（即图域）中进行学习。我们的 MPN 学会将深度特征组合到整个图形中的高阶信息中。因此，尽管依赖于简单的图形公式，我们的方法仍能够解决检测之间的全局交互作用。我们表明，相对于现有技术，我们的框架可以带来实质性的改进，而无需进行大量的工程设计，并且比某些传统的图形分区方法要快一个数量级。

总而言之，我们做出以下贡献：

•我们提出了一种基于消息传递网络的 MOT 求解器，它可以利用问题的自然图结构来执行特征学习和最终解决方案预测。

•我们提出了一种新颖的具有时间意识的神经信息传递更新步骤，该步骤受 MOT 经典图形公式的启发。

•我们在三个公共基准中显示了我们方法的最先进结果。我们的代码将在论文被接受后发布。

2.Relatedwork

大多数最新的 MOT 工作都遵循“按检测跟踪”范式，该方法将问题分为两个步骤：（i）在每个帧中独立检测行人位置，目前，神经网络是当前最先进的[53、1、69]和（ii）跨时间链接相应的检测以形成轨迹。

Tracking as a graph problem. 数据关联可以针对在线应用[10、20、50]或逐轨[7]在逐帧的基础上完成。对于可以完成的视频分析任务，首选批处理方法，因为它们对遮挡功能更强健。对数据关联进行建模的标准方法是使用图形，其中每个检测都是一个节点，边缘表示它们之间的可能链接。然后可以将数据关联表述为最大流量[8]或等效的最小成本问题，其中包括基于距离的固定成本[29，51，72]，包括运动模型[42]或学习成本[40]。两种公式都可以最佳有效地求解。替代公式通常会导致更复杂的优化问题，包括最小集团[71]，通用求解器，例如多切割[64]。最近的趋势是设计越来越复杂的模型，其中包括其他视觉输入，例如多相机序列的重建[43、66]，活动识别[15]，分割[48]，关键点轨迹[14]或联合检测[64]。]。

Learning in tracking. 自从[36]展示了神经网络在图像分类中的潜力以来，神经网络就已经在许多视觉任务中占据了最先进的地位，这已不是什么秘密。这种趋势也已经进入跟踪社区，在该社区中，学习主要用于学习从图像到上述图形算法的最佳成本的映射。 [37]的作者使用暹罗网络直接了解一对检测之间的成本，而[56]中将 CNN 和递归神经网络（RNN）的混合物用于相同目的。越来越多的四联网络[61]或注意力网络[75]导致了改进的结果。在[55]中，作者展示了学习的 reID 功能对于多对象跟踪的重要性。所有上述方法都独立于实际计算最终轨迹的优化方法来学习成本。相反，[33，65，59]将优化求解器整合到学习中。这些方法背后的主要思想是，对于使用它们的求解器，还需要优化成本。 [33，65，21]依靠结构化学习损失，而[59]提出了一个更通用的双层优化框架。鉴于我们的共同目标是将完整的推理模型并入 MOT 学习中，因此这些作品在精神上可以被视为与我们的作品相似。但是，我们为此采用了不同的方法：我们建议直接学习求解器，并将数据关联视为分类任务，而他们的目标是使他们的方法与闭式求解器配合使用以实现良好的性能。而且，所有这些工作都限于学习成对成本[21，59]或其他二次项[65，33]，但不能将高阶信息作为我们的方法。相反，我们建议利用 MOT 的通用图形公式作为在其中进行学习的领域。

Deep Learning on graphs. 图神经网络（GNN）最早是在[58]中引入的，它是可以在图结构域上运行的神经网络的概括。从那时起，一些工作着重于通过开发卷积变体来进一步发展和扩展它们[11、18、35]。最近，大多数方法都包含在一个更通用的框架中，称为神经信息传递[23]，并在[5]中进一步扩展为图形网络。给定一个图形，该图形具有节点的一些初始特征以及可选的边缘，这些模型的主要思想是将节点（和边缘）嵌入表示中，这些表示不仅要考虑节点自身的特征，还要考虑其图中邻居的特征。以及图的整体拓扑。从化学[23]到组合优化[44]，这些方法在许多领域都表现出了卓越的性能。在视觉范围内，它们已成功应用于诸如人类动作识别[24]，视觉问题回答[49]或单个对象跟踪[22]之类的问题。

3.Tracking as a Graph Problem

我们方法的公式是基于 MOT 的经典最小成本流视图[73]。 为了提供一些背景知识并正式介绍我们的方法，我们首先概述网络流量 MOT 公式。 然后，我们解释如何利用该框架将数据关联任务重新定义为学习问题。

3.1.Problem statement

在 tracking-by-detection 中，目标检测结果$O = \{o_1, \dots, o_n\}$，其中 n 是该视频中所有帧的目标数量总和。每一个检测通过$o_i = (a_i, p_i, t_i)$来表示，其中a_i定义为 bbox 区域的原始图像，p_i包含了图像的坐标，t_i是指时间戳。一个轨迹被定义为一个有序目标检测结果的集合 $T_i = \{o_{i_1}, \dots, o_{i_{n_i}}\}$，其中$n_i$是形成轨迹$i$的检测次数。MOT 的目标就是找到轨迹集合$\mathrm{T}_* =$

$\{T_1, \dots, T_m\}$。

这个问题可以用一个无向图$G = (V, E)$来建模，每一个结点$i \in V$表示一个检测$o_i \in O$。构造边缘集合E，使得连接不同帧中的每对检测，即节点，从而可以利用丢失的检测来恢复轨迹。现在，将原始检测集划分为轨迹的任务可以看作是将此图中的节点分组为断开的组件。因此，场景中的每个轨迹$T_i = \{o_{i_1}, \dots, o_{i_{n_i}}\}$可以映射到图中的一组节点$\{i_1, \dots, i_n\}$，反之亦然。

3.2.Network Flow Formulation

为了表示图分区，我们为图中的每个边引入一个二进制变量。在经典的最小成本流公式中，连接节点之间的边的标签定义为 1 的是指属于同一轨迹，并且在轨迹时间上连续，其余边均为 0。

轨迹$T_i = \{o_{i_1}, \dots, o_{i_{n_i}}\}$等效地由边集$\{(i_1, i_2), \dots, (i_{n_i - 1}, i_{n_i})\} \subset E$表示，对应于其在图中的时间顺序路径。 我们将使用这种观察来正式定义边缘标签。 对于不同时间戳的每对节点$(i, j) \in E$，我们将二进制变量$y(i, j)$定义为：

$$y_{(i,j)} := \begin{cases} 1 & \exists T_k \in \mathcal{T}_* \text{ s.t. } (i,j) \in T_k \\ 0 & \text{otherwise.} \end{cases}$$

当$y(i, j) = 1$，那么将(i, j)这条边称为激活态。我们假设T中的轨迹是节点不相交的，即一个节点不能属于一个以上的轨迹。因此，y必须满足一组线性约束。对于每个节点$i \in V$：

$$\sum_{(j,i) \in E \text{ s.t. } t_i > t_j} y_{(j,i)} \leq 1 \qquad (1)$$

$$\sum_{(i,k) \in E \text{ s.t. } t_i < t_k} y_{(i,k)} \leq 1 \qquad (2)$$

这些不等式是流量守恒约束（flow conservation constraints）的简化形式[2]。 在作者的设置中，它们强制每个节点通过激活态的边去链接，最多只能在过去的帧中链接到一个节点，而在接下来的帧中链接到一个节点。

3.3.From Learning Costs to Predicting Solutions

为了使用作者已经描述的框架获得图分区，标准方法是首先将 cost $c_{(i,j)}$与每个二进制变量$y_{(i,j)}$相关联。该 cost 编码了边缘激活态的可能性[38、37、59]。 通过优化可以找到最终分区：

$$\begin{aligned} \min_y \quad & \sum_{(i,j) \in E} c_{(i,j)} y_{(i,j)} \\ \text{Subject to:} \quad & \text{Equation } (1) \\ & \text{Equation } (2) \\ & y_{(i,j)} \in \{0, 1\}, \qquad (i,j) \in E \end{aligned}$$

可以使用多项式时间内的可用求解器来求解上式。

相反，文中建议**直接学习预测图形中哪些边将成为激活态，即，预测二进制变量 y 的最终值**。 为此，作者将任务视为边缘上的分类问题，其中我们的标签是二进制变量。 总体而言，我们利用刚才介绍的经典网络流公式将 MOT 问题视为**完全可学习的任务**。

4. Learning to Track with Message Passing Networks

本文的主要贡献是基于在上一节中描述的图形公式，将多对象跟踪器作为边缘分类器进行训练的微分框架。 给定一组输入检测，本文提出的模型**经过训练以预测图中每个边的二进制流变量 y 的值**。 该方法基于一种**新颖的消息传递网络（MPN）**，它能够捕获 MOT 问题的图结构。 在我们提出的 MPN 框架内，**外观和几何线索会在整个检测集合中传播，从而使**

我们的模型可以对整个图形进行全局推理。

Pipline 的四个主要组成步骤：

1. **Graph construction:** 给定视频中的一组对象检测，我们构造一个图形，其中节点对应于检测，边缘对应于节点之间的连接（第 3.2 节）。
2. **Feature encoding:** 我们从应用于边界框图像的卷积神经网络（CNN）初始化节点外观特征嵌入。 对于每个边缘，即对于不同帧中的每对检测，我们计算一个向量，该向量具有编码其边界框相对大小，位置和时间距离的特征。 然后，我们将其馈送到多层感知器（MLP），该感知器返回几何嵌入（第 4.3 节）。
3. **Neural message passing:** 我们在图形上执行一系列消息传递步骤。 直观地，对于每轮消息传递，**节点与其连接边共享外观信息**，而**边缘与输入节点共享几何信息**。 这将为包含更高阶信息的节点和边生成更新的嵌入，这些信息取决于整个图形的结构（第 4.1 和 4.2 节）。
4. **Training:** 我们使用最终的边缘嵌入将二进制分类为激活/非激活边缘，并使用交叉熵损失训练我们的整个模型（第 4.4 节）。

在测试时，将模型对每条边的预测作为目标流变量的连续近似值（介于 0 和 1 之间）。然后，我们遵循一个简单的方案对它们进行四舍五入，并获得最终轨迹。有关 pipline 的直观概述，请参见图 1。

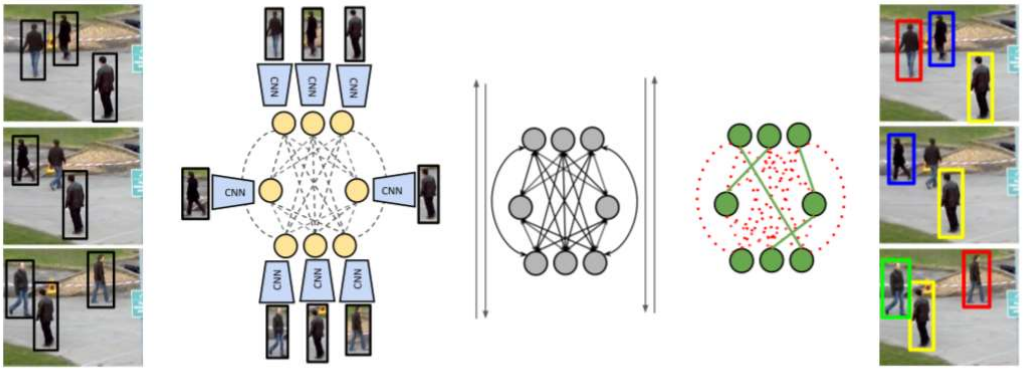


Figure 1: Overview of our method. (a) We receive as input a set of frames and detections. (b) We construct a graph in which nodes represent detections, and all nodes at different frames are connected by an edge. (c) We initialize node embeddings in the graph with a CNN, and edge embeddings with an MLP encoding geometry information (not shown in figure). (c) The information contained in these embeddings is propagated across the graph for a fixed number of iterations through neural message passing. (d) Once this process terminates, the embeddings resulting from neural message passing are used to classify edges into active (colored with green) and non-active (colored with red). During training, we compute the cross-entropy loss of our predictions w.r.t. ground truth labels and backpropagate gradients through our entire pipeline. (e) At inference, we follow a simple rounding scheme to binarize our classification scores and obtain final trajectories.

4.1 Message Passing Networks

在本节中，我们基于[23，34，4，5]中介绍的工作对 MPN 进行简要介绍。$G = (V, E)$是一个图，h_i^0是一个节点嵌入$i \in V$，$h_{(i,j)}^0$是一个边嵌入$(i, j) \in E$，**MPN 的目标是学习一种函数，以在整个图G上传播节点和边缘特征向量中包含的信息**。

传播过程的组织方式是边缘和节点的嵌入的更新，这称为消息传递步骤[23]。在[5、34、4]中，每个**消息传递步骤**又分为**两个更新**：一个从节点到边缘（**v→e**），一个从边缘到节点（**e→v**）。更新是按固定的迭代次数L顺序执行的。对于每个$l \in \{1, \ldots, L\}$，更新的一般形式如下[5]：

$$(v \to e) \quad h_{(i,j)}^{(l)} = \mathcal{N}_e\left([h_i^{(l-1)}, h_j^{(l-1)}, h_{(i,j)}^{(l-1)}]\right) \tag{3}$$

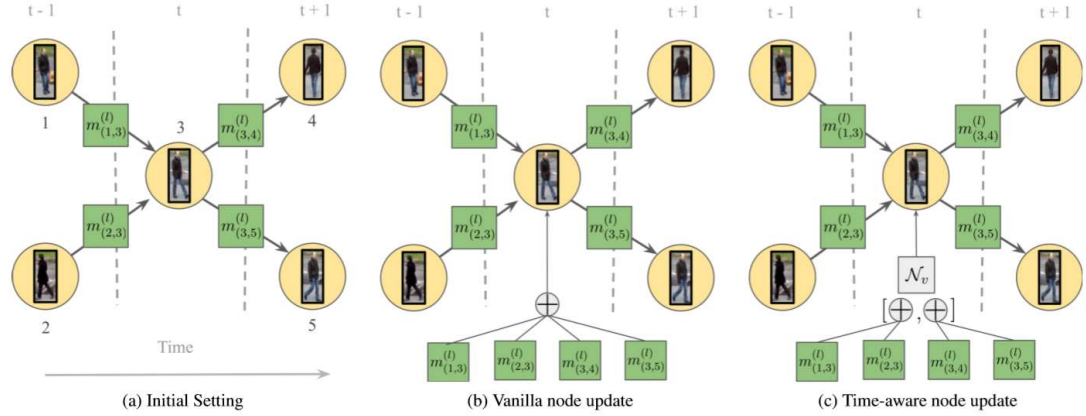
$$(e \to v) \quad m_{(i,j)}^{(l)} = \mathcal{N}_v\left([h_i^{(l-1)}, h_{(i,j)}^{(l)}]\right) \tag{4}$$

$$h_i^{(l)} = \Phi\left(\left\{m_{(i,j)}^{(l)}\right\}_{j \in N_i}\right) \tag{5}$$

N_e和N_v是可学习的函数（例如 MLP），这些函数可以在整个图中共享。[.]表示拼接，$N_i \subset V$是与i邻接的节点，Φ表示阶数不变的运算（例如求和，最大值或平均值）。注意，在L次迭代之后，每个节点都包含图中所有距离L的所有其他节点的信息。因此，L在 CNN 的感受野中起着类似的作用，允许嵌入捕获上下文信息。

4.2 Time-Aware Message Passing

先前的消息传递框架旨在用于任意图形。但是，我们建议利用 MOT 图具有非常特殊的结构。我们的目标是在节点更新步骤中，特别是在我们的网络中对 MOT 特定的感应偏差进行编码。



(a) Initial Setting (b) Vanilla node update (c) Time-aware node update

Figure 2: Visualization of node updates during message passing. Arrow directions in edges show time direction. Note the time division in $t-1$, t, and $t+1$. In this case, we have $N_3^{past} = \{1, 2\}$ and $N_3^{fut} = \{4, 5\}$. 2a shows the starting point after an edge update has been performed (equation 3), and the intermediate node update embeddings (equation 4) have been computed. 2b shows the standard node update in vanilla MPNs, in which all neighbors' embeddings are aggregated jointly. 2c shows our proposed update, in which embeddings from past and future frames are aggregated separately, then concatenated and fed into an MLP to obtain the new node embedding.

回顾公式 4 和 5 中描述的节点更新，该更新允许将每个节点与其邻居进行比较，并汇总来自所有邻居的信息，以使用其他上下文更新其嵌入。 还记得我们的流量守恒约束的结构（等式 1 和 2），这意味着每个节点最多可以连接到将来帧中的一个节点，而另一个可以连接到过去帧中的节点。 可以说，一次聚集所有相邻的嵌入使得更新后的节点嵌入难以捕获是否违反了这些约束（约束满足分析请参见 5.2 节）。

更一般地，将 MOT 图的时间结构显式编码到我们的 MPN 公式中对于我们的学习任务可能是有用的先验。 为了实现这一目标，我们通过将聚合分解为两部分，**将等式 4 和 5 修改为可感知时间的更新规则**：一个是**过去的跨节点**，另一个是**将来的跨节点**。更规范的，让我们分别用N_i^{fut}和N_i^{past}表示i在未来和过去帧中的相邻节点。 让我们也定义两个不同的 MLP，即N_v^{fut}和N_v^{past}。 在每个消息传递步骤l处，对于每个节点$i \in V$，我们开始计算其所有邻居$j \in N_i$的过去和将来的 edge-to-node 嵌入为：

$$m_{(i,j)}^{(l)} = \begin{cases} \mathcal{N}_v^{past}\left([h_i^{(l-1)}, h_{(i,j)}^{(l)}, h_{(i)}^{(0)}]\right) & \text{if} \quad j \in N_i^{past} \\ \mathcal{N}_v^{fut}\left([h_i^{(l-1)}, h_{(i,j)}^{(l)}, h_{(i)}^{(0)}]\right) & \text{if} \quad j \in N_i^{fut} \end{cases} \tag{6}$$

注意，初始嵌入$h_{(i)}^{(0)}$已添加到计算中。 之后，我们根据这些嵌入相对于i的未来位置或过去位置，分别汇总这些嵌入：

$$h_{i,past}^{(l)} = \sum_{j \in N_i^{past}} m_{(i,j)}^{(l)} \tag{7}$$

$$h_{i,fut}^{(l)} = \sum_{j \in N_i^{fut}} m_{(i,j)}^{(l)} \tag{8}$$

现在，这些操作分别产生过去和将来的嵌入$h_{i,past}^{l}$和$h_{i,fut}^{l}$。 我们通过拼接它们并将最终结果馈送到最后一个 MLP（表示为N_v）来计算最终的更新节点嵌入：

$$h_i^{(l)} = \mathcal{N}_v([h_{i,past}^{(l)}, h_{i,fut}^{(l)}]) \tag{9}$$

我们在图 2（c）中总结了我们的时间感知更新。正如我们通过实验证明的那样（参见 5.2），相对于 MPN 的原始节点更新，这种简单的体系结构设计显着提高了性能，如图 2 (b) 所示。

4.3 Feature encoding

MPN 接收作为输入的初始嵌入是由其他可向后传播的网络生成的。

Appearance embedding.我们依赖于 CNN（定义为N_v^{enc}）去学习从 RGB 数据中直接提取特征嵌入。对于每一个检测$o_i \in O$，它的相应图像a_i，我们通过计算$h_i^{(0)} \coloneqq \mathrm{N}_v^{enc}(a_i)$来获得$o_i$的相应节点嵌入。

Geometry embedding. 我们力求获得一种表示形式，该表示形式针对不同帧中的每对检测对其相对位置大小以及时间间隔进行编码。对于时间戳不同的每对检测o_i和o_j，其边界框有左顶点坐标和高、宽表示，(x_i, y_i, h_i, w_i)、(x_j, y_j, h_j, w_j)，我们这样计算它们的相对距离和大小：

$$\left(\frac{2(x_j - x_i)}{h_i + h_j}, \frac{2(y_j - y_i)}{h_i + h_j}, \log\frac{h_i}{h_j}, \log\frac{w_i}{w_j}\right)$$

然后，我们将这些基于坐标的特征向量与时间差$t_j - t_i$，以及相对相对外观$||\mathrm{N}_v^{enc}(a_j) - \mathrm{N}_v^{enc}(a_i)||_2$拼接起来，并将其馈送到神经网络$\mathrm{N}_e^{enc}$，以获得初始边缘嵌入$h_{(i,j)}^{(0)}$。

4.4 Training and inference

Training loss.为了对边缘进行分类，我们使用带有 S 型值单个输出单元的 MLP，我们将其表示为N_e^{class}。对于每一条边$(i,j) \in E$，我们通过 given message passing 步骤l处给定 MPN 的输出嵌入，即$h_{(i,j)}^{(l)}$，计算出预测$\hat{y}_{(i,j)}^{(l)}$，再将其通过N_e^{class}。为了进行训练，对于目标流变量y，我们使用对最后一个消息传递步骤中生成的嵌入的预测的二元交叉熵：

$$\mathcal{L} = \frac{-1}{|E|}\sum_{l=l_0}^{l=L}\sum_{(i,j)\in E} w \cdot y_{(i,j)} \log(\hat{y}_{(i,j)}^{(l)}) + (1 - y_{(i,j)}) \log(1 - \hat{y}_{(i,j)}^{(l)}) \tag{10}$$

其中，$l_0 \in \{1, \ldots, L\}$是计算预测的第一个消息传递步骤，w 表示用于对一值标签加权的正标量，以说明有效边和无效边之间的高度不平衡。

Inference. 在推断过程中，我们将在最后一个消息传递步骤中从模型中获得的输出值解释为 MOT 问题的解决方案，即指标变量 y 的最终值。由于这些预测是 S 形单位的输出，因此它们的值在 0 到 1 之间。获得硬 0 或 1 决策的一种简单方法是通过阈值对输出进行二值化。但是，此过程通常不能保证方程式 1 和 2 中的流量守恒约束得以保留。实际上，由于建议的时间感知更新步骤，当阈值设为 0.5 时，我们的方法平均将满足 98％以上的约束。 之后，一个简单的贪婪舍入方案就足以获得可行的二进制输出。精确的最佳舍入解也可以通过简单的线性程序高效地获得。

5.Experiments

在本节中，我们将首先更好地了解模型的行为。 然后，我们将与三个数据集的已发布方法进行比较，并显示最新的结果。 所有实验均在 MOTChallenge 行人基准上进行。

Datasets and evaluation metrics. 多对象跟踪基准 MOTChallenge 由几个具有挑战性的行人跟踪序列组成，并具有频繁的遮挡和拥挤的场景。 挑战包含三个单独的跟踪基准，即 2D MOT 2015 [41]，MOT16 [47]和 MOT17 [47]。 它们包含具有不同视角，对象大小和数量，相机运动和帧频的序列。 对于所有挑战，我们使用 MOTChallenge 提供的检测结果来确保与其他方法的公平比较。 该基准提供了几种评估指标。 最重要的是多对象跟踪准确性（MOTA）[30]和 ID F1 分数（IDF1）[54]，因为它们量化了多个对象跟踪的两个主要方面，即对象覆盖率和身份保存。

5.1 Implementation details

Network models. 对于用于编码检测外观的网络 Nenc v（请参阅第 4.3 节），我们使用 ImageNet [19]上预训练的 ResNet50 [25]体系结构的前 4 个块，然后使用两个完全连接的层来获得尺寸为 256 的嵌入。

我们在 Market1501 [74]上针对身份验证（ReID）任务训练网络，如[62、32、46]中所述。 经过培训后，将添加两个附加的全连接层，以将 Nenc v 的嵌入大小减小到 32。其余的编码器和分类器网络是 MLP，其确切架构在补充材料中进行了详细说明。

Data Augmentation. 为了训练我们的网络，我们抽样了 8 张图形的批处理，对应于给定序列中的 15 帧，以每秒 5 帧的速度采样。 我们通过从图中随机删除节点以及添加节点（分别模拟丢失的检测和错误警报）来进行数据增强。 相应地重新计算了所得图形的地面真值边缘标签。

我们使用卷积层的学习率 3·10-6 和卷积层的学习率 3·10-4，权重衰减项 10-4 和将 β1 和 β2 分别设置为 0.9 和 0.999 的亚当优化器。 我们训练了 30 个纪元，这已经证明可以在我们的实验中收敛。

Batch Processing. 我们以 15 帧为单位批量处理离线视频，每批之间有 14 个重叠帧，以确保图中两个连接节点之间的最大时间距离在整个图中保持稳定。 根据预训练的 CNN 特征，仅当两个节点都位于前 K 个相互最邻近的邻居（K = 50）中时，我们才通过连接两个节点来限制图的连通性。 每个批次都由我们的网络独立解决，对于批次之间的重叠边缘，我们在舍入步骤之前对来自所有图形解决方案的预测取平均值。 为了填补轨迹上的间隙，我们执行了单双线性插补和遗漏帧。

Baseline. 最近，[9]展示了检测器用于简单数据关联的潜力，为 MOT 建立新的基线，我们也遵循该基线。 请注意，该方法仍使用公共检测，因此，它可以与 MOTChallenge

上的所有方法完全媲美。 [9]的主要缺点之一是无法填补空白，也无法通过遮挡正确地恢复身份。 正如我们将显示的，这正是方法所擅长的地方。

Arch.	MOTA ↑	IDF1 ↑	MT ↑	ML ↓	FP ↓	FN ↓	ID Sw. ↓	Constr. ↑
Vanilla	62.0	63.1	557	386	4401	122584	1111	77.8
T. aware	63.7	69.2	638	378	6676	115078	587	98.8

Table 1: We investigate how our proposed update improves tracking performance with respect to a vanilla MPN. *Vanilla* stands for a basic MPN, *T. aware* denotes our proposed time-aware update. The metric *Constr* refers for the number of flow conservation constraints satisfied on average over entire validation sequences.

5.2.Ablation study

在本节中，我们旨在回答三个主要问题，以了解我们的模型。 首先，我们相对于 4.1 中描述的时间不可知的香草节点更新，**比较了时间感知神经消息传递更新的性能**。 其次，我们**评估网络培训中消息传递步骤的数量对整体跟踪性能的影响**。 第三，我们研究了**不同的信息源（即来自 CNN 的外观嵌入和相对位置信息）如何影响不同的评估指标**。

Experimental Setup. 我们使用 MOT15 和 MOT17 数据集的训练序列进行所有实验。 为了评估我们的模型，我们将 MOT17 序列分为三组，并使用它们通过 3-fold cross-validation 测试我们的模型。 我们将报告在验证期间获得的最佳总体 MOT17 度量（有关详细信息，请参见补充材料）。 为了与显示出较差约束满足条件的基线进行公平比较，我们在所有实验中均通过线性程序使用精确的舍入（请参见第 4.4 节）。

Time-Aware Message Passing. 我们调查了我们提出的时间感知节点更新如何影响性能。 为了公平地比较，我们对基线执行超参数搜索。 尽管如此，我们仍观察到几乎所有指标都取得了显着改善，包括 IDF1 超过 6 分。 正如我们所期望的，我们的模型在**链接检测方面特别强大，因为它利用了邻近的信息和图结构**，使决策更加稳健，因此产生的**身份切换少得多**。 我们还报告了**通过阈值保持模型的输出值直接二值化时满足的约束百分比**。 值得注意的是，我们的具有时间感知节点更新的方法能够自动产生几乎完全可行的结果，而基线的约束满意度要低得多。 这证明了其捕获 MOT 问题结构的能力。

Number of Message Passing Steps. 直观地讲，增加消息传递步骤 L 的数量可以使每个节点和边缘嵌入对进一步的上下文进行编码，并为边缘预测提供迭代优化的能力。因此，人们期望更高的价值产生性能更好的网络。我们在表 3 中通过固定数量的消息传递步骤（从 0 到 15）来训练该假设。我们使用情况 L = 0 作为基线，在该基线中我们在初始边缘嵌入之上训练了一个二进制分类器，因此，不使用任何上下文信息。不出所料，我们看到 IDF-1 和 MOTA 都有明显的上升趋势。此外，我们观察到从 0 到 3 消息传递步骤的深度增长，这表明从图中的成对特征转换为高阶特征时，可以获得最大的改进。我们还注意到，上升趋势停滞在六个消息传递步骤附近，并且在十二个消息传递步骤后没有任何改善。因此，我们在最终配置中使用 L = 12。

Effect of the features. 我们的模型接收两个主要信息流：（i）来自在 RGB 图像上运行的 CNN 的外观信息，以及（i）来自 MLP 编码检测之间相对位置的几何特征。这些分别通过用作初始节点和边缘嵌入而被合并到模型中，随后在神经消息传递过程中得到完善。我们在表 5.2 中显示了一些配置。对于节点，我们探索了使用零值向量 vss 初始化特征向量之间的差异。 CNN 功能。对于 Foredgeembedding，我们尝试从三个特征源进行组合：时间差，相对位置和两个边界框之间 CNN 嵌入中的欧式距离。我们强调一个事实，相对位置似乎是整体性能的关键组成部分，因为在没有其他可用信息的情况下，网络仍可以达到 62.9 的 MOTA 值。但是，节点 CNN 功能强大，可以减少误报和身份切

换的数量。请注意，只有通过在节点和边缘要素上同时嵌入 CNN，我们才能实现更高的准确性和身份保留。

Node Feats.	Edge Feats.	MOTA ↑	IDF1 ↑	MT ↑	ML ↓	FP ↓	FN ↓	ID Sw. ↓
CNN	Time	59.0	51.2	571	381	12623	122730	2609
–	Time+Pos	62.9	67.3	636	372	8435	115534	1004
CNN	Time+Pos	63.2	67.8	643	370	8063	115142	906
–	Time+Pos+CNN	63.2	67.8	648	369	8054	114960	924
CNN	Time+Pos+CNN	63.7	69.2	638	378	6676	115078	587

Table 2: We investigate the influence of incorporating several sources of information in our model. For Nodes, we consider either CNN embeddings (CNN) vs a 0-valued vector (–). For edges we explore combinations of three sources of information: time difference in seconds (Time), relative position features (Pos) and the Euclidean distance between CNN embeddings of the two detections (CNN).

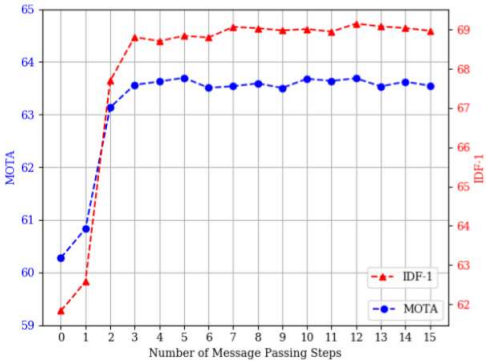


Figure 3: We report the evolution of IDF-1 and MOTA when training networks with an increasing number of message passing steps.

Method	MOTA ↑	IDF1 ↑	MT ↑	ML ↓	FP ↓	FN ↓	ID Sw. ↓	Hz ↑
2D MOT 2015 [41]								
Ours	**48.3**	**56.5**	**32.2**	**24.3**	9640	**21629**	504	11.8
Tracktor [9]	44.1	46.7	18.0	26.2	6477	26577	1318	16.7
KCF [16] (G)	38.9	44.5	16.6	31.5	7321	29501	720	0.3
AP_HWDPL_p [13] (G)	38.5	47.1	8.7	37.4	**4005**	33203	586	6.7
STRN [68]	38.1	46.6	11.5	33.4	5451	31571	1033	13.8
AMIR15 [57]	37.6	46.0	15.8	26.8	7933	29397	1026	1.9
JointMC [31] (G)	35.6	45.1	23.2	39.3	10580	28508	**457**	0.6
DeepFlow[59] (G)	26.8	–	–	–	–	–	–	
MOT16 [47]								
Ours	**55.9**	**59.9**	**26.0**	**35.6**	7086	**72902**	431	11.8
Tracktor [9]	54.4	52.5	19.0	36.9	**3280**	79149	682	16.7
NOTA [12] (G)	49.8	55.3	17.9	37.7	7428	83614	614	–
HCC [46] (G)	49.3	50.7	17.8	39.9	5333	86795	**391**	0.8
LMP [63] (G)	48.8	51.3	18.2	40.1	6654	86245	481	0.5
KCF [16] (G)	48.8	47.2	15.8	38.1	5875	86567	906	0.1
GCRA [45]	48.2	48.6	12.9	41.1	5104	88586	821	2.8
FWT [26] (G)	47.8	44.3	19.1	38.2	8886	85487	852	0.6
MOT17 [47]								
Ours	**55.7**	**59.1**	**27.2**	**34.4**	25013	**223531**	**1433**	11.8
Tracktor[9]	53.5	52.3	19.5	36.6	**12201**	248047	2072	16.7
JBNOT [9] (G)	52.6	50.8	19.7	35.8	31572	232659	3050	5.4
FAMNet [17]	52.0	48.7	19.1	33.4	14138	253616	3072	–
eHAF[60] (G)	51.8	54.7	23.4	37.9	33212	236772	1834	0.7
NOTA [12] (G)	51.3	54.7	17.1	35.4	20,148	252,531	2,285	–
FWT [26] (G)	51.3	47.6	21.4	35.2	24101	247921	2648	0.2
jCC [31] (G)	51.2	54.5	20.9	37.0	25937	247822	1802	1.8

Table 3: Comparison of our method with state-of-the art. We set new state-of-the art results by a significant margin in terms of MOTA and especially IDF1. Our learned solver is more accurate while being magnitudes of order faster when compared to graph partitioning methods, indicated with (G).

5.3.Benchmarke valuation

我们在表 3 中报告了我们的模型在 MOT15，MOT16 和 MOT17 数据集中获得的度量。值得一提的是，将我们的方法与图分区方法进行比较时，性能差异很大（表 3 中的（G）所示）。由于篇幅所限，我们在补充材料中展示了我们的方法与图形方法的详细比较。我们的方法获得了所有挑战的最新结果，尤其是将 IDF1 度量分别提高了 9.4、4.6 和 4.4 个百分点，这证明了其在**身份保存方面的强大性能**。我们**将这种性能提高归因于我们的消息传递体系结构收集高级信息的能力。在链接轨迹时考虑邻居的信息，这使我们的方法能够做出全局性的预测，这不可避免地导致较少的身份切换**。此外，我们还实现了更多的轨迹覆盖，以最大跟踪（MT）轨迹显着增加表示，最多增加了 9 个百分点。值得注意的是，在超越以前的方法的同时，我们比其他 SoA 方法快得多（远远超过一个数量级），尤其是与昂贵的图形划分方法相比，例如[31]。

6.Conclusions

我们已经演示了如何利用 MOT 的网络流量公式将整个跟踪问题视为学习任务。 我们提出了一种完全可区分的流水线，可以同时学习特征提取和数据关联。 算法的核心是具有新的时间感知更新步骤的消息传递网络，该步骤可以捕获问题的图形结构。 在我们的实验中，我们以最先进的状态展示了我们方法的明显性能改进。 我们希望我们的方法能够为以后的工作打开大门，而不仅仅是 MOT 的特征提取，而是着重于将学习集成到整个数据关联任务中。