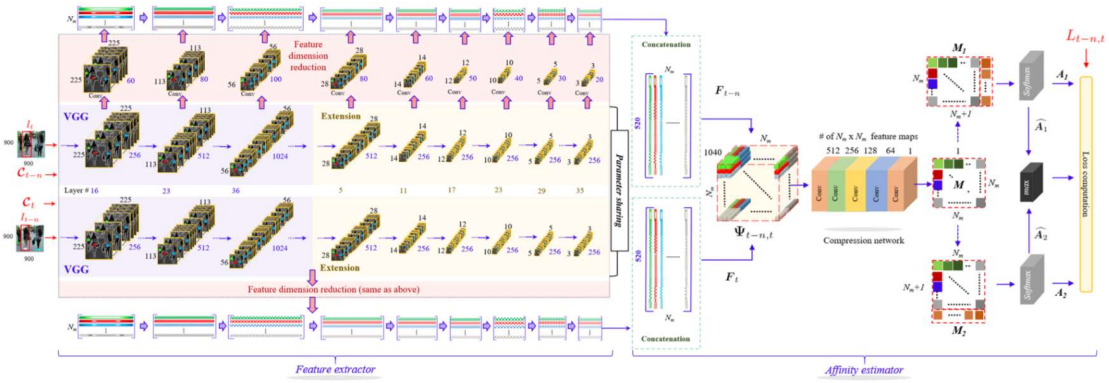
TPAMI 2019 《Deep Affinity Network for Multiple Object Tracking》



这是 18 年十月底发出来的一篇 MOT 论文，要针对 data association 部分，使用了深度学习的方法来克服非深度学习带来的问题。提出了 Deep Affinity Network (DAN)如上图所示。该网络以**端到端**的方式**学习**目标物体的**外观特征**以及在**若干帧内的关联性**，其**外观建模包括**对**物体**及其**周边的分层特征学习**，DAN 通过对连续（不连续也可以，一般为非连续）2 帧内物体之间特征计算其关联性，还包括物体在视频中出现和消失的情况。

算法细节

1、Data preparation

1). 前期的数据准备和预处对图像帧中的每一个像素值进行成比例缩放，缩放因子为[0.7, 1.5]的 random value。之后对将处理过的图片转换成 HSV 格式，并将 Saturation channel 即图片的饱和度也进行缩放，缩放因子为[0.7，1.5]的 random value。再将图片转换成 RGB 格式，最后再以相同的方法进行缩放。

2). 对图片的尺寸进行放大，缩放因子为[1, 1.2]的 random value。扩大的图片中填充的像素值为数据的平均像素值

3). 对图片进行裁剪，裁剪率为[0.8,1]中的 random value。同时保证裁剪的区域包含了所有的检测的物体的中心位置

上述三步中的每一步都是以概率 0.3 顺序应用在帧图像对上。然后帧再 resize 到固定大小，H*W*3，然后以 0.5 的概率进行水平翻转。这些处理的步骤均来自于[W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, “SSD: Single shot multibox detector,” Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)]. 不同的是，对**帧图像对**使用上面的步骤进行同时处理。

DAN 的**输入**为处理之后的**图像帧及检测到的物体的中心位置信息**。作者引入 Nm 用来表示允许一帧中出现的最多的物体的数量，实验中 Nm=80.同时设定 association matrix 为 Nm*Nm，若实际检测的 object 数量比 Nm 少的话，需要使用全 0 的行或列来进行填充。

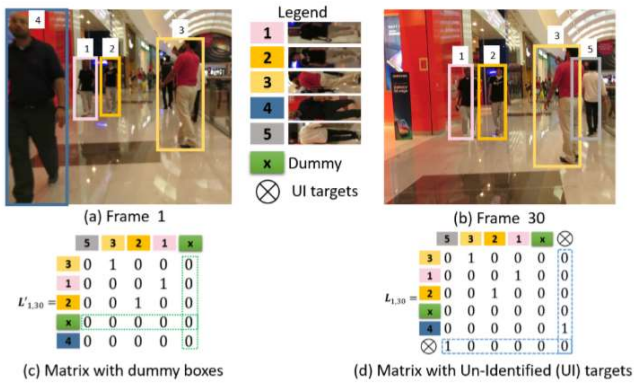


图 c 是进行填充了之后的关联矩阵，其中 Nm=5，图 d 是在图 c 的基础上增加了新出现和离开的物体，列中的 1 表示旧物体离开视频，行中的 1 表示新物体进入视频。

2、Feature extractor

该部分对帧中检测到的**物体的特征进行建模**。特征的抽取是通过以视频中的**帧图像对**和**物体的中心信息作为输入**，加上两道卷积层流分别对应两处输入。该两道流实现时**共享模型参数**，而其模型架构来自于 VGG16 网络。但是会在使用之前对 VGG16 做出修改，将全连接层和 sofatmax 层转换成卷积层，之所以这么做是因为 **objects 的空间信息更容易通过卷积层进行处理表示**。与原始的 VGG16 相比，新网络的输入 frame 尺寸要大很多。结果就是我们在新的 VGG16 网络的最后一层之后仍然能够计算出 56*56 的 feature map。在图 1 中我们将 VGG16 的最后一层索引为第 36 层，因为 BN 和 ReLu 都算作单独的层。

VGG 层之后，通过使用更深层次的卷积层，我们将超过 56*56 的 feature map 的空间维度都要进行缩减。后面的 36 层 Extension 网络渐渐将 feature map 缩减到 3*3 的尺寸，其网络细节在 Table2 给出。

TABLE 2: Architectural details of the *Extension* & *Compression* networks in Fig. 1: I.C denotes the number of input channels for a layer, O.C is the number of output channels, S is the stride size, B.N (Y/N) indicates if the Batch Normalization is applied; and ReLU (Y/N) indicates if the ReLU activation is used. Strides and Paddings are the same in both spatial dimensions.

Sub-network	Index	I.C	O.C	Kernel	S	Padding	B.N	ReLU
Extension (Feature extractor)	0	1024	256	1×1	1	0	Y	Y
	3	256	512	3×3	2	1	Y	Y
	6	512	128	1×1	1	0	Y	Y
	9	128	256	3×3	2	1	Y	Y
	12	256	128	1×1	1	0	Y	Y
	15	128	256	3×3	2	1	Y	Y
	18	256	128	1×1	1	0	Y	Y
	21	128	256	3×3	2	1	Y	Y
	24	256	128	1×1	1	0	Y	Y
	27	128	256	3×3	2	1	Y	Y
	30	256	128	1×1	1	0	Y	Y
	33	128	256	3×3	2	1	Y	Y
Compression (Affinity estimator)	0	1040	512	1×1	1	0	Y	Y
	3	512	256	1×1	1	0	Y	Y
	6	256	128	1×1	1	0	Y	Y
	9	128	64	1×1	1	0	N	Y
	11	64	1	1×1	1	0	N	Y

输入信息包括 frame 中 obejcts 的中心信息，允许我们抽取物体的中心像素值作为代表特征值。将这一概念扩展到网络的特征图，在导致图的空间维度缩减之后，对 object 中心处

的不同卷积层的图进行采样。由于是沿着网络层进行的多个卷积操作，抽样的向量表示不同抽象层面的 object 特征。同时随着网络的深入感受野也越来越大。加上 pooling，感受野会更大，pooling 减少了 feature map 的空间维度，确保从后面层采样的特征另外考虑了完整的对象甚至其周围环境。这些特征是用于跟踪的特征的很好的特性。为了确保这些特征具有足够的表现力，学习大量特征图仍然是必要的。 然而，这将使得通过组合来自多个层的特征而形成的综合特征向量太大而不具有实际用途。为了确保这些特征具有足够的表现力，学习大量特征图仍然是必要的。 然而，这将使得通过组合来自多个层的特征而形成的综合特征向量太大而不具有实际用途。

TABLE 3: Details of *feature dimension reduction* layers in Fig. 1: 3 layers are selected from VGG network. 6 layers are selected from the Extension network. I.C denotes the number of input channels. S.D indicates spatial dimensions of feature maps. O.C is the number of output channels.

Sub-network	Layer index	I.C	S.D	O.C
VGG	16	256	255x255	60
	23	512	113x113	80
	36	1024	56x56	100
Extension network	5	512	28x28	80
	11	256	14x14	60
	17	256	12x12	50
	23	256	10x10	40
	29	256	5x5	30
	35	256	3x3	20

我们通过减少网络中凭经验选择的 9 个层的 feature map 的数量来消除该问题。这种减少是通过从特征提取器的两个主流分支出来的附加卷积层来执行的。附加层使用 1×1 卷积内核来降低维数。Table 3 列出了沿着执行维数降低的卷积层的输入和输出处的 channel 数量的所选九个层的索引。我们的网络连接来自所选九层的特征向量，以形成检测到的对象的 520 维向量。第 t 帧允许有 Nm 个检测对象，因此得到特征矩阵 Ft 520*Nm，相应的第（t-n）帧的特征矩阵 Ft-n 520*Nm。非真实的 object 的特征向量为 0 向量。

3、Affinity estimator

DAN 的该组件的**目的是利用抽取的特征计算 object 之间的联系**。最后网络将 Ft 和 Ft-n 的列整理成 tensor Nm*Nm*（520*2），使得两个特征矩阵的列沿着张量的深度维度以 Nm×Nm 个可能的排列连接，参见图 1 以进行说明。 我们通过使用带有 1*1 卷积核的 5 层卷积压缩网络将该张量映射到矩阵 M Nm*Nm。细节在 Table2 的下半部分。

压缩网络的架构受其输入和输出信号的物理意义的启发。网络将 object 特征的组合编码的 tensor 映射到特征之间的相似性的矩阵。因此，它沿着输入张量的深度执行梯度降维，其中卷积内核不允许特征映射的相邻元素相互影响。通过 DAN 进行前向传播计算 M 矩阵 Nm*Nm。可以看出预测的 M 矩阵必须与 gt data association Lt-n,t 用于计算损失。但是不像 Lt-n,t，M 无法表示输入帧的物体的出现和离开。为了顾及到这种情况，给 M 增加多的一行和一列，得到 M1 Nm*(Nm+1)和 M2 (Nm+1)*Nm。这里我们将行和列向量分别附加到 M，以便保持损失计算的良好定义和物理可解释性。

Network loss:

对 DAN，增强的 M1 的第 m 行将 lt-n 的第 m 个 object 和 lt 中的 Nm+1 个 object 进行关联，其中的最后一个+1 来自于 UI-targets。我们拟合 M1 矩阵上的每行上面的一个独立的概率分布，其通过应用矩阵上的行上的 softmax 操作。因此结果的矩阵 A1 Nm*(Nm+1)的行计算 lt-n 中的每个 object 和 lt 中(包括 UI)的所有 object 之间的概率上的关联。相应的通过列对 M2 进行计算 A2 (Nm+1)*Nm，其列表示从帧 lt 到 lt-n 的类似向后关联。我们强调，

所提出的两帧之间的概率关联的计算允许多个对象在帧之间进入或离开视频。 进入/离开对象的最大数量上限为 Nm - 帧中允许的最大对象数。

我们通过在四个子损失定义 DAN 的损失函数，称为 1）前向丢失 Lf：鼓励从帧 It-n 到帧 It 的正确身份关联。 2）向后丢失 Lb：确保从 It 到 It-n 的正确关联。 3）一致性损失 Lc：拒绝 Lf 和 Lb 之间的任何不一致。 4）汇总损失 La：它抑制关联度预测的非最大前向/后向关联。 下面提供了这些损失的具体定义：

$$\mathcal{L}_f(\boldsymbol{L}_1, \boldsymbol{A}_1) = \frac{\sum_{\text{mat\_el}} (\boldsymbol{L}_1 \odot (-\log \boldsymbol{A}_1))}{\sum_{\text{mat\_el}} (\boldsymbol{L}_1)} \tag{1}$$

$$\mathcal{L}_b(\boldsymbol{L}_2, \boldsymbol{A}_2) = \frac{\sum_{\text{mat\_el}} (\boldsymbol{L}_2 \odot (-\log \boldsymbol{A}_2))}{\sum_{\text{mat\_el}} (\boldsymbol{L}_2)} \tag{2}$$

$$\mathcal{L}_c(\widehat{\boldsymbol{A}_1}, \widehat{\boldsymbol{A}_2}) = ||\widehat{\boldsymbol{A}_1} - \widehat{\boldsymbol{A}_2}||_1 \tag{3}$$

$$\mathcal{L}_a(\boldsymbol{L}_3, \widehat{\boldsymbol{A}_1}, \widehat{\boldsymbol{A}_2}) = \frac{\sum_{\text{mat\_el}} \left(\boldsymbol{L}_3 \odot \left(-\log(\max(\widehat{\boldsymbol{A}_1}, \widehat{\boldsymbol{A}_2}))\right)\right)}{\sum_{\text{mat\_el}} (\boldsymbol{L}_3)} \tag{4}$$

$$\mathcal{L} = \frac{\mathcal{L}_f + \mathcal{L}_b + \mathcal{L}_a + \mathcal{L}_c}{4}. \tag{5}$$

上式中，L_1和L_2分别表示$L_{t-n,\mathrm{n}}$省去最后一行和最后一列的矩阵。$\widehat{A1}$和$\widehat{A2}$分别表示 A1 和 A2 通过省去最后一列和最后一行之后的矩阵。类似的，L_3对$L_{t-n,\mathrm{n}}$省去最后一行和最后一列得到。圆圈的操作符表示 Hadamard product，Σ表示矩阵的系数之和。max 和 log 操作是对单个元素进行操作的。

我们将四个子损失的平均值作为实际损失。我们网络的总体成本函数被定义为训练数据丢失的预期值。上述四个子损失是针对我们的问题精心设计的。在前向和后向损失中，为了通过使用距离度量来近似对应的Lq，我们对通过Aq的相关系数计算得到的概率进行最大化，其中$q \in \{1, 2\}$。我们认为这种策略比最小化二元矩阵（Lq）和概率矩阵（Aq）之间的距离更为明智。同样，鉴于$\widehat{A1}$和$\widehat{A2}$之间的差异预计很小，我们采用 l1 距离而不是更常用的 l2 距离来表示一致性损失。一旦训练了 DAN，我们用它来计算输入帧对的关联度矩阵作为

$$\boldsymbol{A} \in \hat{\mathbb{R}}^{N_m \times N_m + 1} = \hat{\mathcal{A}}(\max(\widehat{\boldsymbol{A}_1}, \widehat{\boldsymbol{A}_2}))$$

其中倾斜的 A 表示追加 A1 的第（Nm + 1）列到其参数中的矩阵。在我们定义关联度矩阵 A 时使用的最大操作也证明了为计算总损失而执行的最大化。因此，上面定义的四个子损失是互补的，导致 gt 数据关联的系统近似。

4、DAN deployment

尽管 DAN 的 feature extractor 作为双流网络进行训练，但是实际部署是作为单流模型实现的。因为参数在两个流之间共享。在下图中，通过分别描述两个组件来表示 DAN 的部署。该网络将单帧I_t以及 object 的中心位置作为输入。feature extractor 计算帧的特征矩阵，并将其传递给 affinity estimator。后者使用前一帧的特征矩阵，比如I_{t-n}来计算帧对的拼接的张量$\Psi_{t-n,t}$。 然后通过网络的简单前向传递和连接操作将张量映射到关联度矩阵，如上所述。因此，每个帧仅通过对象检测器和特征提取器一次，但是这些特征被多次用于计算具有成对的多个其他帧的关联度。

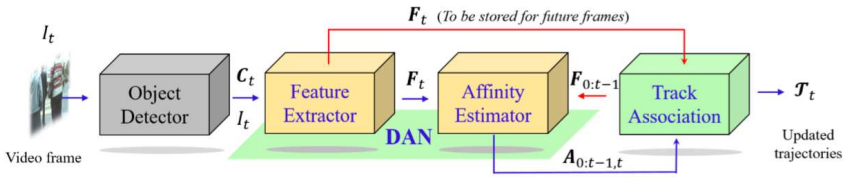


Fig. 3: Deep tracking with DAN deployment: For the t^{th} frame $\boldsymbol{I}_t$, the object centers $\mathcal{C}_t$ provided by the detectors are used to compute the feature matrix $\boldsymbol{F}_t$ with *one* stream Feature Extractor of DAN. The $\boldsymbol{F}_t$ is paired with each of the last t feature matrices $\boldsymbol{F}_{0:t-1}$, and each pair is processed by the Affinity Estimator to compute the same number of affinity matrices $\boldsymbol{A}_{0:t-1,t}$. The $\boldsymbol{F}_t$ is also stored for computing affinity matrices in the future. The trajectory set $\mathcal{T}_t$, is updated by associating the current frame with t previous frames using the computed affinity matrices.

为了将当前帧中的对象与多个先前帧相关联，我们存储帧的特征矩阵及其时间戳。 在第 0 帧之后，初始化我们的方法并得到F_0，我们可以使用它们的特征矩阵计算当前帧中的 object 与任何先前帧中的对象之间的关联度。 如上图所示，计算的关联度矩阵用于通过深入回顾先前帧来更新轨迹集。

深度跟踪关联如下进行。 我们使用与I_0中检测到的对象数一样多的轨迹来初始化轨道集 T0。 这里的轨迹是一组 2-元组，每个元组包含帧的时间戳和 object id。 借助于将匈牙利算法[34]应用于累加器矩阵$\Lambda_t \in \mathbb{R}^{Z(T_{t-1})\times(Z(C_t)+1)}$，在第 t 时间戳处更新轨迹集。 使用的累加器矩阵整合当前帧中的对象与先前帧之间的关联度。 索引(i,j)处的Λ_t系数是先前δ_b帧的轨道集T_{t-1}中的第 i 个 object 与第 t 帧中的第 j 个 object 的关联度之和，其中δ_b是我们的方法的参数。

在每个时间戳，我们能够使用 DAN 有效地计算高达δ_b个关联度矩阵，以回顾现有的轨迹。 为简化起见，我们在上图中使$\delta_b = t$。 成功将匈牙利算法应用于我们的问题的一个微妙问题是我们允许多个对象在其帧与帧之间留下视频。 因此，可以将多个轨迹分配给我们的累加器矩阵中允许的单个 UI 目标列（继承自关联度矩阵）。 我们通过重复Λt 的最后一列来处理这个问题，直到 Tt-1 中的每个轨迹被分配给增强矩阵的唯一列。 这确保了所有未识别的轨迹可以映射到未识别的对象。

总的来说，我们的跟踪器是一种在线方法，因为它不使用未来的帧来预测物体轨迹。 因此，它可以与连续视频流一起使用。 在这种情况下的一个实际问题是在很长的时间间隔内可能导致很长的粘性。 参数δ_b也限制了我们与轨道相关联的最大时间戳数。 如果轨迹的帧数超过δ_b，我们将删除轨道中最旧的节点。 类似地，对于从视频中消失的对象，我们允许在从当前轨道集移除其轨迹之前等待δ_w帧。 出于纯粹的实用原因，我们在框架中引入了这些参数。 它们的值可以根据跟踪器的板上的计算和存储容量进行调整。