Multi-Object Tracking with Quadruplet Convolutional Neural Networks

Jeany Son Mooyeol Baek Minsu Cho Bohyung Han
Dept. of Computer Science and Engineering, POSTECH, Korea
{jeany, mooyeol, mscho, bhhan}@postech.ac.kr

Abstract

We propose Quadruplet Convolutional Neural Networks (Quad-CNN) for multi-object tracking, which learn to associate object detections across frames using quadruplet losses. The proposed networks consider target appearances together with their temporal adjacencies for data association. Unlike conventional ranking losses, the quadruplet loss enforces an additional constraint that makes temporally adjacent detections more closely located than the ones with large temporal gaps. We also employ a multi-task loss to jointly learn object association and bounding box regression for better localization. The whole network is trained end-to-end. For tracking, the target association is performed by minimax label propagation using the metric learned from the proposed network. We evaluate performance of our multi-object tracking algorithm on public MOT Challenge datasets, and achieve outstanding results.

1. Introduction

Visual tracking for multiple targets in videos has been widely studied for various applications, such as human motion analysis, autonomous driving, and video surveillance. Despite substantial progress in recent years, even the state-of-the-art multi-object tracking algorithms still suffer from various challenges such as severe occlusions and noisy detections in crowded scenes. Such issues frequently affect tracking performance in real world scenarios.

Multi-object tracking aims to find an optimal set of trajectories of moving objects within a video. This problem is typically formulated as a data association task, where an external detector localizes target bounding boxes in each frame, and then a tracking algorithm associates corresponding detection boxes across frames. This data association is a challenging task in the presence of occlusions, missing objects, and false alarms. Hence, rather than relying on associations over two consecutive frames, existing methods typically leverage multiple candidate trajectories within a larger temporal window [56, 3, 1, 6, 50, 8, 22]. In spite of these efforts, existing multi-object tracking algorithms still suffer from massive and inaccurate detections.

In recent years, deep learning techniques have achieved the state-of-the-art performance in a variety of computer vision tasks such as image classification [25, 44, 17], semantic segmentation [31, 36], and object tracking [35, 16]. However, there are only a few deep learning approaches to multi-object tracking [28, 48, 33], and their performances are not as competitive as the techniques based on hand-crafted features. There are a couple of reasons that hamper the use of deep learning techniques for multi-object tracking. First, training data for multi-object tracking is not yet sufficient to train deep neural networks with a large number of parameters. Only a limited number of sequences are available due to the cost of annotating ground-truths for video frames. Second, existing deep neural networks pretrained on the datasets for image classification have critical limitations in discriminating objects with subtle difference and capturing motion features in video. While the success of multi-object tracking relies on the effective use of both target appearance and motion, joint learning of the two factors in deep neural networks has not been investigated in depth.

Motivated by this fact, we propose a novel multi-object tracking algorithm using Quadruplet Convolutional Neural Networks (Quad-CNN), which learns to associate detections across video frames using both appearance and motion cues. Specifically, unlike conventional ranking losses, the proposed quadruplet loss introduces an additional constraint that temporally adjacent detections have smaller distances than distant ones. This allows us to learn temporally smooth appearance models of target objects, and is realized by combining appearance embedding with motion-aware position embedding for metric learning. In addition, we incorporate bounding-box regression to refine initial detections and improve localizations. We employ a multi-task loss to jointly learn object association and bounding-box regression, and the whole network is trained end-to-end. In tracking, we compute distances between all pairs of detections within a temporal sliding window using the learned metric, and apply a minimax label propagation to associate detections.

The main contributions of this paper is four-fold:

- We propose a quadruplet architecture of deep neural

network, referred to as Quad-CNN, to learn object association for multi-object tracking. For metric learning, the Quad-CNN combines appearance embedding of detections and their sequence-specific motion-aware position embedding.

- We employ a multi-task loss to jointly learn object association and bounding-box regression, and the whole network is trained end-to-end in a unified framework.
- We adopt a modified minimax label propagation algorithm to make fast and robust data association for multi-object tracking.
- We achieve outstanding performance in MOT challenge benchmark datasets especially among the algorithms based on deep neural networks.

The rest of the paper is organized as follows. We first discuss related work in Section 2. The architecture and characteristics of Quad-CNN are presented in Section 3. The procedure of our association algorithm based on minimax label propagation is discussed in Section 4. Section 5 describes the implementation details of our method. Section 6 provides experimental results.

2. Related Work

Early multi-object tracking algorithms handling data association problems often use recursive Bayesian filters such as Kalman filter [4] and particle filter [38], which rely on the first-order Markov assumption. Another direction is to match object hypotheses given by detections between two consecutive frames using their affinities measure by appearance, position, size, etc. [24, 52, 42]. However, tracking algorithms based on local data association (*e.g.*, between two adjacent frames) have critical limitations in handling occlusions or noisy detections, and consequently tend to produce short fragmented trajectories. On the contrary, some multi-target tracking algorithms construct a set of trajectories through global or delayed optimization [56, 3, 1, 6, 50].

Several multi-object tracking algorithms based on convolutional neural networks (CNNs) [28, 48] and recurrent neural networks (RNNs) [33] have been proposed, but the benefit of deep neural networks are substantial even compared with hand-crafted features. Leal-Taixe *et al*. [28] learn descriptors using Siamese CNN, where images and optical flow maps are provided as multi-modal inputs. They use gradient boosting to combine local features extracted by Siamese CNN and contextual features. Wang *et al*. [48] jointly learn Siamese CNNs and temporally constraint metrics to obtain appearance-based tracklet affinity model. A Long Short-Term Memory (LSTM) is trained end-to-end for online multi-object tracking [33]. This work is the first fully end-to-end learning method based on deep learning, but its performance does not reach the accuracy of the state-of-the-art methods. Kim *et al*. [22] uses deep features pre-trained on large datasets as appearance features for multiple hypothesis tracking.

Multi-object tracking aims to associate the detections, so the design of a similarity function between detections is a critical factor. Siamese network [5, 9] and triplet network [49, 18] are simple methods to measure the similarity between two objects. Siamese network uses a contrastive loss to train the network, which encourages the network to have small distances between the pairs that belong to the same objects while enforcing the object with different identities to have large distances. This network is applied to face verification and identification [45, 43], single object tracking [47] and multi-target tracking [28, 48]. The triplet network, an improved version of Siamese network, is more discriminative and more robust to intra-class variations [18] since it uses a ranking loss. It has been used for feature learning [18, 26], unsupervised representation learning in videos [51], and face recognition [41] and person re-identifications [7]. Recently, generalized versions of the triplet network using higher order relationships have been proposed [57, 19, 37], and these methods are useful for fine-grained feature representation learning.

The most distinctive part of our algorithm from existing multi-object tracking algorithms based on metric learning is that it learns metrics for both appearance and motion cues simultaneously in a single CNN framework using quadruplet relationships. It is also notable that we obtain sequence-agnostic models for metric learning regardless of intrinsic and extrinsic camera parameters.

3. Quad-CNN for Multi-Object Tracking

This section describes the details of our Quad-CNN for multi-object tracking, including how to learn data association and bounding box regression jointly.

3.1. Main Idea

Our quadruplet network deals with multi-level rank orders by generalizing Siamese and triplet networks, which are often employed to learn embedding of target appearances. We are motivated by the fact that, for data association in multi-object tracking, the embedding should consider not only class labels of detected objects but also their detection timestamps since object appearances change over time in videos. We introduce a Quad-CNN to learn an embedding with such constraint, where the similarity between detected objects are determined by both their labels and temporal distances. Figure 1 illustrates the quadruplet relationship defined in this paper.

Since this quadruplet association relies on accurate object localization, we employ bounding-box regression as an additional objective to learn the network. The Quad-CNN

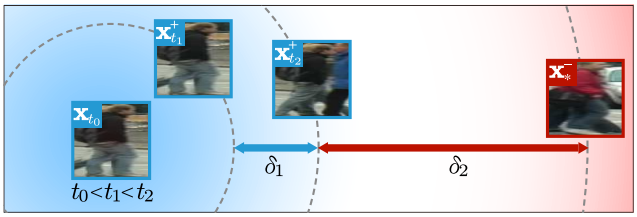


Figure 1: Quadruplet relationship in our Quad-CNN. We aim to enforce a positive pair of detections to have a smaller distance than a negative pair, and a temporally adjacent pair of detections to have a smaller distance than a temporally distant pair.

minimizes a multi-task loss $\mathcal{L}$ on each patch in a mini-batch corresponding to both data association and bounding box regression, which is given by

$$\mathcal{L} = \mathcal{L}_{\text{rank}} + \lambda \mathcal{L}_{\text{bbreg}}, \tag{1}$$

where $\mathcal{L}_{\text{rank}}$ and $\mathcal{L}_{\text{bbreg}}$ are losses for quadruplet ranking and bounding box regression, respectively.

Figure 2 illustrates the overall architecture of our Quad-CNN, where the CNNs can be replaced by any deep network (*e.g.* VGG, AlexNet).

3.2. Quadruplet Rank Loss for Robust Association

Let us denote a set of N quadruplets of image patches, corresponding to detections in multi-object tracking, by $\{(\mathbf{x}_{i,t_0}, \mathbf{x}^+_{i,t_1}, \mathbf{x}^+_{i,t_2}, \mathbf{x}^-_{i,*})\}_{i=1}^N$, where $\mathbf{x}_{t_0}$ is an anchor patch at frame t_0, $(\mathbf{x}^+_{t_1}, \mathbf{x}^+_{t_2})$ are positive patches with constraints $t_0 < t_1 < t_2$, and $\mathbf{x}^-_*$ is a negative patch from arbitrary frames. Given the quadruplet, the relations among these four image patches are given by

$$d(\mathbf{x}_{t_0}, \mathbf{x}^+_{t_1}) + \delta_1 + \delta_2 < d(\mathbf{x}_{t_0}, \mathbf{x}^+_{t_2}) + \delta_2 < d(\mathbf{x}_{t_0}, \mathbf{x}^-_*), \tag{2}$$

where $d(\cdot, \cdot)$ is a distance metric between the features of two patches extracted from the last fully connected layer of the network, and (δ_1, δ_2) denote margins ($\delta_1 \ll \delta_2$). The ranking loss of the Quad-CNN is expressed as

$$\begin{aligned} &\mathcal{L}_{\text{rank}} = \\ &\frac{1}{2N}\sum_{i=1}^{N} \max\{0, d(\mathbf{x}_{i,t_0}, \mathbf{x}^+_{i,t_1}) - d(\mathbf{x}_{i,t_0}, \mathbf{x}^+_{i,t_2}) + \delta_1\} \\ &+ \frac{1}{2N}\sum_{i=1}^{N} \max\{0, d(\mathbf{x}_{i,t_0}, \mathbf{x}^+_{i,t_2}) - d(\mathbf{x}_{i,t_0}, \mathbf{x}^-_{i,*}) + \delta_2\}. \end{aligned} \tag{3}$$

For robust association between detections, we learn a distance metric $d(\cdot, \cdot)$ using position discrepancy and appearance dissimilarity, which is given by

$$d(\mathbf{x}_i, \mathbf{x}_j) = \alpha^{ij}_a \|\mathbf{a}_i - \mathbf{a}_j\|_2^2 + \alpha^{ij}_m \|\mathbf{m}_{i\to j} - \mathbf{m}_{j\to j}\|_2^2, \tag{4}$$

where $\mathbf{a}_i$ denotes a learned appearance feature of patch $\mathbf{x}_i$, $\mathbf{m}_{i\to j}$ means a sequence-specific motion-aware position feature using a linear motion model from patch $\mathbf{x}_i$ to $\mathbf{x}_j$, and $(\alpha^{ij}_a, \alpha^{ij}_p)$ are trained weights for distances of appearance and position features. Detailed description about these two features and the learned metric are discussed below.

Appearance feature In multi-object tracking of pedestrians (or many other kinds of objects), the extent of main target objects can be roughly into two parts, *e.g.*, upper body and lower body. When the number of training examples is not sufficient, it may be difficult to learn a robust feature embedding of joint appearance for upper and lower body due to overfitting issue. To address this problem, we slice the output of the last convolution layer (pool5) into two corresponding parts, and learn two fully connected layers (f_{top} and f_{bottom}) separately for each part. Compared to a single combined linear layer, these two sliced layers reduce the total number of parameters since they do not share the connected nodes in the preceding layer. According to our experiment, this slicing strategy prevents the network from overfitting and improves accuracy of pedestrian tracking. Outputs of these separate fully connected layers (upper and lower body adaptation layers) are then concatenated to generate a single appearance feature vector $\mathbf{a}_i$ for patch $\mathbf{x}_i$.

Sequence-specific motion-aware position feature In addition to appearance features, position features of patches are trained by the proposed Quad-CNN. Let us first define a motion-aware position feature from patch $\mathbf{x}_i$ to $\mathbf{x}_j$, which is learned from the following input vector:

$$\mathbf{p}_{i\to j} = [u_i + \dot{u}_i \cdot \Delta t_{ij}, v_i + \dot{v}_i \cdot \Delta t_{ij}, w_i, h_i, \Delta t_{ij}], \tag{5}$$

where $[u_i, v_i]$ is the center position of the detection $\mathbf{x}_i$, $[\dot{x}_i, \dot{y}_i]$ is the velocity vector at $[u_i, v_i]$, (w_i, h_i) are width and height of detection $\mathbf{x}_i$, and Δt_{ij} is the temporal difference between detection $\mathbf{x}_i$ and $\mathbf{x}_j$. Note that the velocity vector is estimated by a linear motion model with the optical flow between adjacent frames computed by [11]. Then, the embedding network f_{pos} extracts a motion-aware position feature by

$$\hat{\mathbf{p}}_{i\to j} = f_{\text{pos}}(\bar{\mathbf{p}}_{i\to j}; \theta_{\text{pos}}), \tag{6}$$

where $\bar{\mathbf{p}}_{i\to j}$ is the transformation of $\mathbf{p}_{i\to j}$ by bounding box regression (discussed in Section 3.3), θ_{pos} is the model parameter for the embedding network. Note the motion-aware position embedding learns canonical spaces for the positions of all detections. This motion-aware position embedding is learned based on the criterion that position feature $\hat{\mathbf{p}}_{i\to j}$ with a estimated motion from $\mathbf{x}_i$ to $\mathbf{x}_j$ should be close to the position feature $\hat{\mathbf{p}}_{j\to j}$ without motion if two detections belong to the same object.

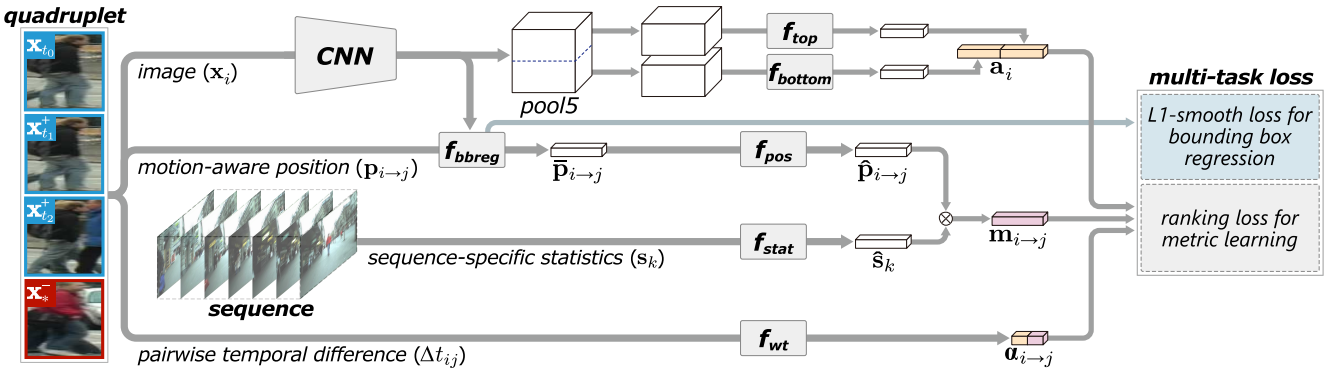


Figure 2: The architecture of the proposed Quad-CNN for multi target tracking using temporal coherency.

The motion-aware position embedding is reasonable but not sufficient to measure universal similarity for all sequences. This is because all videos have different geometric configurations mainly due to the variations of intrinsic and extrinsic camera parameters. For example, objects captured by a low-angle camera typically have large scale variations even with subtle changes in their y-coordinates while objects in a top-down view do not have any correlation between their locations and sizes. Therefore, the features in 2D image space should be properly normalized and adjusted to handle various camera positions and orientations in a unified framework.

We propose a sequence-specific embedding by reflecting the characteristics of the sequences. Let $\mathbf{s}_k$ denote statistics of a sequence k, which is given by

$$\mathbf{s}_k = [\sigma(\mathbf{X}_k)/\sigma(\mathbf{Y}_k), \sigma(\mathbf{X}_k), \mu(\mathbf{W}_k), \sigma(\mathbf{W}_k), \omega_k], \quad (7)$$

where $\mathbf{X}_k, \mathbf{Y}_k$ are a set of 2D coordinates of all detections in sequence k, $\mathbf{W}_k$ is a set of widths of all detections in sequence k, ω_k is the width of a frame in sequence k, and $\mu(\cdot)$ and $\sigma(\cdot)$ denote mean and standard deviation, respectively. Then, the sequence-specific embedding features are obtained by learning the following function:

$$\hat{\mathbf{s}}_k = f_{\text{stat}}(\mathbf{s}_k; \theta_{\text{stat}}), \quad (8)$$

where f_{stat} is a network for sequence statistics embedding, θ_{stat} is the model parameter of f_{stat}.

To obtain the sequence-specific motion-aware position embedding $\mathbf{m}_{i\to j}$, we employ multiplicative interactions between sequence statistics and motion-aware position predictions. It learns the representation using correlations between two vectors, and the sequence-specific motion-aware position prediction $\mathbf{m}_{\mathbf{i}\to\mathbf{j}}$ can be written as

$$\mathbf{m}_{i\to j} = \hat{\mathbf{s}}_k \odot \hat{\mathbf{p}}_{i\to j}, \quad (9)$$

where $\odot$ denotes Hadamard product.

Feature weighting Since the inputs of two features—appearance and position features—have completely different characteristics in terms of magnitude and dimensionality, it is not straightforward to estimate their relative importance. Hence, we learn the weights for the two distance terms in Eq. (4) with respect to temporal differences of detections. In other words, if detections are close in time, position information is more important than appearance, while, if they are temporally distant from each other, then it would be better to focus on their appearances. Thus, given a temporal difference Δt_{ij} of two patches $\mathbf{x}_i$ and $\mathbf{x}_j$, the weights $(\alpha_a^{ij}, \alpha_p^{ij})$ in Eq. (4) are learned as

$$\boldsymbol{\alpha}_{i\to j} \equiv [\alpha_a^{ij}, \alpha_p^{ij}] = f_{\text{wt}}(\Delta t_{ij}; \theta_{\text{wt}}), \quad (10)$$

where f_{wt} is the feature weighting network and θ_{wt} is the model parameter for f_{wt}.

3.3. Bounding Box Regression Loss

To handle noisy localization of detected objects, we perform bounding box regression as in [14]. The bounding box regression loss $\mathcal{L}_{\text{bbreg}}$ is given by

$$\mathcal{L}_{\text{bbreg}} = \sum_{i\in\{u,v,w,h\}} \text{smooth}_{L_1}(g_i - p_i), \quad (11)$$

where $g = \{g_u, g_v, g_w, g_h\}$ denotes an offset of ground-truth bounding box, and $p = \{p_u, p_v, p_w, p_h\}$ is a predicted bounding-box regression offset. The L_1 smooth loss function is given by

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise.} \end{cases} \quad (12)$$

For bounding box regression, we adopt the parameterizations of the 4 coordinates of g as in [14].

We update the input of the network for motion-aware position feature, $\mathbf{p}_{i\to j}$, by bounding box regression and obtain $\bar{\mathbf{p}}_{i\to j}$, which is given by

$$\bar{\mathbf{p}}_{i\to j} = [\bar{u}_i + \dot{u}_i \cdot \Delta t_{ij}, \bar{v}_i + \dot{v}_i \cdot \Delta t_{ij}, \bar{w}_i, \bar{h}_i, \Delta t_{ij}] \quad (13)$$

where the coordinates of bounding boxes are updated as

$$\bar{u}_i = w_i \cdot p_u + u_i, \qquad \bar{w}_i = w_i \cdot \exp(p_w),$$
$$\bar{v}_i = h_i \cdot p_v + v_i, \qquad \bar{h}_i = h_i \cdot \exp(p_h).$$

Note that the updated input $\bar{\mathbf{p}}_{i \to j}$ is used to compute motion-aware position feature as shown in Eq. (6).

3.4. Training and Inference

We optimize the parameters by backpropagating the joint loss in terms of ranking and bounding box regression given by Eq. (1). Training is straightforward for both loss terms since the ranking loss is based on hinge losses and the bounding box regression is a well-known technique.

For inference in a video with given detections, we extract appearance features, perform bounding box regressions, and compute motion-aware position features for all detections. The sequence statistics are also computed using all detections in the current video, and the sequence-specific motion-aware position features are obtained by combining the sequence statistics and motion-aware position features. The weights for appearance and position features are also computed for all pairs of detections in the same temporal sliding window. Using the sequence-specific motion-aware position features and the appearance features, pairwise distances are computed by using the learned metric in Eq. (4). Then, target association can be achieved by our minimax label propagation, which is discussed in Section 4.

4. Minimax Label Propagation for MOT

We now describe the proposed tracking framework to solve data association problem based on a modified minimax label propagation.

4.1. Problem Formulation

Multi-object tracking can be formulated with graphs, where the cost of an edge corresponds to a distance between two end nodes. Optimization methods such as k-shortest path algorithm [3] and network flow [39, 50] are typically used to find the optimal trajectories of targets.

We employ the minimax label propagation technique for semi-supervised multi-class classification proposed in [23] due to its simplicity and fast speed. We modify the original algorithm to fit multi-object tracking application; we enforce a pairwise exclusion constraint [34] that detections within the same frame do not have the same label, and add the capability to generate new labels for handling entering objects. Detections are associated using minimax label propagation along the minimax paths and labels are propagated through the paths from labelled to unlabeled nodes. In comparison to the shortest path algorithm, also known as Dijkstra algorithm, the minimax label propagation only takes account of the maximum edge costs along the path.

(a) Minimax path (b) Shortest path

Figure 3: Minimax path vs. Shortest path. Each number colored in white denotes the optimal distance to the node from initial labelled nodes, and the number on each edge denotes a distance (cost) between two patches. (a) The minimax path minimizes the maximum edge cost on the path. (b) The shortest path minimizes the sum of all edge costs along the path.

Since this method attempts to maintain small variations of associated detections along each path, we believe that it is well suited for matching problems. Furthermore, minimax paths better captures cluster structures of nonconvex shapes, compared to the shortest paths. Figure 3 illustrates the difference between the minimax path and the shortest path methods.

4.2. Minimax Label Propagation

Graph construction Let us denote a directed acyclic graph for multi-object tracking by $G = (V, E)$ where V and E represent sets of nodes and edges, respectively. A node $v_i \in V$ corresponds to a detection $\mathbf{x}_i$, and an edge $e_{ij} \in E$ connects a pair of detections, $\mathbf{x}_i$ and $\mathbf{x}_j$, within each temporal sliding window. The learned distance metric in Eq. (4) is used to compute a cost from v_i to v_j:

$$c(i, j) \equiv d(\mathbf{x}_i, \mathbf{x}_j). \tag{14}$$

The set of edges E only contains pairs of detections satisfying the following criteria:

$$E = \{e_{ij} | c(i,j) \leq c(i,k) \ {}^{\forall}k \in \{l | t_l = t_j\},$$
$$c(i,j) < r_{\text{thr}},\ t_i < t_j,\ |t_j - t_i| < \tau\}, \tag{15}$$

where t_j is a timestamp of v_j, τ is a size of temporal windows and r_{thr} is a threshold value for reliable pairs. We use the margin δ_2 as a threshold value r_{thr}. Note that, for a given node v_i, we choose a node v_j in each future frame that can be reached with the minimum cost. All the edges are connected forward with respect to detection timestamps, and no edge is created between nodes in the same frame. Since we allow pairs of detections to be connected in a temporal sliding window, short-term occlusion or missing detections can be handled.

Algorithm 1 Minimax Label Propagation for MOT

for $j \in \{1, \dots, N_\ell\}$ **do** $r_j \leftarrow 0, \ell_j \leftarrow j$
for $j \in \{N_\ell + 1, \dots, N\}$ **do** $r_j \leftarrow \infty, \ell_j \leftarrow$ undef
$\mathcal{Q}^{(0)} \leftarrow \{1, \dots, N_\ell\}$, id $\leftarrow N_\ell + 1$
for $m \in \{0, 1, 2, \dots\}$, until $\mathcal{Q}^{(m)} = \{\}$ **do**
  $\mathcal{Q}^{(m+1)} \leftarrow \{\}$
  for $i \in \mathcal{Q}^{(m)}$ **do**
    $\mathcal{N}_i \leftarrow \{\}$
    for $T \in \{t_i + 1, \dots, t_i + \tau\}$ **do**
      if $\min_{j \in \{l | t_l = T\}} c(i, j) < r_{\text{thr}}$ **then**
        $\mathcal{N}_i \leftarrow \mathcal{N}_i \cup \{\arg\min_{j \in \{l | t_l = T\}} c(i, j)\}$
    for $j \in \mathcal{N}_i$ **do**
      $r_j^* \leftarrow \max(c(i, j), r_i),\ r' \leftarrow \infty$
      if $(\ell_i = \ell_{j'})$ & $(t_j = t_{j'}),\ ^\exists j'$ **then**
        $r' \leftarrow r_{j'}$
      if $(r_j^* < r_j)$ & $(r_j^* < r')$ **then**
        $r_j \leftarrow r_j^*, \ell_j \leftarrow \ell_i$
        if $r' < \infty$ **then** $r_{j'} \leftarrow \infty,\ \ell_{j'} \leftarrow$ undef
        $\mathcal{Q}^{(m+1)} \leftarrow \mathcal{Q}^{(m+1)} \cup j$
  if $\mathcal{Q}^{(m+1)} = \{\}$ & $\ell_k =$ undef, $^\exists k$ **then**
    $k' \leftarrow \arg\min_k t_k,\ \ell_{k'} \leftarrow$ id, $r_{k'} \leftarrow 0$
    $\mathcal{Q}^{(m+1)} \leftarrow \mathcal{Q}^{(m+1)} \cup k',$ id $\leftarrow$ id $+ 1$

Modified minimax label propagation Unlike conventional minimax label propagation, we use a pairwise exclusion constraint for multi-object tracking [34]: two detections in the same frame cannot be on the same target. Our minimax label propagation algorithm for multi-object tracking is summarized in Algorithm 1. From an initial set of N_ℓ nodes with labels, which corresponds to labelled bounding boxes in the first frame of video, the algorithm propagates the labels to detections in subsequent frames. We assign to each node v_j a minimax distance r_j from the initial labelled nodes, which is computed in a recursive manner:

$$i^* = \arg\min_{i \in \mathcal{P}_j} (\max(c(i, j), r_i)), \tag{16}$$

$$r_j = \max(c(i^*, j), r_{i^*}), \tag{17}$$

where $\mathcal{P}_j$ denotes an index set of immediate parents of v_j in G. Each label is propagated along the minimax paths in the constructed directed acyclic graph G. If there already exists a node j' having the same label ℓ_i in the same frame, the node with a lower cost takes the label. Once all labels are propagated, we assign a new label to one of unlabeled nodes in the earliest frame and propagate it in the same manner. This process is iterated until all nodes are labeled.

5. Implementation

This section describes training details of the proposed quadruplet network for multi-target tracking.

5.1. Architecture

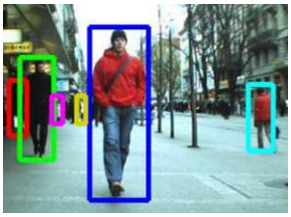
Figure 2 illustrates high-level design of our network. The architecture of convolutional layers for appearance feature learning are identical to the corresponding part of AlexNet [25]. We initialize the weights of convolutional layers using a pretrained triplet network [51], which is trained on a video dataset in an unsupervised manner.

We replace fully connected layers of AlexNet with our custom layers. Specifically, we divide pool5 features into upper and lower parts to handle each body part separately. Each of the two feature maps are then followed by two fully connected layers in 1,024 and 32 dimensions subsequent with ReLU. We concatenate the two separate feature maps to get the final appearance feature.

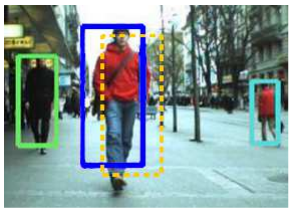
We employ a 5D fully connected layers to learn each of motion-aware position features $\hat{\mathbf{p}}_{i \to j}$ and sequence-specific embedding features $\hat{\mathbf{s}}_k$. The weights of appearance and position features are obtained from a 2D fully connected layer. For training, we use the standard stochastic gradient descent (SGD) method and initial learning rates are set to 0.0001 for the pretrained layers and 0.001 for the other layers. Initial momentum and weight decay are set to 0.9 and 0.0005, respectively. The network converges after approximately 8K SGD iterations with 200 examples in mini-batches.

Training with detections The training data for multi-object tracking typically provide detection bounding boxes as well as ground-truth bounding boxes with associated IDs. Most multi-object tracking algorithms train models with ground-truth bounding box annotations. However, since the characteristics of detections and ground-truths are different and only detection bounding boxes are available for inference, it is more desirable to use detection bounding boxes for training. In other words, we believe that the use of detections makes trained models more robust partly because the environment of training and testing are more consistent. However, training with detections is not straightforward because detection bounding boxes are not given ground-truth IDs. Hence, we match detections with ground-truths using Hungarian algorithm, and assign an ID to each detection in the training dataset. Figure 4 demonstrates the difference between training examples based on ground-truths and detections. Note that there are some detection bounding boxes with missing IDs due to misalignment with ground-truth, and missing detections due to imperfect detector performance.

Quadruplet sampling Since it is not plausible to enumerate all possible quadruplets in a mini-batch for training, we perform forward propagation of all examples in a mini-batch and compute the losses for a subset of quadruplets using extracted features. This quadruplet sampling strategy is important for effective error backpropagation and fast



(a) Ground-truths



(b) Detections with ID

Figure 4: Assigning IDs to detections using ground-truth bounding boxes. In (b), the dotted box denotes a detection with missing ID and we can find three misdetections.

convergence of training. To generate a quadruplet, we first pick an anchor example and two positive examples obtained from different frames in a mini-batch. Given the triplet of positive examples, we select a negative example that has a different ID from the anchor. Note that, all examples in a mini-batch are obtained from the same video. We have two options to select negative examples from a mini-batch as discussed in [51]. One is random selection of detections with different ID in the same video, and the other is to use a hard negative mining—collecting the top k examples that have the highest ranking losses in a mini-batch. In our experiments, we use 10 random negative samples and 10 hard negative samples for each anchor.

Data Augmentation We employ several data augmentation techniques widely used to training CNNs. We resize each input example to 256×256 and randomly crop 227×227 images from the input. The positions of the patches are adjusted by considering their translations offsets induced by cropping. All sequences are flipped horizontally and the flipped videos are treated as separate sequences for training. Color jittering is applied to reduce overfitting and to improve generalization of the network.

6. Experimental Results

This section presents our results on the standard benchmarks with comparisons to several recent algorithms.

6.1. Datasets

We tested our tracking algorithm on the Multiple Object Tracking (MOTChallenge) Benchmark[1] to evaluate performances. The MOTChallenge benchmark provides a set of sequences with ground-truth annotations and a framework for a fair performance evaluation of algorithms. The dataset contains video sequences collected from other datasets for multi-object tracking and new challenging sequences captured by both static and moving cameras along with detailed annotations. In 2DMOT2015 dataset [30], sequences are divided into two subsets—one is for training and the other is for testing—and each subset consists of 11 sequences. The 2DMOT2015 dataset also provides public object detection bounding boxes from the Aggregate Channel Features pedestrian detector [10]. MOT16 [32] dataset contains 7 training and 7 testing sequences, where DPM v5 [13] is used to obtain the bounding boxes.

[1] http://motchallenge.net/

Table 3: Results of Quad-CNN variants on the generated 2DMOT2015 validation set.

Method	MOTA↑	MT↑	ML↓
Quad-CNN	**16.9**	**24.6%**	**47.9%**
(a) Quad-CNN_noBBR	13.6	20.6%	50.4%
(b) Quad-CNN_CW	14.3	**24.6%**	49.0%
(c) Quad-CNN_noSLICE	14.3	21.5%	51.0%
(d) Quad-CNN_noSSE	15.2	22.4%	48.4%
TRIPLET	16.1	21.5%	52.4%

Table 4: Ablation study on the 2DMOT2015 test dataset.

Method	MOTA↑	MOTP↑	MT↑	ML↓
Quad-CNN	**33.8**	**73.4**	12.9%	**36.9%**
(a) TRIPLET	32.6	73.1	12.1%	42.7%
(b) TRIPLET_noBBR	29.4	71.6	**13.0%**	40.1%
(c) TRIPLET_CW	29.3	73.0	8.2%	48.5%

6.2. Evaluation Metrics

The MOTChallenge Benchmark employs multiple metrics for evaluation of multi-object tracking algorithms. These include Multiple Object Tracking Accuracy (MOTA), Multiple Object Tracking Precision (MOTP), average number of false alarms per frame (FAF), Mostly Track targets (MT, percentage of ground truth objects whose trajectories are covered by the tracking output at least 80%), Mostly Lost targets (ML, percentage of ground truth objects whose trajectories are covered by the tracking output less than 20%), the total number of False Positives (FP), the total number of False Negatives (FN), the total number of ID Switches (IDS), the total number of times a trajectory is Fragmented (Frag), and the number of frameworks processed in one second (Hz).

6.3. Evaluation on MOTChallenge Benchmark

We evaluated performance of the proposed quadruplet network denoted by Quad-CNN on 2DMOT2015 dataset. Table 1 presents the results, where arrows indicate favorable directions of quantitative results. Quad-CNN achieves the state-of-the-art performance and outperforms the algorithms based on deep neural networks with large margins.

We also present performance of Quad-CNN on MOT16 dataset in the MOTChallenge benchmark. The quantitative results are presented in Table 2, which shows that Quad-CNN is competitive with the state-of-the-art methods.

Table 1: Results on the 2DMOT2015 test dataset: (a) comparison with the state-of-the-art methods (b) comparison with the algorithms based on deep neural networks. Our method is denoted by Quad-CNN.

	Tracker	MOTA↑	MOTP↑	FAF↓	MT↑	ML↓	FP↓	FN↓	IDsw↓	Frag↓	Hz↑
	Quad-CNN	**33.8**	**73.4**	1.4	12.9%	**36.9%**	7,898	32,061	703	1,430	3.7
	NOMT [8]	33.7	71.9	1.3	12.2%	44.0%	7,762	32,547	442	**823**	11.5
	TDAM [54]	33.0	72.8	1.7	13.3%	39.1%	10,064	**30,617**	464	1,506	5.9
	MHT_DAM [22]	32.4	71.8	1.6	**16.0%**	43.8%	9,064	32,060	435	826	0.7
	MDP [53]	30.3	71.3	1.7	13.0%	38.4%	9,717	32,422	680	1,500	1.1
(b)	SCEA [55]	29.1	71.1	1.0	8.9%	47.3%	6,060	36,912	604	1,182	6.8
	LP_SSVM [50]	25.2	71.7	1.4	5.8%	53.0%	8,369	36,932	646	849	41.3
	LINF1 [12]	24.5	71.3	1.0	5.5%	64.6%	5,864	40,206	**298**	744	7.5
	JPDA_m [15]	23.8	68.2	1.1	5.0%	58.1%	6,373	40,084	365	869	32.6
	MotiCon [29]	23.1	70.9	1.8	4.7%	52.0%	10,404	35,844	1,018	1,061	1.4
	CNNTCM [48]	29.6	71.8	1.3	11.2%	44.0%	7,786	34,733	712	943	1.7
(c)	SiameseCNN [28]	29.0	71.2	**0.9**	8.5%	48.4%	**5,160**	37,798	639	1,316	52.8
	RNN_LSTM [33]	19.0	71.0	2.0	5.5%	45.6%	11,578	36,706	1,490	2,081	**165.2**

Table 2: Results on the MOT16 test dataset. Our method is denoted by Quad-CNN.

Tracker	MOTA↑	MOTP↑	FAF↓	MT↑	ML↓	FP↓	FN↓	IDsw↓	Frag↓	Hz↑
Quad-CNN	44.1	76.4	1.1	14.6%	44.9%	6,388	94,775	745	1,096	1.8
NOMT [8]	**46.4**	**76.6**	1.6	**18.3%**	41.4%	9,753	**87,565**	**359**	**504**	2.6
JMC [46]	46.3	75.7	1.1	15.5%	**39.7%**	6,373	90,914	657	1,114	0.8
oICF [21]	43.2	74.3	1.1	11.3%	48.5%	6,651	96,515	381	1,404	0.4
MHT_DAM [22]	42.9	76.6	1.0	13.6%	46.9%	5,668	97,919	499	659	0.8
LINF1 [12]	41.0	74.8	1.3	11.6%	51.3%	7,896	99,224	430	963	1.1
EAMTT_pub [40]	38.8	75.1	1.4	7.9%	49.1%	8,114	102,452	965	1,657	11.8
OVBT [2]	38.4	75.4	1.9	7.5%	47.3%	11,517	99,463	1,321	2,140	0.3
LTTSC-CRF [27]	37.6	75.9	2.0	9.6%	55.2%	11,969	101,343	481	1,012	0.6
JPDA_m [15]	26.2	76.3	**0.6**	4.1%	67.5%	**3,689**	130,549	365	638	**22.2**

6.4. Ablation Study

We performed ablation study to further investigate the effectiveness of our algorithm. For the purpose, we created a training and validation split using the original training set of 2DMOT2015, where training set includes TUD-Stadtmitte, PETS09-S2L1, ETH-Sunnyday, ADL-Rundle-6, KITTI-13, Venice-2 and validation set is composed of TUD-Campus, ETH-Bahnhof, ETH-Pedcross2, ADL-Rundle-8, KITTI-17.

Four variants of our algorithm are tested: Quad-CNN (a) without bounding box regression (noBBR), (b) with constant weights $\alpha_a = \alpha_m = 1$ (CW), (c) without slicing pool5 features (noSLICE) and (d) without sequence-specific embedding for position features (noSSE). TRIPLET denotes testing with the triplet loss. As illustrated in Table 3, our full model presents better performance than all others, which clearly suggests that all the integrated components in our method is helpful for performance gains.

We also conducted analysis on the 2DMOT2015 test set with three variants of the triplet loss: (a) positive/negative relationship only (TRIPLET), (b) TRIPLET without bounding box regression (TRIPLET_noBBR) and (c) TRIPLET with constant weights (TRIPLET_CW). All other settings of these three variants are identical to our Quad-CNN. Table 4 presents the results, which again shows clear benefit of quadruplet ranking loss, bounding box regression, and weight learning employed in our algorithm.

7. Conclusion

We proposed a new multi-object tracking algorithm using Quad-CNN, which has an additional constraint that enforces temporally adjacent detections to be more similar than the ones with large temporal gaps. To better capture the nature of multi-object tracking in the metric learning, the appearance embedding is combined with sequence-specific motion-aware position embedding. We employ a multi-task loss to jointly learn object association and bounding-box regression, and the whole network is trained end-to-end. Target association is performed by a minimax label propagation using the similarity learned from the proposed network. Our algorithm achieves the state-of-the art performance on the MOTChallenge benchmark.

Acknowledgements

This work was supported by the ICT R&D program of MSIP/IITP [2014-0-00147; Basic Software Research in Human-level Lifelong Machine Learning (Machine Learning Center), 2014-0-00059; Development of Predictive Visual Intelligence Technology (DeepView), 2016-0-00563; Research on Adaptive Machine Learning Technology Development for Intelligent Autonomous Digital Companion].

References

[1] A. Andriyenko and K. Schindler. Multi-target tracking by continuous energy minimization. In *CVPR*, 2011. 1, 2

[2] Y. Ban, S. Ba, X. Alameda-Pineda, and R. Horaud. Tracking multiple persons based on a variational bayesian model. In *ECCVW*, 2016. 8

[3] J. Berclaz, F. Fleuret, E. Türetken, and P. Fua. Multiple object tracking using k-shortest paths optimization. *TPAMI*, 33(9):1806–1819, 2011. 1, 2, 5

[4] J. Black, T. Ellis, and P. Rosin. Multi view image surveillance and tracking. In *Proceedings of the Workshop on Motion and Video Computing*, 2002. 2

[5] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah. Signature verification using a siamese time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688, 1993. 2

[6] A. Butt and R. Collins. Multi-target tracking by lagrangian relaxation to min-cost network flow. In *CVPR*, 2013. 1, 2

[7] D. Cheng, Y. Gong, S. Zhou, J. Wang, and N. Zheng. Person re-identification by multi-channel parts-based cnn with improved triplet loss function. In *CVPR*, 2016. 2

[8] W. Choi. Near-online multi-target tracking with aggregated local flow descriptor. In *ICCV*, 2015. 1, 8

[9] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005. 2

[10] P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *TPAMI*, 36(8):1532–1545, 2014. 7

[11] A. Dosovitskiy, P. Fischery, E. Ilg, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, T. Brox, et al. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2015. 3

[12] L. Fagot-Bouquet, R. Audigier, Y. Dhome, and F. Lerasle. Improving multi-frame data association with sparse representations for robust near-online multi-object tracking. In *ECCV*, 2016. 8

[13] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *TPAMI*, 32(9):1627–1645, 2010. 7

[14] R. Girshick. Fast r-cnn. In *ICCV*, 2015. 4

[15] S. Hamid Rezatofighi, A. Milan, Z. Zhang, Q. Shi, A. Dick, and I. Reid. Joint probabilistic data association revisited. In *ICCV*, 2015. 8

[16] B. Han, J. Sim, and H. Adam. Branchout: Regularization for online ensemble tracking with convolutional neural networks. In *CVPR*, 2017. 1

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

[18] E. Hoffer and N. Ailon. Deep metric learning using triplet network. In *Similarity-Based Pattern Recognition*, pages 84–92. 2015. 2

[19] C. Huang, C. C. Loy, and X. Tang. Local similarity-aware deep feature embedding. In *NIPS*, 2016. 2

[20] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[21] H. Kieritz, S. Becker, W. Hübner, and M. Arens. Online multi-person tracking using integral channel features. In *International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2016. 8

[22] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg. Multiple hypothesis tracking revisited. In *ICCV*, 2015. 1, 2, 8

[23] K.-H. Kim and S. Choi. Label propagation through minimax paths for scalable semi-supervised learning. *Pattern Recognition Letters*, 45:17–25, 2014. 5

[24] S. Kim, S. Kwak, J. Feyereisl, and B. Han. Online multi-target tracking by large margin structured learning. In *ACCV*, 2012. 2

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 6

[26] B. Kumar, G. Carneiro, and I. Reid. Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions. In *CVPR*, 2015. 2

[27] N. Le, A. Heili, and J.-M. Odobez. Long-term time-sensitive costs for crf-based tracking by detection. In *ECCVW*, 2016. 8

[28] L. Leal-Taixé, C. Canton-Ferrer, and K. Schindler. Learning by tracking: Siamese cnn for robust target association. In *CVPRW*, 2016. 1, 2, 8

[29] L. Leal-Taixé, M. Fenzi, A. Kuznetsova, B. Rosenhahn, and S. Savarese. Learning an image-based motion context for multiple people tracking. In *CVPR*, 2014. 8

[30] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler. Motchallenge 2015: Towards a benchmark for multi-target tracking. *arXiv preprint arXiv:1504.01942*, 2015. 7

[31] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1

[32] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. MOT16: A benchmark for multi-object tracking. *arXiv:1603.00831 [cs]*, Mar. 2016. arXiv: 1603.00831. 7

[33] A. Milan, S. H. Rezatofighi, A. Dick, I. Reid, and K. Schindler. Online multi-target tracking using recurrent neural networks. In *AAAI*, 2016. 1, 2, 8

[34] A. Milan, K. Schindler, and S. Roth. Multi-target tracking by discrete-continuous energy minimization. *TPAMI*, 38(10):2054–2068, 2016. 5, 6

[35] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, 2016. 1

[36] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015. 1

[37] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep metric learning via lifted structured feature embedding. In *CVPR*, 2016. 2

[38] K. Okuma, A. Taleghani, N. De Freitas, J. J. Little, and D. G. Lowe. A boosted particle filter: Multitarget detection and tracking. In *ECCV*. 2004. 2

[39] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *CVPR*, 2011. 5

[40] R. Sanchez-Matilla, F. Poiesi, and A. Cavallaro. Online multi-target tracking with strong and weak detections. In *ECCVW*, 2016. 8

[41] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015. 2

[42] X. Song, J. Cui, H. Zha, and H. Zhao. Vision-based multiple interacting targets tracking via on-line supervised learning. *ECCV*, 2008. 2

[43] Y. Sun, Y. Chen, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. In *NIPS*, 2014. 2

[44] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 1

[45] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, 2014. 2

[46] S. Tang, B. Andres, M. Andriluka, and B. Schiele. Subgraph decomposition for multi-target tracking. In *CVPR*, 2015. 8

[47] R. Tao, E. Gavves, and A. W. Smeulders. Siamese instance search for tracking. In *ICCV*, 2016. 2

[48] B. Wang, L. Wang, B. Shuai, Z. Zuo, T. Liu, K. Luk Chan, and G. Wang. Joint learning of convolutional neural networks and temporally constrained metrics for tracklet association. In *CVPRW*, 2016. 1, 2, 8

[49] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. In *CVPR*, 2014. 2

[50] S. Wang and C. Fowlkes. Learning optimal parameters for multi-target tracking. In *BMVC*, 2015. 1, 2, 5, 8

[51] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015. 2, 6, 7

[52] B. Wu and R. Nevatia. Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. *IJCV*, 75(2):247–266, 2007. 2

[53] Y. Xiang, A. Alahi, and S. Savarese. Learning to track: Online multi-object tracking by decision making. In *ICCV*, 2015. 8

[54] M. Yang and Y. Jia. Temporal dynamic appearance modeling for online multi-person tracking. *CVIU*, 153:16–28, 2016. 8

[55] J. H. Yoon, C.-R. Lee, M.-H. Yang, and K.-J. Yoon. Online multi-object tracking via structural constraint event aggregation. 8

[56] L. Zhang, Y. Li, and R. Nevatia. Global data association for multi-object tracking using network flows. In *CVPR*, 2008. 1, 2

[57] X. Zhang, F. Zhou, Y. Lin, and S. Zhang. Embedding label structures for fine-grained feature representation. *CVPR*, 2016. 2