Deep Affinity Network for Multiple Object Tracking

ShiJie Sun, Naveed Akhtar, HuanSheng Song, Ajmal Mian, Mubarak Shah

Abstract—Multiple Object Tracking (MOT) plays an important role in solving many fundamental problems in video analysis and computer vision. Most MOT methods employ two steps: Object Detection and Data Association. The first step detects objects of interest in every frame of a video, and the second establishes correspondence between the detected objects in different frames to obtain their tracks. Object detection has made tremendous progress in the last few years due to deep learning. However, data association for tracking still relies on hand crafted constraints such as appearance, motion, spatial proximity, grouping etc. to compute affinities between the objects in different frames. In this paper, we harness the power of deep learning for data association in tracking by jointly modeling object appearances and their affinities between different frames in an end-to-end fashion. The proposed Deep Affinity Network (DAN) learns compact, yet comprehensive features of pre-detected objects at several levels of abstraction, and performs exhaustive pairing permutations of those features in any two frames to infer object affinities. DAN also accounts for multiple objects appearing and disappearing between video frames. We exploit the resulting efficient affinity computations to associate objects in the current frame deep into the previous frames for reliable on-line tracking. Our technique is evaluated on popular multiple object tracking challenges MOT15, MOT17 and UA-DETRAC. Comprehensive benchmarking under twelve evaluation metrics demonstrates that our approach is among the best performing techniques on the leader board for these challenges. The open source implementation of our work is available at https://github.com/shijieS/SST.git.

Index Terms—Multiple object tracking, Deep tracking, Deep affinity, Tracking challenge, On-line tracking.

1 Introduction

Tracking multiple objects in videos [1] requires detection of objects in individual frames and associating those across multiple frames. In Multiple Object Tracking (MOT), natural division of the task between object *detection* and *association* allows off-the-shelf deep learning based object detectors [2], [3], [4], [5] to be used for tracking. However, the problem of object association is yet to fully benefit from the advances in deep learning due to the peculiar nature of the task of affinity computation for the objects that can be multiple frames apart in video.

Currently, a widely used tracking approach [6], [7], [8], [9], [10] first computes a representation model of pre-detected objects in a video. Later, that model is used to estimate affinities between the objects across different frames. The approaches following this pipeline exploit several distinct types of representation models, including; appearance models [10], [11], [12], [13], motion models [14], [15], [16], [17], and composite models [9], [18], [19]. The appearance models focus on computing easy-to-track object features that encode appearances of local regions of objects or their bounding boxes [20], [21], [22]. Currently, hand-crafted features are common for appearance modeling. However, such features are not robust to illumination variations and occlusions. Moreover, they also fall short on discriminating distinct objects with relatively high similarity.

The motion models encode object dynamics to predict object locations in the future frames. Motion modeling techniques [14], [23], [24] use linear models under constant velocity assumption [14]. These models exploit the smoothness of object's velocity, position or acceleration in the video frames. Motion model based tracking has also witnessed attempts that account for non-linearity [25] to better represent complexity of the real-world motions. Nevertheless, both linear and non-linear models fail to handle long inter-frame object occlusions well. Moreover, they are also challenged by the scenarios of complex motions. Hence, composite model based tracking [9], [18], [19] aims at striking a balance between motion and appearance modeling. However, such a balance is hard to achieve in real-world [26].

To address the challenges of multiple object tracking, we leverage the representation power of Deep Learning. We propose a Deep Affinity Network (DAN), as shown in Fig. 1, that jointly learns target object appearances and their affinities in a pair of video frames in an end-to-end fashion. The appearance modeling accounts for hierarchical feature learning of objects and their surroundings at multiple levels of abstraction. The DAN estimates affinities between the objects in a frame pair under exhaustive permutations of their compact features. The computed affinities also account for the cases of multiple objects entering or leaving the videos between the two frames. This is done by enabling the softmax layer of DAN to separately look forward and backward in time for un-identifiable objects in the frame pair. We propose an appropriate loss function for DAN to account for this unique ability. The DAN models object appearance with a two-stream convolutional network with shared parameters, and estimates object affinities in its later layers using the hierarchical features from its initial layers. The overall network does not assume the input frame pairs to appear consecutively in a video. This promotes robustness against object occlusions in the induced model.

We exploit the efficient affinity computation by the proposed model to associate objects in a given video frame to the objects in multiple previous frames for reliable trajectory

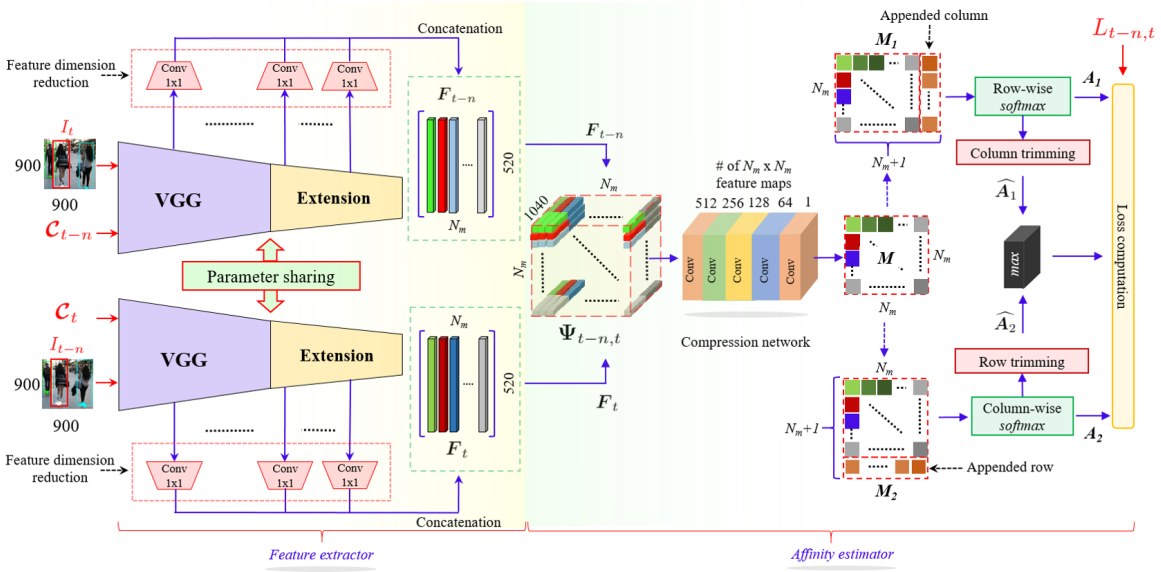

Fig. 1: Schematics of Deep Affinity Network (DAN): A pair of video frames $\boldsymbol{I}_t$ and $\boldsymbol{I}_{t-n}$, which are n time stamps apart are input to the network along with the sets of centers $\boldsymbol{\mathcal{C}}_t$, $\boldsymbol{\mathcal{C}}_{t-n}$ of pre-detected objects in those frames. The frame pair is processed by two extended VGG-like networks with shared parameters. The number of feature maps in nine selective layers of these networks are reduced using 1×1 convolutional kernels. Compact features extracted from those maps are concatenated to form 520-dimensional feature vectors. Exhaustive permutations of those vectors in the feature matrices $\boldsymbol{F}_t$ and $\boldsymbol{F}_{t-n}$ are encoded in a tensor $\boldsymbol{\Psi}_{t-n,t} \in \mathbb{R}^{1040 \times N_m \times N_m}$, where N_m is the number of objects in each frame. The tensor $\boldsymbol{\Psi}_{t-n,t}$ is mapped to a matrix $\boldsymbol{M} \in \mathbb{R}^{N_m \times N_m}$ using five convolution layers. To account for multiple identities leaving and entering between the frames, $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ are formed by appending an extra column and an extra row to $\boldsymbol{M}$. Row- and column-wise softmax is performed over $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ respectively. The resulting matrices $\boldsymbol{A}_1, \boldsymbol{A}_2$ and their column- and row-trimmed variants $\widehat{\boldsymbol{A}}_1, \widehat{\boldsymbol{A}}_2$ are employed in network loss computation using the ground truth data association matrix $\boldsymbol{L}_{t-n,t}$. Affinity matrix for a pair of frames is predicted using the matrices $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ predicted by DAN.

generation using the Hungarian algorithm [27]. This results in accurate on-line multiple object tracking. The proposed approach performs tracking at 6.3 frames per second on the popular MOT15 [28], MOT17 [29] and UA-DETRAC [30], [31] challenge datasets, while surpassing the existing leading approaches on a majority of the evaluation metrics. Our technique significantly outperforms its nearest competitors for on-line multiple pedestrian tracking.

The rest of the article is organized as follows. In Section 2, we review the related literature and datasets of multiple object tracking. In Section 3, we introduce our tracking technique and present the DAN. Section 4 details the results of our approach on tracking challenges. We present ablations analysis for the developed technique in Section 5 and draw conclusions in Section 6.

2 Related Work

Multiple Object Tracking (MOT) has attracted significant interest of researchers in recent years. For a general review of this research direction, we refer to [1], where Luo et al. organized the existing literature under multiple criteria and summarized the recent techniques based on components of the tracking problem. We also refer to the survey by Emami et al. [32] who viewed MOT as an assignment problem and unified a variety of tracking approaches under this formulation. Below, we focus more on the contributions that consider MOT from a data association perspective, and also review recent deep learning based techniques in MOT that relate to our method.

Tracking-by-detection is a generic framework employed by several multiple object trackers [33], [34], [35]. In this framework, the objects are first detected and then associated in different frames. Due to parallel developments of multiple object detectors [2], [3], [4], [5], techniques following this line of approach focus more on the data association aspect of tracking. These techniques can be broadly categorized as local and global tracking methods. The local methods [36], [37], [38] consider only two frames for data association. This makes them computationally efficient, however, their performance is susceptible to tracking-irrelevant factors such as camera motion, and pose variation etc.

In contrast to local methods, global techniques [39], [40], [41] perform data association using a larger number of frames. Recent methods in this direction cast data association into a network flow problem [42], [43], [44], [45]. For instance, Berclaz et al. [44] solved a constrained flow optimization problem for multiple object tracking, and used the k-shortest paths algorithm for associating the tracks. Chari et al. [46] added a pairwise cost to the min-cost network flow framework and proposed a convex relaxation solution with a rounding heuristic for tracking. Similarly, Shitrit et al. [45] used multi-commodity network flow for MOT. Although popular, this line of methods rely on object detectors rather strongly [33] that makes them less desirable for scenarios where occlusions or misdetections are encountered frequently. Shu et al. [38] handled occlusions up to a scale by extending a part based human detector [47]. There have also been attempts to use dense detections with-

out non-maximum suppression for tracking [48], [49]. One major goal of these techniques is to mitigate the problems caused by occlusions and close proximity of the targets that are not handled well by the detectors used for tracking.

There are also techniques that mitigate the problems with object detectors in tracking by employing an on-line trainable classifier for the target objects [50], [51], [52]. Application of on-line classifiers is more beneficial for single object tracking. However, this notion has failed to gain popularity in MOT, and has only been successfully applied in limited scenarios [53]. There are also instances of MOT under the tracking-by-detection framework that explicitly focus on mitigating the adverse affects of occlusions. For example, Milan et al. [54] employed a continuous energy minimization framework for MOT that incorporates occlusion explicit reasoning and appearance modeling. To handle occlusions and clutter, Wen et al. [19] proposed a data association technique based on undirected hierarchical relation hyper-graph. Bochinski et al. [55] leveraged the intersection-over-union and predefined thresholds for associating objects in video frames. Focusing on computational efficiency, they showed acceptable tracking performance with a speed of up to 100 fps. Chen et al. [56] proposed a multiple hypothesis tracking method by accounting for scene detections and detection-detection correlations between video frames. Their method can handle false trajectories while dealing with close object hypotheses.

In the context of deep learning based tracking, models pre-trained for classification tasks are popular [57], [58], [59], [60], [61]. These models are used to extract object features that are employed for object association in tracking. Bertinetto et al. [57] proposed to use a fully convolutional Siamese Network [62] for single object tracking. Similarly, Bea et al. [18] modified the Siamese Network to learn discriminative deep representations for multiple object tracking with object association. They combined on-line transfer learning with the modified network to fine-tune the latter for on-line tracking. Son et al. [58] proposed a quadruplet convolutional neural network that learns to associate objects detected in different video frames. Their network is used by a minimax label propagation method to associate the targets. Schulter et al. [59] also proposed a network that is trained for data association in the context of multiple object tracking. Feichtenhofer et al. [60] proposed a multi-function CNN for simultaneous detection and tracking under the detection framework R-FCN [63]. Insafutdinov et al. [61] proposed a deep learning based method for pose estimation and tracking. They trained a model that groups body parts and tracks a person using the head joint. However, their method underperforms if heads are occluded.

Li et al. [64] proposed a deep learning based technique for single object tracking that uses a Siamese regional proposal network to perform real-time tracking. Similarly, Zhang et al. [65] proposed a long-term single object tracking framework based on two networks. The first network, that performs regression, generates a series of candidate target objects and their similarity scores for the frames. A second verification network then evaluates these candidates for tracking. Whereas these works contribute specialized deep networks for the tracking problem, their scope is limited to single object tracking.

In deep learning based tracking, one important aspect of Multiple Object Tracking research is collection and annotation of the data itself. This direction has also seen dedicated contributions from the tracking research community. A summary of the existing multiple object tracking datasets can be found in [1]. Among those datasets, two are particularly relevant to this work, namely the MOT dataset [66] and the UA-DETRAC dataset [30]. Introduced by Milan et al. [66], the MOT dataset is used as a benchmark by a popular recent multiple object tracking challenge MOT17 (https://motchallenge.net/data/MOT17/). It provides a variety of real-world videos for pedestrian tracking. The UA-DETRAC challenge (https://detrac-db.rit.albany.edu/) uses the large scale UA-DETRAC dataset [30] that contains videos of traffic in the real-world conditions. We provide further details on both datasets in Section 4. Other popular examples of object tracking datasets include KITTI [67] and PETS [68].

The representation power of deep learning makes it an attractive choice for appearance modeling based tracking techniques, especially under the tracking-by-detection framework. Nevertheless, the data association component of this framework is yet to fully benefit from deep learning, especially in a manner that association modeling is also tailored to appearance modeling in the overall framework. This work fills this gap by jointly modeling the appearance of objects in a pair of frames and learning their associations in those frames in an end-to-end manner. The proposed Deep Affinity Network performs comprehensive appearance modeling and uses it further to estimate object affinities in the frame pair. The efficient affinity computation allows our technique to look back into multiple previous frames for object association. This keeps the proposed technique robust to object occlusions for tracking.

3 Proposed Approach

We harness the representation power of deep learning to perform on-line multiple object tracking. Central to our technique is a CNN-based Deep Affinity Network (DAN), see Fig. 1, that jointly models object appearances and their affinities across two different frames that are not necessarily adjacent. The overall approach exploits the efficient affinity estimation by DAN to associate objects in the current frame to those in multiple previous frames to compute reliable trajectories. We provide a detailed discussion on our approach below. However, we first introduce notations and conventions for a concise description in the text and figures.

Notations:

- $\boldsymbol{I}_t$ denotes the t^{th} video frame under 0-based indexing.
- $t-n:t$ denotes an interval from $t-n$ to t.
- A subscript $t-n,t$ indicates that the entity is computed for the frame pair $\boldsymbol{I}_{t-n}, \boldsymbol{I}_t$.
- $\boldsymbol{\mathcal{C}}_t$ is the set of center locations of the objects in the t^{th} frame with $\mathcal{C}_t^i$ as its i^{th} element.
- $\boldsymbol{F}_t$ is the feature matrix associated with the t^{th} frame, where its i^{th} column F_t^i is the feature vector of the i^{th} object in that video frame.
- $\boldsymbol{\Psi}_{t-n,t}$ is a tensor that encodes all possible pairings of the columns of $\boldsymbol{F}_{t-n}$ and $\boldsymbol{F}_t$ along its depth dimension.

- $\boldsymbol{L}_{t-n,t}$ denotes a binary data association matrix encoding the correspondence between the objects detected in frames $\boldsymbol{I}_{t-n}$ and $\boldsymbol{I}_t$. If object '1' in $\boldsymbol{I}_{t-n}$ corresponds to the n^{th} object in $\boldsymbol{I}_t$, then the n^{th} element of the first row of $\boldsymbol{L}_{t-n,t}$ is non-zero.
- $\boldsymbol{A}_{t-n,t}$ denotes the affinity matrix that encodes similarities between the bounding boxes in $(t-n)^{\text{th}}$ frame and the t^{th} frame.
- $\boldsymbol{\mathcal{T}}_t$ denotes the set of trajectories or tracks until the t^{th} time stamp. The i^{th} element of this set is itself a set of 2-tuples, containing indices of frames and detected objects. For example, $\mathcal{T}_t^i = \{(0,1),(1,2)\}$ indicates the short track connecting the 1^{st} object in frame 0 and the 2^{nd} object in frame 1.
- $\mathcal{Z}(\cdot)$ is an operator that computes the number of elements in a set/matrix in its argument.
- $\boldsymbol{\Lambda}_t \in \mathbb{R}^{\mathcal{Z}(\boldsymbol{\mathcal{T}}_{t-1})\times(\mathcal{Z}(\boldsymbol{\mathcal{C}}_t)+1)}$ is an accumulator matrix whose coefficient at index (i,j) integrates the affinities of the i^{th} identity in track set $\boldsymbol{\mathcal{T}}_{t-1}$ to the j^{th} object in the t^{th} frame over previous δ_b frames.
- N_m denotes the allowed maximum number of objects in a frame.
- B denotes the batch size during training.

Conventions:
The shape of the output at each network layer is described as $Batch \times Channel \times Width \times Height$. For the sake of brevity, we often leave out the $Batch$ dimension.

3.1 Object detection and localization

The object detection stage in our approach expects a video frame as input and outputs a set of bounding boxes for the target objects in that frame. For the t^{th} frame, we compute object center locations $\boldsymbol{\mathcal{C}}_t$ using the output bounding boxes. We evaluate our method (see Section 4) for different on-line challenges in multiple object tracking that provide their own object detectors. For the MOT17 challenge [29], we use the provided Faster R-CNN [2] and SDP [69] detectors; for the MOT15 [28], we use [70]; and for the UA-DETRAC [30], [31], we use the EB detector [71]. Our choices of detectors are entirely based on the challenges used to evaluate our method. However, the proposed framework is also compatible with the other existing multiple object detectors. Since our main contribution is in object tracking and not detection, we refer to the original works for the detectors.

3.2 Deep Affinity Network (DAN)

We model the appearance of objects in video frames and compute their cross-frame affinities using the Deep Affinity Network (DAN), shown in Fig. 1. To align our discussion with the existing tracking literature, we present the proposed network as two components, namely (a) Feature extractor, and (b) Affinity estimator. However, the overall proposed network is end-to-end trainable. The DAN training requires video frame $\boldsymbol{I}_t$ along with its object centers $\boldsymbol{\mathcal{C}}_t$; and video frame $\boldsymbol{I}_{t-n}$ along with its object centers $\boldsymbol{\mathcal{C}}_{t-n}$. We do not restrict the two frames to appear consecutively in video. Instead, we allow them to be n time stamps apart, such that $n \in \mathbb{N} \overset{\text{rand}}{\sim} [1, N_V]$. Whereas our network is eventually deployed to track objects in consecutive video frames, training it with non-consecutive frames benefits the overall approach in reliably associating objects in a given frame to those in multiple previous frames. DAN also requires the ground truth binary data association matrix $\boldsymbol{L}_{t-n,t}$ of the input frame pair for computing the network cost during training. The inputs to DAN are shown in red color in Fig. 1. We describe working details of the internal components of our network below after discussing the data preparation.

3.2.1 Data preparation

Multiple object tracking datasets e.g. [29], [31] often lack in fully capturing the aspects of camera photometric distortions, background scene variations and other practical factors to which tracking approaches should remain robust. For model based approaches, it is important that the training data contains sufficient variations of such tracking-irrelevant factors to induce robustness in the learned models. Hence, we perform the following preprocessing steps over the available data.

1) *Photometric distortions*: Each pixel of a video frame is first scaled by a random value in the range $[0.7, 1.5]$. The resulting image is converted to HSV format, and its Saturation channel is again scaled by a random value in $[0.7, 1.5]$. The frame is then converted back to RGB format and rescaled by a random value in the same range. This process of photometric distortion is similar to [72], that also inspires the used range values.
2) *Frame expansion*: We expand the frames by a random ratio sampled in the range $[1, 1.2]$. The expansion results in increasing the frame sizes. To achieve the new size, we pad the original frame with extra pixels. The value of these extra pixels is set to the mean pixel value of the training data.
3) *Cropping*: We crop the frames using cropping ratios randomly sampled in the range $[0.8, 1]$. We keep only those crops that contain the center points of all the detected boxes in the original frames.

Each of the above steps is applied to the frame pairs sequentially with a probability 0.3. The frames are then resized to fixed dimensions $H \times W \times 3$, and horizontally fliped with a probability of 0.5. The overall strategy of modifying the training data is inspired by Liu et al. [72] who alter images to train an object detector. However, different from [72], we simultaneously process two frames by applying the above-mentioned transformations to them.

The resulting processed frames are used as inputs to DAN along with the associated object centers computed by the detector. This stage also accounts for object occlusions. Fully occluded objects in the training data are ignored by our training procedure at this point. We set the visibility threshold of 0.3 to treat an object as fully occluded. The remaining partially occluded objects become the positive samples of objects with occlusions. We compute data association matrices for the frames by introducing an upper bound N_m on the maximum number of objects allowed in a given frame. In our experiments, $N_m = 80$ proved a generous bound for the benchmark challenges. For consistency, we introduce additional rows and columns (with all zeros) in the data association matrix corresponding to *dummy* bounding boxes in each video frame so that all

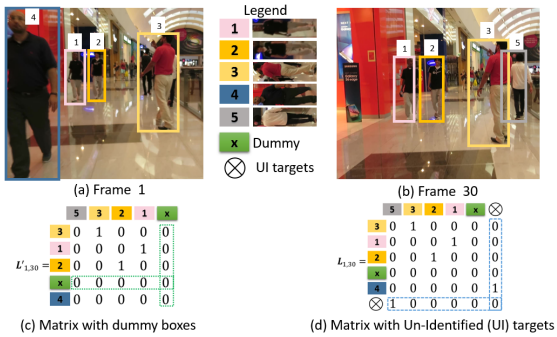

Fig. 2: Illustration of data association matrix between frame 1 and 30, with $N_m = 5$. Frame 1 and 30 in (a) and (b) jointly contain 5 detected objects. (c) Creation of an intermediate matrix that considers *dummy* objects (rows and columns with zeros) to achieve $N_m = 5$ per frame. (d) Augmentation with extra column and row to include Un-Identified (UI) targets (objects leaving and entering respectively) between the two frames.

TABLE 1: The input data for training DAN. The tensor dimensions are expressed as Channels × Height × Width.

Input	Dimensions/Size
$\boldsymbol{I}_{t-n}, \boldsymbol{I}_t$	$3 \times H \times W$
$\boldsymbol{L}_{t-n,t}$	$1 \times (N_m + 1) \times (N_m + 1)$
$\boldsymbol{\mathcal{C}}_t, \boldsymbol{\mathcal{C}}_{t-n}$	N_m

the frames eventually contain N_m objects and the matrix size is $N_m \times N_m$. Figure 2 illustrates our approach for the construction of data association matrix, where we let $N_m = 5$ for simplicity. In the figure, frames 1 and 30 contain four detected persons each, amounting to five distinct identities. Figure 2c shows the construction of an intermediate matrix $\boldsymbol{L}'_{1,30}$ for the frames with a row and a column for dummy bounding boxes. In Fig. 2d, the association matrix is augmented with an extra row and an extra column, labeled as *un-identified* targets (UI targets). The augmented column accounts for currently tracked objects leaving the video and the augmented row accounts for the new objects entering the video. We eventually use the form of ground truth association matrix shown in Fig. 2d to train the DAN. In the shown illustration, the last column of the augmented matrix has a 1 for object-4 because it has left and the last row has a 1 for object-5 because it has appeared in Frame 30. Notice that, using this convention, DAN is able to account for multiple objects leaving and entering the video i.e. by placing 1's at multiple rows in the last column and by placing 1's at multiple columns in the last row, respectively. After data preparation, the entities available as inputs to our network are summarized in Table 1.

3.2.2 Feature extractor

We refer to the first major component of DAN as *feature extractor* given its functionality. This sub-network models comprehensive, yet compact features of the detected objects in video frames. As shown in Fig. 1, the feature extraction is performed by passing pairs of video frames and object centers through two streams of convolution layers. These streams share the model parameters in our implementation, whereas their architecture is inspired by VGG16 network [73]. We use the VGG architecture after converting its fully-connected and softmax layers to convolution layers. This modification is made because the spatial features of objects, which are of more interest in our task, are better encoded by convolution layers. Compared to the original VGG16, the input frame size for our network is much larger (i.e. $3 \times 900 \times 900$) due to the nature of the task at hand and the available tacking datasets. Consequently, we are still able to compute 56×56 feature maps after the last layer of the modified VGG network. In Fig. 1, we index the last layer of VGG as the 36^{th} layer under the convention that Batch Normalization [74] and ReLU activations [75] are counted as separate layers. We refer to [73] for the details on the original VGG architecture.

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

We reduce the spatial dimensions of our feature maps beyond 56×56, after the VGG layers, by introducing further convolution layers. The 36-layer *Extension* network gradually reduces feature maps to size 3×3. Our choice of gradually reducing the feature maps to size 3×3 is empirical. We conjecture that it results in better performance because it ensures comprehensive appearance modeling at multiple levels of abstraction. Due to their large receptive fields, the latter layers of the Extension sub-network are able to better model object surroundings, which helps in the overall performance of our technique. The architectural details of the extension network are provided in the top half of Table 2. The table counts the output of VGG network as input to the first layer of the extension that is indexed 0. Since the Batch Normalization and ReLU are counted as separate layers, the first column of the table increments indexes with step size 3.

Knowing the object center locations in the input frames (as $\boldsymbol{\mathcal{C}}_t$ and $\boldsymbol{\mathcal{C}}_{t-n}$) allows us to extract center pixels of the objects as their representative features. We extend this notion to the feature maps of our network and sample the maps of different convolution layers at object centers after accounting for reduction in their spatial dimensions. Due to multiple sequential convolution operations along the network layers, the sampled vectors represent object

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

features at different levels of abstraction. These are highly desirable characteristics of features for tracking. To ensure such features are sufficiently expressive, it remains imperative to learn a large number of feature maps. However, this would make the comprehensive feature vector formed by combining features from multiple layers too large to be practically useful.

We side-step this issue by reducing the number of feature maps of nine empirically selected layers in our network. This reduction is performed with additional convolution layers branching out from the two main streams of the feature extractor (see Fig. 1). The additional layers use 1×1 convolution kernels for dimensionality reduction. Table 3 lists indices of the selected nine layers along the number of channels at input and output of the convolution layers performing dimensionality reduction. Our network concatenates feature vectors from the selected nine layers to form a 520-dimensional vector for a detected object. By allowing N_m detections in the t^{th} frame, we obtain a feature matrix $\boldsymbol{F}_t \in \mathbb{R}^{520 \times N_m}$. Correspondingly, we also construct $\boldsymbol{F}_{t-n} \in \mathbb{R}^{520 \times N_m}$ for the $(t-n)^{\text{th}}$ frame. Recall that these matrices also contain features for dummy objects that actually do not exist in the video frames. We implement these features as zero vectors.

3.2.3 Affinity estimator

The objective of this component of DAN is to encode affinities between the objects using their extracted features. To that end, the network arranges the columns of $\boldsymbol{F}_t$ and $\boldsymbol{F}_{t-n}$ in a tensor $\boldsymbol{\Psi} \in \mathbb{R}^{N_m \times N_m \times (520 \times 2)}$, such that the columns of the two feature matrices are concatenated along the depth dimension of the tensor in $N_m \times N_m$ possible permutations, see Fig. 1 for illustration. We map this tensor onto a matrix $\boldsymbol{M} \in \mathbb{R}^{N_m \times N_m}$ through a *compression network* that uses 5 convolution layers with 1×1 kernels. Specifications of this network are given in the bottom half of Table 2.

The architecture of the compression network is inspired by the physical significance of its input and output signals. The network maps a tensor that encodes combinations of object features to a matrix that codes similarities between the features (hence, the objects). Thus, it performs a gradual dimension reduction along the depth of the input tensor with convolutional kernels that do not allow neighboring elements of feature maps to influence each other. For a moment, consider a forward-pass through the DAN until computation of the matrix $\boldsymbol{M} \in \mathbb{R}^{N_m \times N_m}$. The remainder of our network must compare this matrix to the ground truth data association matrix $\boldsymbol{L}_{t-n,t} \in \mathbb{R}^{(N_m+1)\times(N_m+1)}$ for loss computation. However, unlike $\boldsymbol{L}_{t-n,t}$, $\boldsymbol{M}$ does not account for the objects that enter or leave the video between the two input frames. To take care of those objects, we also append an extra column and an extra row to $\boldsymbol{M}$ to form matrices $\boldsymbol{M}_1 \in \mathbb{R}^{N_m \times (N_m+1)}$ and $\boldsymbol{M}_2 \in \mathbb{R}^{(N_m+1)\times N_m}$, respectively. This is analogous to the augmentation of $\boldsymbol{L}_{t-n,t}$ explained in Section 3.2.1. However, here we separately append the row and column vectors to $\boldsymbol{M}$ in order to keep the loss computation well-defined and physically interpretable. The vectors appended to $\boldsymbol{M}$ take the form $\mathbf{v} \in \mathbb{R}^{N_m} = \gamma\mathbf{1}$, where $\mathbf{1}$ is a vector of ones, and γ is a hyper-parameter of the proposed DAN.

Network loss:

In our formulation, the m^{th} row of $\boldsymbol{M}_1$ associates the m^{th} identity in frame $\boldsymbol{I}_{t-n}$ to $N_m + 1$ identities in frame $\boldsymbol{I}_t$, where $+1$ results from the unidentified (UI) objects in $\boldsymbol{I}_t$. We fit a separate probability distribution over each row of $\boldsymbol{M}_1$ by applying a *row*-wise softmax operation over the matrix. Thus, a row of the resulting matrix $\boldsymbol{A}_1 \in \mathbb{R}^{N_m \times (N_m+1)}$ encodes probabilistic associations between an object in frame $\boldsymbol{I}_{t-n}$ and all identities in frame $\boldsymbol{I}_t$. We correspondingly apply a *column*-wise softmax operation over $\boldsymbol{M}_2$ to compute $\boldsymbol{A}_2 \in \mathbb{R}^{(N_m+1)\times N_m}$, whose columns signify similar *backward* associations from frame $\boldsymbol{I}_t$ to $\boldsymbol{I}_{t-n}$. It is emphasized that our probabilistic object association allows for multiple objects entering or leaving the video between the two frames of interest. The maximum number of objects that can enter or leave video is upper-bounded by N_m - the maximum number of allowed objects in a frame.

We define the loss function for DAN with the help of four sub-losses, referred to as 1) Forward-direction loss $\mathcal{L}_f$: that encourages correct identity association from $\boldsymbol{I}_{t-n}$ to $\boldsymbol{I}_t$. 2) Backward-direction loss $\mathcal{L}_b$: that ensures correct associations from $\boldsymbol{I}_t$ to $\boldsymbol{I}_{t-n}$. 3) Consistency loss $\mathcal{L}_c$: to rebuff any inconsistency between $\mathcal{L}_f$ and $\mathcal{L}_b$. 4) Assemble loss $\mathcal{L}_a$: that suppresses non-maximum forward/backward associations for affinity predictions. Concrete definitions of these losses are provided below:

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

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

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

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

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

In the above equations, $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ are the trimmed versions of $\boldsymbol{L}_{t-n,t}$ constructed by ignoring the last row and the last column of $\boldsymbol{L}_{t-n,t}$, respectively. $\widehat{\boldsymbol{A}_1}$ and $\widehat{\boldsymbol{A}_2}$ denote the matrices $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ trimmed to the size $N_m \times N_m$ by respectively

dropping the last column and the last row. Similarly, $\boldsymbol{L}_3$ drops out both the last row and the last column of $\boldsymbol{L}_{t-n,t}$. The operator $\odot$ denotes the Hadamard product, and $\sum_{\text{coeff}}(.)$ sums up all coefficients of the matrix in its argument to a scalar value. The *max* and *log* operations are also performed element-wise. In the above defined losses, one noteworthy observation is regarding separate losses for 'Forward' and 'Backward' associations. We define these losses using two individual matrices $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$. With an extra column and 'row'-wise softmax operation applied to its coefficients, $\boldsymbol{A}_1$ is inherently dissimilar to $\boldsymbol{A}_2$ that is computed with a 'column'-wise softmax operation and has an extra row. To account for different matrix sizes and more importantly, the difference in the physical significance of their coefficients, we define individual losses for them.

We compute the final loss $\mathcal{L}$ as the mean value of the four sub-losses. The overall cost function of our network is defined as the Expected value of the training data loss. The afore-mentioned four sub-losses are carefully designed for our problem. In the Forward and Backward direction losses, instead of forcing $\boldsymbol{A}_q$, where $q \in \{1, 2\}$; to approximate corresponding $\boldsymbol{L}_q$ by using a distance metric, we maximize the probabilities encoded by the relevant coefficients of $\boldsymbol{A}_q$. We argue that this strategy is more sensible than minimizing a distance between a binary matrix $\boldsymbol{L}_q$ and a probability matrix $\boldsymbol{A}_q$. Similarly, given the difference between $\widehat{\boldsymbol{A}_1}$ and $\widehat{\boldsymbol{A}_2}$ is expected to be small, we employ ℓ_1-distance instead of the more commonly used ℓ_2-distance for the Consistency loss. Once the DAN is trained, we use it to compute the affinity matrix for an input frame pair as $\boldsymbol{A} \in \mathbb{R}^{N_m \times N_m+1} = \mathcal{A}(\max(\widehat{\boldsymbol{A}_1}, \widehat{\boldsymbol{A}_2}))$, where $\mathcal{A}(.)$ appends the $(N_m + 1)^{\text{th}}$ column of $\boldsymbol{A}_1$ to the matrix in its argument. The max operation used in our definition of the affinity matrix $\boldsymbol{A}$ also justifies the maximization performed to compute the Assemble loss. Thus, the four sub-losses defined above are complementary that result in a systematic approximation of the ground truth data association.

3.3 DAN deployment

Whereas the *feature extractor* component of DAN is trained as a two-stream network, it is deployed as a one-stream model in our approach. This is possible because the parameters are shared between the two streams. In Fig. 3, we illustrate the deployment of DAN by showing its two major components separately. The network expects a single frame $\boldsymbol{I}_t$ as its input, along the object center locations $\mathcal{C}_t$. The feature extractor computes the feature matrix $\boldsymbol{F}_t$ for the frame and passes it to the *affinity estimator*. The latter uses the feature matrix of a previous frame, say $\boldsymbol{I}_{t-n}$ to compute the permutation tensor $\boldsymbol{\Psi}_{t-n,t}$ for the frame pair. The tensor is then mapped to an affinity matrix by a simple forward pass through the network and a concatenation operation, as described above. Thus each frame is passed through the object detector and feature extractor only once, but the features are used multiple times for computing affinities with multiple other frames in pairs.

3.4 Deep track association

To associate objects in the current frame with multiple previous frames, we store feature matrices of the frames with their time stamps. After frame 0, that initializes our approach and results in $\boldsymbol{F}_0$, we can compute affinities between the objects in the current frame and those in any previous frame using their feature matrices. As illustrated in Fig. 3, the computed affinity matrices are used to update trajectory sets by looking back deep into the previous frames.

Deep track association is performed as follows. We initialize our track set $\boldsymbol{\mathcal{T}}_0$ with as many trajectories as the number of detected objects in $\boldsymbol{I}_0$. A trajectory here is a set of 2-tuples, each containing the time stamp of the frame and the object identity. The trajectory set is updated at the t^{th} time stamp with the help of Hungarian algorithm [27] applied to an accumulator matrix $\boldsymbol{\Lambda}_t \in \mathbb{R}^{\mathcal{Z}(\boldsymbol{\mathcal{T}}_{t-1}) \times (\mathcal{Z}(\boldsymbol{\mathcal{C}}_t)+1)}$. The Hungarian algorithm solves an assignment problem using combinatorial optimization. We formulate this problem using the accumulator matrix, whose rows represent the existing object identities and columns encode their similarities to the objects in the current frame. The Hungarian algorithm computes unique assignments of the objects to the identities. This is done by maximizing the affinities between the current frame objects and the objects already assigned to the identities in previous frames. The used accumulator matrix integrates the affinities between objects in the current frame and the previous frames. A coefficient of $\boldsymbol{\Lambda}_t$ at index (i, j) is the sum of affinities of the i^{th} identity in the track set $\boldsymbol{\mathcal{T}}_{t-1}$ to the j^{th} object in the t^{th} frame for the previous δ_b frames, where δ_b is a parameter of our approach.

At each time stamp, we are able to efficiently compute up to δ_b affinity matrices using the DAN to look back into the existing tracks. We let $\delta_b = t$ in Fig. 3 for simplicity. One subtle issue in successfully applying the Hungarian algorithm to our problem is that we allow multiple objects to leave a video between its frames. Therefore, multiple trajectories could be assigned to the *single* Un-Identified target column in our accumulator matrix (inherited from the affinity matrices). We handle this issue by repeating the last column of $\boldsymbol{\Lambda}_t$ until every trajectory in $\boldsymbol{\mathcal{T}}_{t-1}$ gets assigned to a unique column of the augmented matrix[1]. This ensures that all un-identified trajectories can be mapped to un-identified objects.

Overall, our tracker is an on-line approach in the sense that it does not use future frames to predict object trajectories. Hence, it can be used with continuous video streams. One practical issue in such cases is that very long tacks may result over large time intervals. The parameter δ_b also bounds the maximum number of time stamps we associate with a track. We remove the oldest node in a track if the number of frames for its trajectory exceeds δ_b. Similarly, for the objects disappearing from a video, we allow a waiting time of δ_w frames before removing its trajectory from the current track set. We introduce these parameters in our framework for purely pragmatic reasons. Their values can be adjusted according to the on-board computational and memory capacity of a tracker.

4 Experiments

In this section, we evaluate the proposed approach on three well-known multiple object tracking challenges, namely

1. The number of columns of *augmented* $\boldsymbol{\Lambda}_t$ are allowed to exceed $N_m + 1$. However, it does not have any ramifications as that matrix is only utilized by the Hungarian algorithm.

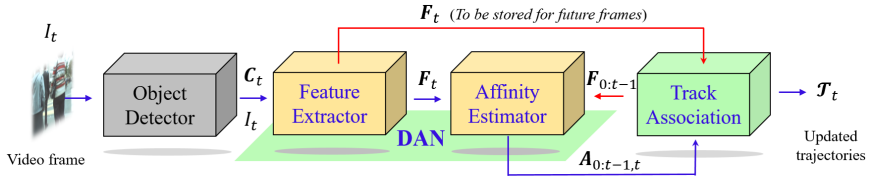

Fig. 3: Deep tracking with DAN deployment: For the t^{th} frame $\boldsymbol{I}_t$, the object centers $\mathcal{C}_t$ provided by the detectors are used to compute the feature matrix $\boldsymbol{F}_t$ with *one* stream Feature Extractor of DAN. The $\boldsymbol{F}_t$ is paired with each of the last 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.

Multiple Object Tracking 17 (MOT17) [66], Multiple Object Tracking 15 (MOT15) [28] and UA-DETRAC [30], [31]. All these are on-line challenges where tracking results are computed by a hosting server once a new technique is submitted for evaluation. Annotated training data is provided for learning the models, however, labels of the test data remain undisclosed. The servers perform comprehensive evaluation of the submitted techniques using several standard metrics. We first report the implementation details of the proposed technique, followed by the on-line challenges; their training datasets and the used evaluation metrics. The performance of our approach and its comparison to other techniques currently on the leader board is also presented.

4.1 Implementation details

We implement the Deep Affinity Network (DAN) using the Pytorch framework [76]. Training is performed on NVIDIA GeForce GTX Titan GPU. Hyper-parameters of DAN are optimized with the help of MOT17 dataset, for which we specify a validation set to train our model. MOT17 is selected for parameter optimization due to its manageable size. The hyper-parameter values finally used in our implementation are as follows. Batch size $B = 8$, number of training epochs per model = 120, number of maximum objects allowed per frame $N_m = 80$, and $\gamma = 10$. We let $N_V = 30$. Our network has an input frame size of 900×900. All the training and testing data is first resized to these dimensions before passing it through the network. We use the SGD optimizer [77] for training the DAN, for which we respectively use 0.9 and 5e-4 for the momentum and weight decay parameters. We start the learning process with 0.01 as the learning rate, which is decreased to $1/10^{\text{th}}$ of the previous value at epochs 50, 80 and 100. Once the network is trained, we still need to decide on the values of the parameters δ_b and δ_w. We select the best values of these parameters with a grid search for optimal MOTA metric on our validation set. We use multiples of three in the range [3,30] to form the grid, based on which $\delta_w = 12$ and $\delta_b = 15$ are selected in the final implementation.

4.2 Multiple Object Tracking 17 (MOT17)

The Multiple Object Tracking 17 (MOT17) [66] is among the latest on-line challenges in tracking. Similar to its previous version MOT16 [29], this challenge contains seven different indoor and outdoor scenes of public places with pedestrians as the objects of interest. A video for each scene is divided into two clips, one for training and the other for testing. The dataset provides detections of objects in the video frames with three detectors, namely SDP [69], Faster-RCNN [2] and DPM [47]. The challenge accepts both on-line and off-line tracking approaches, where the latter are allowed to use the future video frames to predict tracks.

4.2.1 Dataset

The challenge provides seven videos for training along with their ground truth tracks and detected boxes from three detectors. The scenes vary significantly in terms of background, illumination conditions and camera view points. We summarize the main attributes of the provided training data in Table 4. As can be noticed, the provided resolution for scene 05 is different from the others. Similarly, there are also variations in camera frame rates, the average number of objects per frame (i.e. density) and the total number of tracks. Scene 07 is also captured with a lower angle that resulted in significant occlusions. All these variations make the dataset challenging.

The testing data consists of clips from the same seven scenes, excluding the training video clips. There are $17,757$ frames in testing data, and it is public knowledge that there are $2,355$ tracks in those clips with $564,228$ boxes. However, the track labels and boxes are not available publicly, and evaluation is performed by the host server in private.

4.2.2 Evaluation metrics

We benchmark our approach comprehensively using twelve standard evaluation metrics, that include both CLEAR MOT metrics [78], and the MT/ML metrics [79]. We summarize these metrics in Table 5. Definitions of MOTA and MOTAL are provided below:

$$\text{MOTA} = 1 - \frac{\sum_t (\text{FP}_t + \text{FN}_t + \text{ID\_Sw}_t)}{\sum_t \text{GT}_t}. \quad (6)$$

$$\text{MOTAL} = 1 - \frac{\sum_t (\text{FP}_t + \text{FN}_t + \log_{10}(\text{ID\_Sw}_t + 1))}{\sum_t \text{GT}_t}. \quad (7)$$

In the above equations, the subscript 't' indicates that the values are computed at the t^{th} time stamp, whereas 'GT' stands for ground truth. We refer to Table 5 for the other symbols used in the equations.

4.2.3 Results

We train our DAN with the training data provided by the hosting server. Upon submission, our approach was

TABLE 4: Attributes of MOT17 training data [29]. The last three columns indicate the number of detected boxes by the detectors in complete video. 'Density' denotes the average number of pedestrians per frame, and 'Move' indicates if the video is recorded by a moving camera (Y) or not (N).

Video Index	Resolution	FPS	Length (frames)	Boxes	Tracks	Density	Move	DPM [47]	SDP [69]	FRCNN [2]
02	1920 × 1080	30	600	18581	62	31.0	N	7267	11639	8186
04	1920 × 1080	30	1050	47557	83	45.3	N	39437	37150	28406
05	640 × 480	14	837	6917	133	8.3	Y	4333	4767	3848
09	1920 × 1080	30	525	5325	26	10.1	N	5976	3607	3049
10	1920 × 1080	30	654	12839	57	19.6	Y	8832	9701	10371
11	1920 × 1080	30	900	9436	75	15.5	Y	8590	7509	6007
13	1920 × 1080	25	750	11642	110	8.3	Y	5355	7744	8442

TABLE 5: Metrics used for benchmarking.

Metric	Better	Perfect	Description
MOTA	higher	100%	Overall Tracking Accuracy. See Eq. (6).
MOTAL	higher	100%	Log Tracking Accuracy. See Eq. (7).
MOTP	higher	100%	Percentage alignment of predicted bounding box and ground truth.
Rcll	higher	100%	The percentage of detected targets.
IDF1	higher	100%	F1 score of the predicted identities.
MT	higher	100%	Mostly tracked targets. The percentage of ground-truth trajectories covered by a track hypothesis for 80% of their life or more.
ML	lower	0	Mostly lost targets. The percentage of ground-truth trajectories covered by a track hypothesis for 20% of their life or less.
FP	lower	0	Number of false positives.
FN	lower	0	Number of false negatives.
ID_Sw.	lower	0	Identity switches, see [80] for details.
Frag	lower	0	The count of trajectory fragmentations.
Hz	higher	Inf.	Processing speed in frames per second.

benchmarked by the server itself. In Table 6, we summarize current results of the published techniques on the leader board taken directly from the challenge server. Whereas our method is *on-line*, the table also includes results of the best performing *off-line* methods to highlight competitive performance of our approach. As can be seen, the proposed approach (named DAN after the network) is able to outperform the existing on-line and off-line methods on five metrics, whereas the performance generally remains competitive on the other metrics. In particular, our results are significantly better than the existing on-line methods for MOTA and MOTAL that are widely accepted as comprehensive multiple object tracking metrics.

In Fig. 4, we show two examples of our method's tracking performance on MOT17 challenge. The color of bounding boxes in the shown frames indicate the trajectory identity predicted by our tracker. The numbers mentioned on the frames are for reference in the text only. The figure presents typical examples of inter-frame occlusions occurring in tracking datasets. In scene 06 (first row), identity-1 disappears in frame 633 (and adjacent frames - not shown), and then reappears in frame 643. Our tracker is able to easily recover from this occlusion (the same color of bounding boxes). Similarly, the occlusion of identity-1 in frame 144 of scene 07 is also handled well by our approach. Our tracker temporarily misjudges the trajectory in frame 141 by assigning it a wrong identity, i.e. 4 due to severe partial occlusion. However, it is able to quickly recover from this situation by looking deeper into the previous frames.

4.3 Multiple Object Tracking 15 (MOT15)

To comprehensively benchmark our technique for pedestrian tracking, we also evaluate it on Multiple Object Tracking 15 (MOT15) challenge [28] that deals with multiple pedestrian tracking similar to MOT17. However, due to its earlier release in 2014, MOT15 benchmarks more methods as compared to MOT17. MOT15 provides 11 video sequences along with their ground truth tracks and detections using the detector proposed by Dollar et al. [70]. The provided ground truth detections do not consider object occlusions and a bounding box for a completely occluded object of interest is provided for training anyway. Such training data can be misleading for our appearance modeling based tracker. Hence, we did not train or fine-tune the DAN with MOT15 training data. Instead, we directly applied our MOT17 model to the MOT15 challenge. The hosting server computed the results of our method using its own detector [70].

4.3.1 Results

There are 4 video sequences (Venice-1, ADL-Rundle-1, ADL-Rundle-3 and ETH-Crossing) in MOT15 that contain the same scenes as the video sequences 01, 06, 07 and 08 in the MOT17. For a fair comparison of our technique with the benchmarked approaches, we report the average results on these scenes in Table 7. All of the approaches in the table contain the training video clips of the same four scenes in their training data. As can be seen, the proposed approach significantly outperforms the existing methods evaluated on the MOT15 challenge on three metrics, especially on the MOTA metric.

One particularly noticeable method in Table 7 is CDA-DDAL [18] that also uses deep features for appearance modeling, but computes appearance affinities using ℓ_2-distance. Along appearance affinities, it additionally uses shape and motion affinities with the Hungarian algorithm for tracklet association. Despite the use of additional affinities under a related pipeline, CDA-DDAL significantly underperforms as compared to our tracker. Our technique mainly achieves the performance gain due to effective simultaneous appearance modeling and affinity prediction with DAN. We emphasize that our tracker only uses appearance affinities predicted by DAN.

4.4 UA-DETRAC

The UA-DETRAC challenge [30], [31] is based on a large-scale tracking dataset for vehicles. It comprises 100 videos that record around 10 hours of vehicle traffic. The recording is made in 24 different locations, and it includes a wide variety of common vehicle types and traffic conditions. The

TABLE 6: MOT17 challenge results from the server: The symbol ↑ indicates that higher values are better, and ↓ implies lower values are favored. The proposed method is *online*. Offline methods are included for reference only.

Tracker	Type	MOTA↑	MOTAL↑	MOTP↑	Rcll↑	IDF1↑	MT↑	ML↓	FP↓	FN↓	ID_Sw↓	Frag↓	Hz↑
FWT_17 [9]	offline	51.3173	51.786	77.0024	56.0583	47.5597	21.4	35.3	24101	247921	2648	4279	0.2
jCC [81]	offline	51.1614	51.4802	75.9164	56.0777	**54.4957**	20.9	37.0	25937	247822	**1802**	2984	1.8
MHT_DAM [10]	offline	50.7132	51.1228	**77.5219**	55.1777	47.1803	20.8	36.9	22875	252889	2314	**2865**	0.9
EDMT17 [56]	offline	50.0464	50.4471	77.2553	56.1689	51.2532	**21.6**	36.3	32279	247297	2264	3260	0.6
MHT_bLSTM [82]	offline	47.5226	47.8887	77.4943	52.494	51.9188	18.2	41.7	25981	268042	2069	3124	1.9
IOU17 [55]	offline	45.4765	46.5371	76.8505	50.0814	39.4037	15.7	40.5	**19993**	281643	5988	7404	**1522.9**
PHD_GSDL17 [16]	*online*	48.0439	48.7518	77.1522	52.8641	49.6294	17.1	35.6	23199	265954	3998	8886	6.7
EAMTT [83]	*online*	42.6344	43.4291	76.0305	48.8728	41.7654	12.7	42.7	30711	288474	4488	5720	1.4
GMPHD_KCF [17]	*online*	39.5737	40.603	74.5414	49.6253	36.6362	8.8	43.3	50903	284228	5811	7414	3.3
GM_PHD [84]	*online*	36.3560	37.1719	76.1957	41.3771	33.9243	4.1	57.3	23723	330767	4607	11317	38.4
DAN (Proposed)	*online*	**52.4224**	**53.916**	76.9071	**58.4225**	49.4934	21.4	**30.7**	25423	**234592**	8431	14797	6.3

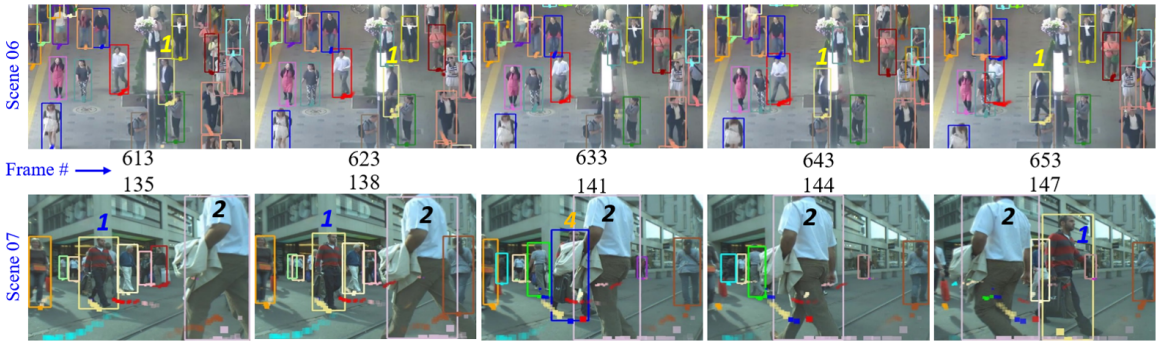

Fig. 4: Tracking example of the proposed method from MOT17 (taken from the host server). The predicted tracks are identified by the color of bounding boxes. The mentioned identity numbers are for reference in the text only. In both scenes, our approach successfully tracks identity-1 despite inter-frame occlusions. Frame 141 of Scene 07 causes a temporary mis-identification, however, our approach is able to recover well due to deep track association.

TABLE 7: MOT15 challenge results: The symbols ↑ and ↓ respectively indicate that higher and lower values are preferred.

Tracker	Type	MOTA ↑	MOTP ↑	IDF1 ↑	MT ↑	ML ↓	FP ↓	FN ↓	ID_Sw ↓	Frag ↓	HZ ↑
QuadMOT [58]	offline	29.30	**75.73**	40.88	10.48	34.28	1022.25	3461.00	1670.97	2732.67	6.2
CNNTCM [85]	offline	23.03	73.90	34.90	7.93	41.88	1167.25	3607.50	2792.58	2606.37	1.7
SiameseCNN [86]	offline	28.78	72.90	39.70	10.83	42.20	633.00	3681.50	846.93	2020.80	52.8
MHT_DAM [10]	offline	27.58	73.53	44.50	**21.30**	34.60	1290.25	3223.75	1453.52	2170.06	0.7
LP_SSVM [87]	offline	21.18	73.35	36.15	8.70	37.80	1517.00	3396.00	2002.88	2342.29	41.3
AMIR15 [88]	*online*	31.88	73.40	44.13	11.73	**22.53**	1105.75	3055.75	3273.27	6360.93	1.9
HybridDAT [62]	*online*	31.48	75.03	46.90	19.70	27.38	1299.00	2869.00	1373.40	3674.18	4.6
AM [89]	*online*	30.23	72.20	**46.98**	12.90	46.75	510.50	3711.00	**755.37**	4133.45	0.5
SCEA [90]	*online*	23.85	73.08	32.65	8.03	45.78	751.50	3847.00	2220.33	3157.50	6.8
RNN_LSTM [91]	*online*	13.83	72.28	20.78	6.40	37.58	1779.75	3615.50	5287.59	7520.59	**165.2**
CDA-DDAL [18]	*online*	32.80	70.70	38.70	9.70	42.20	4983.00	35690.00	**614.00**	1583	1.20
DAN (Proposed)	*online*	**38.30**	71.10	45.60	17.60	41.20	1290.25	**2700.00**	1648.08	**1515.60**	6.3

scenes include urban highways, traffic crossings, and T-junctions etc. Overall, the dataset contains about 140k video frames, 8,250 vehicles, and 1,210k bounding boxes. Similar to the MOT challenges, the UA-DETRAC challenge accepts submissions of tracking approaches and the host server evaluates their performance on a separate test data.

4.4.1 Dataset

We summarize the main attributes of the dataset in Table 8. The table contains information on both training and testing sets. The available videos have a consistent frame size of 540×960, and a frame rate of 25 fps. All the videos are recorded with static cameras, generally installed at high locations. Although in terms of variations this dataset may appear less challenging than MOT datasets, the larger size of data and the scenes of crossings and junctions make tracking in this dataset a difficult task.

TABLE 8: Attributes of UA-DETRAC dataset [30], [31]. The 'Boxes' column indicates the total number of bounding boxes in videos. 'Length' is given in number of frames.

Type	Videos	Length	Boxes	Vehicles	Tracks	Density
Training	60	84k	578k	5936	5.9k	6.88
Testing	40	56k	632k	2314	2.3k	11.29

4.4.2 Evaluation metrics

The evaluation metrics used by UA-DETRAC are similar to those introduced in Table 5, however, they are computed slightly differently using Precision-Recall curve, which is indicated with the prefix 'PR' in the table. As an example, to compute PR-MOTA, the thresholds of the detectors are gradually varied to compute a 2D precision-recall curve. Then, for each point on the plot, MOTA value is estimated to get a 3D curve. The PR-MOTA is computed as the integral

TABLE 9: UA-DETRAC challenge results: The symbol ↑ indicates that higher values are better, and ↓ implies lower values are favored. The names of approaches also include the used detectors.

Name	PR-MOTA ↑	PR-MOTP ↑	PR-MT ↑	PR-ML ↓	PR-FP ↓	PR-FN ↓	PR-ID_Sw ↓	Hz ↑
EB [71]+IOUT [55]	19.4	28.9	**17.7**	18.4	14796.5	171806.8	2311.3	**6902.1**
R-CNN [92]+IOUT [55]	16.0	**38.3**	13.8	20.7	22535.1	193041.9	5029.4	-
CompACT [93]+GOG [42]	14.2	37.0	13.9	19.9	32092.9	180183.8	3334.6	389.5
CompACT [93]+CMOT [13]	12.6	36.1	16.1	18.6	57885.9	167110.8	**285.3**	3.8
CompACT [93]+H2T [19]	12.4	35.7	14.8	19.4	51765.7	173899.8	852.2	3.0
RCNN [92]+DCT [94]	11.7	38.0	10.1	22.8	336561.2	210855.6	758.7	0.7
CompACT [93]+IHTLS [95]	11.1	36.8	13.8	19.9	53922.3	180422.3	953.6	19.8
CompACT [93]+CEM [96]	5.1	35.2	3.0	35.3	12341.2	260390.4	267.9	4.6
Proposed (EB [71]+**DAN**)	**20.2**	26.3	14.5	**18.1**	**9747.8**	**135978.1**	518.2	6.3

score of the resulting curve. The same procedure is also adopted for computing the other metrics.

4.4.3 Results

We report the quantitative results of our approach in the last row of Table 9. The table also summarizes results of other top approaches on the leader board for UA-DETRAC challenge at the time of submission of this work. The method names (first column) include the used detectors, e.g. we used the EB detector [71], hence EB+DAN. We note that for every technique in the Table (including ours), the choice of detector is empirical. We also tested our tracker with CompACT and RCNN detectors. The values of PR-MOTA and PR-MOTAP metrics for CompACT+DAN are 18.6 and 35.8, respectively. For RCNN+DAN, these values are 15.1 and 37.1. The overall performance of our tracker improves with the accuracy of the detector. In the context of detection-based tracking, this implicates an accurate tracking component of the overall technique. These results ascertain the overall effectiveness of the proposed approach in tracking vehicles on roads.

We also illustrate tracking results of our approach in Fig. 5 with the help of two examples. Again, the predicted trajectories are specified by the colors of bounding boxes, and the mentioned identity numbers are only for referencing in the text. In the top row (scene MVI_40762) the EB detector fails to detect identity-2 in frame 130 - 134 (only frame 132 is shown) due to occlusion by identity-1. However once the detection is made again, our tracker is able to assign the object correctly to its trajectory from frame 136 onward. Similarly, in scene MVI_40855, the detector is unable to detect identity-1 in frames 132-143 due to occlusion by identity-2. Nevertheless, when the identity-1 is detected again in frame 154, our approach assigns it to its correct trajectory. These examples demonstrate robustness of our approach to missed detections under tracking-by-detection framework.

5 Discussion

The strength of our approach comes from comprehensive appearance modeling and effective affinity computation, such that the latter is fully tailored to the former under end-to-end training of DAN. DAN is the first deep network that models objects' appearance and computes inter-frame object affinities *simultaneously*. This unique property of our network also sets its performance apart form other methods. In Fig. 6, we illustrate the object association abilities of DAN under practical conditions. Each column of the figure shows a pair of frames that are n time stamps apart, where n is randomly sampled from [1, 30]. The figure shows association between the objects as computed by DAN. Firstly, it can be noticed that the association is robust to large illumination variations. Secondly, DAN is able to comfortably handle significant object occlusions. Additionally, despite the existence of multiple similar looking objects, the network is able to correctly associate the objects. For instance, see Fig. 6(e) where there are multiple white and red cars but this does not cause any problem. The chosen examples in Fig. 6 are random. We observed similar level of performance by DAN for all the cases we tested.

To evaluate the accuracy of the affinity matrix predicted by DAN, we compare it to more specialized methods for affinity computation, DCML [97] and SSIM [98]. We compute affinities between two frames that are 'n' time stamps apart in MOT17 dataset, where $n = 1, 5, 10, 15, 20, 25, 30$. The predicted affinity matrices are then used for data association. In Fig. 7, we plot the (element-wise) mean absolute error between the predicted and the ground truth association matrices for all possible frame pairs of MOT17 for each value of 'n'. The consistent low error of DAN demonstrates the advantage of using a deep network with specialized loss for affinity prediction.

The only noticeable situation where DAN underperformed in terms of data association was when the frames contained similar looking objects at very close locations in the scene at multiple time stamps. This sometimes resulted in ID switches between those objects. However, due to the Deep Track Association (Section 3.4), the proposed method was able to recover well from such scenarios. Since our technique is essentially an appearance modeling based tracker, it can be expected to underperform in conditions that are inherently unsuitable for this tracking paradigm, e.g. highly crowded scenes with multiple similar looking objects. Nevertheless, this issue is associated with all appearance based trackers.

For an effective architecture of our network, we tested numerous intuitive alternatives. We present a few interesting choices out of those as an ablation analysis here. We choose these cases for their ability to add to our understanding the role of important components in the final network. We introduce DAN-Remove model, that removes the 'Feature dimension reduction' layers in Fig. 1 and directly concatenates the features to estimate object affinities. As another case, we replace the 'Compression network' in affinity estimator component by a single convolutional layer. Hence, the resulting DAN-Replace model maps object features to affinities abruptly instead of doing it gradually. In another choice, instead of defining our ensemble loss with the *max*

Fig. 5: Tracking examples from the UA-DETRAC challenge. Predicted tracks are identified by the bounding box colors and the identity numbers are for reference only. The proposed tracker is able to assign identities to their correct track (in both cases) despite missed detection in several frames due to limited performance of the detector for occluded objects.

Fig. 6: Illustration of cross-frame associations based on DAN outputs. The frame pairs are randomly chosen $n \sim [1, 30]$ time-stamps apart. The association remains robust to illumination conditions, partial occlusions and existence of multiple similar looking objects in the video frames.

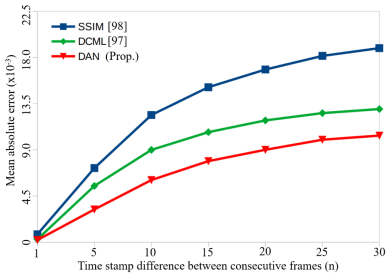

Fig. 7: Mean absolute error of predicted data association on MOT17. All possible frame pairs are used for each n - the time stamp difference between the two frames in a pair.

operation, we use the *mean* operation, and refer to the resulting network as DAN-Mean. We also introduce DAN-Curtail that removes the 'Extension' sub-network from the 'Feature extractor' component of the overall network.

In Table 10, we provide the MOT17 training loss values at different epochs for the different variants of DAN as well as the MOTA and IDF1 scores at 120 epochs. The MOTA and IDF1 scores are different here from Table 5 where the scores on test data from the challenge server are listed. From the table, it is apparent that the proposed DAN is able to achieve better results in fewer epochs compared to all the other choices. The closest performance is achieved by DAN-Mean that has the same architecture as DAN but a slightly different loss function. The importance of feature compression is clear from the poor performance of DAN-Remove. Similarly, from the results of DAN-Replace, it is also apparent that gradual compression of feature maps for affinity estimation is more desirable than an abrupt compression. Finally, the contribution of the Extension sub-network in the overall model is evident from the significant drop in performance of DAN-Curtail.

TABLE 10: Loss values of DAN variants at 50-120 epochs on MOT17 while training. MOTA and IDF1 scores are computed at 120 epochs.

Variant	50	80	100	110	120	MOTA	IDF1
DAN-Replace	0.155	0.115	0.108	0.112	0.111	52.1	48.5
DAN-Remove	0.209	0.181	0.169	0.131	0.124	51.7	46.2
DAN-Mean	0.134	0.083	0.080	0.076	0.075	53.4	60.7
DAN-Curtail	0.367	0.271	0.233	0.212	0.209	45.2	43.7
DAN	**0.107**	**0.057**	**0.045**	**0.045**	**0.043**	**53.5**	**62.3**

We report the average runtime performance of the major components of our tracking technique in Table 11. Each row of the table shows different number of objects to be tracked, i.e. N_m. The timings are for MOT17 validation set used in our experiments, computed with local NVIDIA GeForce GTX Titan GPU. Using the same GPU, the average time taken by the proposed DAN to process a single pair of frames during training is 188.54 ms, that translates to 1508.33 ms for a mini-batch of 8 samples. DAN is the only trainable component in our technique, and it is trained in an end-to-end manner. From the table, we can see that a 4 times increase in N_m only results in a 1.2 times increase in the overall runtime of our technique.

TABLE 11: Average runtime (ms) for the major components of our technique for MOT17 validation set. N_m denotes the maximum number of allowed objects in a frame.

N_m	Feature Extractor	Affinity Estimator	Track Association
20	2.23	10.55	0.14
40	2.29	10.90	0.17
60	2.46	11.52	0.18
80	2.50	12.83	0.19

6 Conclusion

We presented a multiple object tracker that performs on-line tracking by associating the objects detected in the current frame with those in multiple previous frames. The tracker derives its strength from our proposed Convolutional Neural Network architecture, referred to as Deep Affinity Network (DAN). The proposed DAN models features of pre-detected objects in the video frames at multiple levels of abstraction, and infers object affinities across different frames by analyzing exhaustive permutations of the extracted features. The cross-frame objects similarities and object features are recorded by our approach to trace the trajectories of the object. We evaluated our approach on three on-line multiple object tracking challenges MOT17, MOT15 and UA-DETRAC, using twelve different evaluation metrics. The proposed tracker is able to achieve excellent overall performance with the highest Multiple Object Tracking Accuracy on all the challenges. It also achieved an average speed of 6.3 frames per second, while leading the result board as the best performing on-line tracking approach on many of the evaluation metrics.

7 Acknowledgement

This research was supported by ARC grant DP160101458 and DP190102443, the National Natural Science Foundation of China (Grant No. 61572083), the Joint Found of of Ministry of Education of China (Grant No. 6141A02022610), Key projects of key R & D projects in Shaanxi (Grant No. 2018ZDXM-GY-047), Team cultivation project of Central University (Grant No. 300102248402) and China Scholarship Council. Mubarak Shah acknowledges the support of ODNI, and IARPA, via IARPA R&D Contract No. D17PC00345. The views, findings, opinions, and conclusions or recommendations contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purpose not withstanding any copyright annotation thereon. The GPU used for this work was donated by NVIDIA Corporation.

References

[1] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, X. Zhao, and T.-K. Kim, "Multiple Object Tracking: A Literature Review," *arXiv:1409.7618v4*, pp. 1–18, 2017.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE TPAMI*, vol. 39, no. 6, pp. 1137–1149, 2017.

[3] D. Impiombato, S. Giarrusso, T. Mineo, O. Catalano, C. Gargano, G. La Rosa, F. Russo, G. Sottile, S. Billotta, G. Bonanno, S. Garozzo, A. Grillo, D. Marano, and G. Romeo, "You Only Look Once: Unified, Real-Time Object Detection Joseph," *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 794, pp. 185–192, 2015.

[4] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," 2016.

[5] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *Proc. CVPR*, 2008.

[6] W. Hu, X. Li, W. Luo, X. Zhang, S. Maybank, and Z. Zhang, "Single and multiple object tracking using log-euclidean riemannian subspace and block-division appearance model," *IEEE TPAMI*, vol. 34, no. 12, pp. 2420–2440, 2012.

[7] L. Zhang and L. Van Der Maaten, "Preserving structure in model-free tracking," *IEEE TPAMI*, vol. 36, no. 4, pp. 756–769, 2014.

[8] Y. Xiang, A. Alahi, and S. Savarese, "Learning to track: Online multi-object tracking by decision making," in *Proc. IEEE CVPR*, vol. 2015 Inter, 2015, pp. 4705–4713.

[9] R. Henschel, L. Leal-Taixé, D. Cremers, and B. Rosenhahn, "Fusion of Head and Full-Body Detectors for Multi-Object Tracking," *Proc. CVPRW*, 2018.

[10] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg, "Multiple hypothesis tracking revisited," in *Proc. ICCV*, vol. 2015 Inter, 2015, pp. 4696–4704.

[11] S. Zhang, X. Lan, H. Yao, H. Zhou, D. Tao, and X. Li, "A Biologically Inspired Appearance Model for Robust Visual Tracking," *IEEE Transactions on Neural Networks and Learning Systems*, 2016.

[12] H. Nam and B. Han, "Learning Multi-Domain Convolutional Neural Networks for Visual Tracking," *Cvpr*, pp. 4293–4302, 2015.

[13] S. H. Bae and K. J. Yoon, "Robust online multi-object tracking based on tracklet confidence and online discriminative appearance learning," *Proc. CVPR*, pp. 1218–1225, 2014.

[14] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, "Robust tracking-by-detection using a detector confidence particle filter," *Proc. ICCV*, pp. 1515–1522, 2009.

[15] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua, "Multicamera people tracking with a probabilistic occupancy map," *IEEE TPAMI*, vol. 30, no. 2, pp. 267–282, 2008.

[16] Z. Fu, P. Feng, S. M. Naqvi, and J. A. Chambers, "Particle PHD filter based multi-target tracking using discriminative group-structured dictionary learning," in *ICASSP*, 2017, pp. 4376–4380.

[17] T. Kutschbach, E. Bochinski, V. Eiselein, and T. Sikora, "Sequential sensor fusion combining probability hypothesis density and kernelized correlation filters for multi-object tracking in video data," in *Proc. AVSS*, 2017.

[18] S. H. Bae and K. J. Yoon, "Confidence-Based Data Association and Discriminative Deep Appearance Learning for Robust Online Multi-Object Tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 595–610, 2018.

[19] L. Wen, W. Li, J. Yan, Z. Lei, D. Yi, and S. Z. Li, "Multiple target tracking based on undirected hierarchical relation hypergraph," *Proc. CVPR*, vol. 1, pp. 1282–1289, 2014.

[20] C. H. Kuo, C. Huang, and R. Nevatia, "Multi-target tracking by on-line learned discriminative appearance models," in *Proc. CVPR*, 2010, pp. 685–692.

[21] H. Izadinia, I. Saleemi, W. Li, and M. Shah, "(MP)2T: Multiple people multiple parts tracker," in *Lecture Notes in Computer Science*, vol. 7577 LNCS, no. PART 6, 2012, pp. 100–114.

[22] K. Yamaguchi, A. C. Berg, L. E. Ortiz, and T. L. Berg, "Who are you with and where are you going?" in *Cvpr 2011*, 2011, pp. 1345–1352.

[23] K. Shafique, W. L. Mun, and N. Haering, "A rank constrained continuous formulation of multi-frame multi-target tracking problem," in *Proc. CVPR*, 2008.

[24] Q. Yu, G. Medioni, and I. Cohen, "Multiple Target Tracking Using Spatio-Temporal Markov Chain Monte Carlo Data Association," in *Proc. CVPR*, 2007, pp. 1–8.

[25] R. Nevatia, "Multi-target tracking by online learning of non-linear motion patterns and robust appearance models," in *Proc. CVPR*. IEEE, 2012, pp. 1918–1925.

[26] G. Ning, Z. Zhang, C. Huang, Z. He, X. Ren, and H. Wang, "Spatially Supervised Recurrent Convolutional Neural Networks for Visual Object Tracking," 2016.

[27] J. Munkres, "Algorithms for the Assignment and Transportation Problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[28] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler, "MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking," pp. 1–15, 2015.

[29] A. Milan, L. Leal-Taixé, I. D. Reid, S. Roth, and K. Schindler, "MOT16: A Benchmark for Multi-Object Tracking," *CoRR*, vol. abs/1603.0, 2016.

[30] L. Wen, D. Du, Z. Cai, Z. Lei, M.-C. Chang, H. Qi, J. Lim, M.-H. Yang, and S. Lyu, "UA-DETRAC: A New Benchmark and Protocol for Multi-Object Detection and Tracking," 2015.

[31] S. Lyu, M. C. Chang, D. Du, L. Wen, H. Qi, Y. Li, Y. Wei, L. Ke, T. Hu, M. Del Coco, P. Carcagni, D. Anisimov, E. Bochinski, F. Galasso, F. Bunyak, G. Han, H. Ye, H. Wang, K. Palaniappan, K. Ozcan, L. Wang, L. Wang, M. Lauer, N. Watcharapinchai, N. Song, N. M. Al-Shakarji, S. Wang, S. Amin, S. Rujikietgumjorn, T. Khanova, T. Sikora, T. Kutschbach, V. Eiselein, W. Tian, X. Xue, X. Yu, Y. Lu, Y. Zheng, Y. Huang, and Y. Zhang, "UA-DETRAC 2017: Report of AVSS2017 & IWT4S Challenge on Advanced Traffic Monitoring," in *Proc. IEEE AVSS*, 2017.

[32] P. Emami, P. M. Pardalos, L. Elefteriadou, and S. Ranka, "Machine Learning Methods for Solving Assignment Problems in Multi-Target Tracking," vol. 1, no. 1, pp. 1–35, 2018.

[33] Y. Tian, A. Dehghan, and M. Shah, "On detection, data association and segmentation for multi-target tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2018.

[34] L. Wen, D. Du, S. Li, X. Bian, and S. Lyu, "Learning non-uniform hypergraph for multi-object tracking," *arXiv preprint arXiv:1812.03621*, 2018.

[35] H. Sheng, Y. Zhang, J. Chen, Z. Xiong, and J. Zhang, "Heterogeneous association graph fusion for target association in multiple object tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.

[36] K. Shafique and M. Shah, "A noniterative greedy algorithm for multiframe point correspondence," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 1, pp. 51–65, 2005.

[37] D. Reid *et al.*, "An algorithm for tracking multiple targets," *IEEE transactions on Automatic Control*, vol. 24, no. 6, pp. 843–854, 1979.

[38] G. Shu, A. Dehghan, O. Oreifej, E. Hand, and M. Shah, "Part-based multiple-person tracking with partial occlusion handling," in *Proc. CVPR*. IEEE, 2012, pp. 1815–1821.

[39] A. Roshan Zamir, A. Dehghan, and M. Shah, "GMCP-tracker: Global multi-object tracking using generalized minimum clique graphs," in *Lecture Notes in Computer Science*, 2012.

[40] B. Wu and R. Nevatia, "Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors," *International Journal of Computer Vision*, vol. 75, no. 2, pp. 247–266, 2007.

[41] A. Dehghan, S. Modiri Assari, and M. Shah, "Gmmcp tracker: Globally optimal generalized maximum multi clique problem for multiple object tracking," in *Proc. CVPR*, 2015, pp. 4091–4099.

[42] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes, "Globally-optimal greedy algorithms for tracking a variable number of objects," *Proc. CVPR*, pp. 1201–1208, 2011.

[43] A. A. Butt and R. T. Collins, "Multi-target tracking by lagrangian relaxation to min-cost network flow," in *Proc. CVPR*, 2013, pp. 1846–1853.

[44] J. Berclaz, F. Fleuret, E. Türetken, and P. Fua, "Multiple object tracking using k-shortest paths optimization," *IEEE TPAMI*, vol. 33, no. 9, pp. 1806–1819, 2011.

[45] H. B. Shitrit, J. Berclaz, F. Fleuret, and P. Fua, "Multi-commodity network flow for tracking multiple people," *IEEE TPAMI*, vol. 36, no. 8, pp. 1614–1627, 2014.

[46] V. Chari, S. Lacoste-Julien, I. Laptev, and J. Sivic, "On pairwise costs for network flow multi-object tracking," *Proc. CVPR*, vol. 07-12-June, pp. 5537–5545, 2015.

[47] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE TPAMI*, vol. 32, no. 9, pp. 1627–1645, 2010.

[48] B. Leibe, K. Schindler, N. Cornelis, and L. Van Gool, "Coupled object detection and tracking from static cameras and moving vehicles," *IEEE TPAMI*, vol. 30, no. 10, pp. 1683–1698, 2008.

[49] S. Tang, B. Andres, M. Andriluka, and B. Schiele, "Subgraph decomposition for multi-target tracking," in *Proc. CVPR*, 2015, pp. 5033–5041.

[50] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-Learning-Detection." *IEEE TPAMI*, vol. 34, no. 7, pp. 1409–1422, 2011.

[51] S. Hare, A. Saffari, and P. H. Torr, "Struck: Structured output tracking with kernels," in *Proc. ICCV*. IEEE, 2011, pp. 263–270.

[52] S. Wang, H. Lu, F. Yang, and M.-H. Yang, "Superpixel tracking," 2011.

[53] L. Zhang and L. van der Maaten, "Structure preserving object tracking," in *Proc. CVPR*, 2013, pp. 1838–1845.

[54] A. Milan, S. Roth, and K. Schindler, "Continuous energy minimization for multitarget tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 1, pp. 58–72, 2014.

[55] E. Bochinski, V. Eiselein, and T. Sikora, "High-Speed tracking-by-detection without using image information," in *Proc. IEEE AVSS*, 2017.

[56] J. Chen, H. Sheng, Y. Zhang, and Z. Xiong, "Enhancing Detection Model for Multiple Hypothesis Tracking," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 2143–2152.

[57] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," *LNCS*, vol. 9914, pp. 850–865, 2016.

[58] J. Son, M. Baek, M. Cho, and B. Han, "Multi-Object Tracking with Quadruplet Convolutional Neural Networks," *Cvpr*, pp. 5620–5629, 2017.

[59] S. Schulter, P. Vernaza, W. Choi, and M. Chandraker, "Deep Network Flow for Multi-Object Tracking," *Cvpr*, pp. 6951–6960, 2017.

[60] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Detect to Track and Track to Detect," *Proc. ICCV*, vol. 2017-Octob, pp. 3057–3065, 2017.

[61] E. Insafutdinov, M. Andriluka, L. Pishchulin, S. Tang, E. Levinkov, B. Andres, and B. Schiele, "ArtTrack: Articulated multi-person tracking in the wild," *Proc. CVPR*, vol. 2017, pp. 1293–1301, 2017.

[62] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," *Proc. CVPR*, vol. 1, pp. 539–546, 2005.

[63] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, "CNN architectures for large-scale audio classification," *Proc. ICASSP*, no. Nips, pp. 131–135, 2017.

[64] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with siamese region proposal network," in *Proc. CVPR*, 2018, pp. 8971–8980.

[65] Y. Zhang, D. Wang, L. Wang, J. Qi, and H. Lu, "Learning regression and verification networks for long-term visual tracking," *arXiv preprint arXiv:1809.04320*, 2018.

[66] A. Milan, L. Leal-Taixe, I. Reid, S. Roth, and K. Schindler, "MOT16: A Benchmark for Multi-Object Tracking," pp. 1–12, 2016.

[67] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *Proc. CVPR*, 2012, pp. 3354–3361.

[68] L. Patino, T. Cane, A. Vallee, and J. Ferryman, "PETS 2016: Dataset and Challenge," in *Proc. CVPRW*, 2016, pp. 1240–1247.

[69] F. Yang, W. Choi, and Y. Lin, "Exploit All the Layers: Fast and Accurate CNN Object Detector with Scale Dependent Pooling and Cascaded Rejection Classifiers," in *Proc. CVPR*, 2016, pp. 2129–2137.

[70] P. Dollar, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[71] L. Wang, Y. Lu, H. Wang, Y. Zheng, H. Ye, and X. Xue, "Evolving boxes for fast vehicle detection," *Proceedings - IEEE International Conference on Multimedia and Expo*, pp. 1135–1140, 2017.

[72] 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*, vol. 9905 LNCS, pp. 21–37, 2016.

[73] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *International Conference on Learning Representations (ICRL)*, pp. 1–14, 2015.

[74] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[75] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," *Proceedings of the 27th International Conference on Machine Learning*, no. 3, pp. 807–814, 2010.

[76] A. Paszke, G. Chanan, Z. Lin, S. Gross, E. Yang, L. Antiga, and Z. Devito, "Automatic differentiation in PyTorch," *Advances in Neural Information Processing Systems 30*, no. Nips, pp. 1–4, 2017.

[77] S. Zhang, A. E. Choromanska, and Y. LeCun, "Deep learning with elastic averaging SGD," in *Advances in Neural Information Processing Systems*, 2015, pp. 685–693.

[78] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The CLEAR MOT metrics," *Eurasip Journal on Image and Video Processing*, vol. 2008, 2008.

[79] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *Lecture Notes in Computer Science*, vol. 9914 LNCS, 2016, pp. 17–35.
[80] Y. Li, C. Huang, and R. Nevatia, "Learning to associate: Hybridboosted multi-target tracker for crowded scene," in *Proc. CVPRW*, 2009, pp. 2953–2960.
[81] M. Keuper, S. Tang, Y. Zhongjie, B. Andres, T. Brox, and B. Schiele, "A Multi-cut Formulation for Joint Segmentation and Tracking of Multiple Objects," *CoRR*, vol. abs/1607.0, 2016.
[82] C. Kim, F. Li, and J. Rehg, "Multi-object Tracking with Neural Gating Using Bilinear LSTM," in *Proc. ECCV*, 2018, pp. 200–215.
[83] R. Sanchez-Matilla, F. Poiesi, and A. Cavallaro, "Online Multi-target Tracking with Strong and Weak Detections," in *Computer Vision – ECCV 2016 Workshops*, G. Hua and H. Jégou, Eds. Cham: Springer International Publishing, 2016, pp. 84–99.
[84] V. Eiselein, D. Arp, M. Pätzold, and T. Sikora, "Real-time multi-human tracking using a probability hypothesis density filter and multiple detectors," in *Proc. IEEE AVSS 2012*, 2012, pp. 325–330.
[85] B. Wang, L. Wang, B. Shuai, Z. Zuo, T. Liu, K. L. Chan, and G. Wang, "Joint Learning of Convolutional Neural Networks and Temporally Constrained Metrics for Tracklet Association," in *Proc. CVPRW*, 2016.
[86] L. Leal-Taixe, C. Canton-Ferrer, and K. Schindler, "Learning by Tracking: Siamese CNN for Robust Target Association," in *Proc. CVPRW*, 2016.
[87] S. Wang and C. C. Fowlkes, "Learning Optimal Parameters for Multi-target Tracking with Contextual Interactions," *International Journal of Computer Vision*, 2017.
[88] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the Untrackable: Learning to Track Multiple Cues with Long-Term Dependencies," in *Proc. ICCV*, 2017.
[89] Q. Chu, W. Ouyang, H. Li, X. Wang, B. Liu, and N. Yu, "Online Multi-object Tracking Using CNN-Based Single Object Tracker with Spatial-Temporal Attention Mechanism," in *Proc. ICCV*, 2017.
[90] J. H. Yoon, C.-R. Lee, M.-H. Yang, and K.-J. Yoon, "Online Multi-object Tracking via Structural Constraint Event Aggregation," in *Proc. CVPR*, 2016.
[91] A. Milan, S. H. Rezatofighi, A. Dick, K. Schindler, and I. Reid, "Online Multi-target Tracking using Recurrent Neural Networks," *Arxiv*, 2016.
[92] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. CVPR*, 2014.
[93] Z. Cai, M. Saberian, and N. Vasconcelos, "Learning complexity-aware cascades for deep pedestrian detection," in *Proc. ICCV*, vol. 2015 Inter, 2015, pp. 3361–3369.
[94] A. Andriyenko, K. Schindler, and S. Roth, "Discrete-continuous optimization for multi-target tracking," in *Proc. CVPR*, 2012, pp. 1926–1933.
[95] C. Dicle, O. I. Camps, and M. Sznaier, "The way they move: Tracking multiple targets with similar appearance," in *Proc. ICCV*, 2013, pp. 2304–2311.
[96] A. Andriyenko and K. Schindler, "Multi-target tracking by continuous energy minimization," in *Proc. CVPR*, 2011, pp. 1265–1272.
[97] N. Wojke and A. Bewley, "Deep cosine metric learning for person re-identification," in *Proc. WACV*. IEEE, 2018, pp. 748–756.
[98] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli *et al.*, "Image quality assessment: from error visibility to structural similarity," *IEEE TIP*, vol. 13, no. 4, pp. 600–612, 2004.

ShiJie Sun received his B.S. in software engineering from The University of Chang'an University and is currently working towards the Ph.D. degree in intelligent transportation and information system engineering with Chang'an University. Currently, he is a visiting (joint) Ph.D. candidate at the University of Western Australia since October 2017. His research interests include machine learning, object detection, localization & tracking, action recognition.

Naveed Akhtar received his PhD in Computer Vision from The University of Western Australia (UWA) and Master degree in Computer Science from Hochschule Bonn-Rhein-Sieg, Germany. His research in Computer Vision is regularly published in reputed venues of the filed. He also serves as an Associate Editor of IEEE Access. Currently, he is a Lecturer at UWA. Previously, he also served as a Research Fellow at UWA and the Australian National University. His research interests include multiple object tracking, adversarial deep learning, action recognition, and hyperspectral image analysis.

HuanSheng Song received the B.S. and M.S. degrees in communication and electronic systems and the Ph.D. degree in information and communication engineering from Xian Jiaotong University, Xian, China, in 1985, 1988, and 1996, respectively. Since 2004, he has been with the Information Engineering Institute, Changan University, Xian, where he became a Professor in 2006 and was nominated as the Dean in 2012. He has been involved in research on intelligent transportation systems for many years and has led a research team to develop a vehicle license plate reader and a traffic event detection system based on videos, which has brought about complete industrialization. His current research interests include image processing, recognition and tracking as well as intelligent transportation systems.

Ajmal Mian is a Professor of Computer Science at The University of Western Australia. He has received two prestigious fellowships and several research grants from the Australian Research Council and the National Health and Medical Research Council of Australia with a combined funding of over $12 million. He was the West Australian Early Career Scientist of the Year 2012 and has received several awards including the Excellence in Research Supervision Award, EH Thompson Award, ASPIRE Professional Development Award, Vice-chancellors Mid-career Award, Outstanding Young Investigator Award, the Australasian Distinguished Dissertation Award and various best paper awards. His research interests are in computer vision, machine learning, 3D shape analysis, face recognition, human action recognition and video analysis.

Mubarak Shah the trustee chair professor of computer science, is the founding director of the Center for Research in Computer Vision at University of Central Florida. He is an editor of an international book series on video computing, was editor-in-chief of Machine Vision and Applications journal, and an associate editor of ACM Computing Surveys journal. His research interests include video surveillance, visual tracking, human activity recognition, visual analysis of crowded scenes, video registration, UAV video analysis, and so on. He is a fellow of the IEEE, AAAS, IAPR, and SPIE.