How To Train Your Deep Multi-Object Tracker

Yihong Xu, Aljosa Osep, Yutong Ban, Radu Horaud, Laura Leal-Taixé, Xavier Alameda-Pineda

How To Train Your Deep Multi-Object Tracker

Yihong Xu[1] Aljoša Ošep[2] Yutong Ban[1,3] Radu Horaud[1]
Laura Leal-Taixé[2] Xavier Alameda-Pineda[1]
[1]Inria, LJK, Univ. Grenoble Alpes, France [2]Technical University of Munich, Germany
[3]Distributed Robotics Lab, CSAIL, MIT, USA
[1]{firstname.lastname}@inria.fr [2]{aljosa.osep, leal.taixe}@tum.de [3]yban@csail.mit.edu

Abstract

The recent trend in vision-based multi-object tracking (MOT) is heading towards leveraging the representational power of deep learning to jointly learn to detect and track objects. However, existing methods train only certain sub-modules using loss functions that often do not correlate with established tracking evaluation measures such as Multi-Object Tracking Accuracy (MOTA) and Precision (MOTP). As these measures are not differentiable, the choice of appropriate loss functions for end-to-end training of multi-object tracking methods is still an open research problem. In this paper, we bridge this gap by proposing a differentiable proxy of MOTA and MOTP, which we combine in a loss function suitable for end-to-end training of deep multi-object trackers. As a key ingredient, we propose a Deep Hungarian Net (DHN) module that approximates the Hungarian matching algorithm. DHN allows estimating the correspondence between object tracks and ground truth objects to compute differentiable proxies of MOTA and MOTP, which are in turn used to optimize deep trackers directly. We experimentally demonstrate that the proposed differentiable framework improves the performance of existing multi-object trackers, and we establish a new state of the art on the MOTChallenge benchmark. Our code is publicly available from https://github.com/yihongXU/deepMOT.

1. Introduction

Vision-based multi-object tracking (MOT) is a long-standing research problem with applications in mobile robotics and autonomous driving. It is through tracking that we become aware of surrounding object instances and anticipate their future motion. The majority of existing methods for pedestrian tracking follow the tracking-by-detection paradigm and mainly focus on the association of detector responses over time. A significant amount of research investigated combinatorial optimization techniques for this challenging data association problem [42, 41, 48, 58, 8, 7].

Recent data-driven trends in MOT leverage the representational power of deep networks for learning identity-preserving embeddings for data association [28, 50, 54], learning the appearance model of individual targets [14, 59] and learning to regress the pose of the detected targets [4]. However, these methods train individual parts of the MOT pipeline using proxy losses (*e.g.* triplet loss [47] for learning identity embeddings), that are only indirectly related to the MOT evaluation measures [6]. The main difficulty in defining loss functions that resemble standard tracking evaluation measures is due to the need of computing the optimal matching between the predicted object tracks and the ground-truth objects. This problem is usually solved by using the Hungarian (Munkres) algorithm (HA) [27], which contains non-differentiable operations.

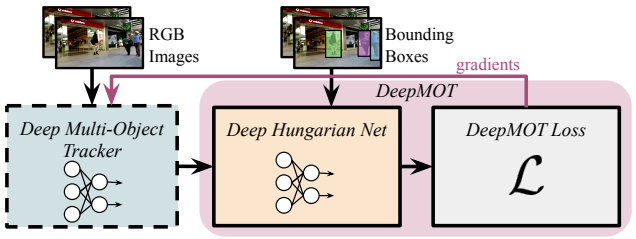

Figure 1. We propose DeepMOT, a general framework for training deep multi-object trackers including the DeepMOT loss that directly correlates with established tracking evaluation measures [6]. The key component in our method is the Deep Hungarian Net (DHN) that provides a soft approximation of the optimal prediction-to-ground-truth assignment, and allows to deliver the gradient, back-propagated from the approximated tracking performance measures, needed to update the tracker weights.

The significant contribution of this paper is a novel, differentiable framework for the training of multi-object trackers (Fig. 1): it proposes a differentiable variant of the standard CLEAR-MOT [6] evaluation measures, which we combine into a novel loss function, suitable for end-to-end training of MOT methods. In particular, we introduce a differentiable network module – Deep Hungarian Net (DHN) – that approximates the HA and provides a soft approximation of the optimal prediction-to-ground-truth assignment. The proposed approximation is based on a bi-directional recurrent neural network (Bi-RNN) that computes the (soft) assignment matrix based on the prediction-

to-ground-truth distance matrix. We then express both the MOTA and MOTP [6] as differentiable functions of the computed (soft) assignment matrix and the distance matrix. Through DHN, the gradients from the approximated tracking performance measures are back-propagated to update the tracker weights. In this way, we can train object trackers in a data-driven fashion using losses that directly correlate with standard MOT evaluation measures. In summary, this paper makes the following contributions:

(i) We propose novel loss functions that are directly inspired by standard MOT evaluation measures [6] for end-to-end training of multi-object trackers.

(ii) In order to back-propagate losses through the network, we propose a new network module – Deep Hungarian Net – that learns to match predicted tracks to ground-truth objects in a differentiable manner.

(iii) We demonstrate the merit of the proposed loss functions and differentiable matching module by training the recently published Tracktor [4] using our proposed framework. We demonstrate improvements over the baseline and establish a new state-of-the-art result on MOTChallenge benchmark datasets [37, 30].

2. Related Work

Tracking as Discrete Optimization. With the emergence of reliable object detectors [15, 18, 31] tracking-by-detection has become the leading tracking paradigm. These methods first perform object detection in each image and associate detections over time, which can be performed online via frame-to-frame bi-partite matching between tracks and detections [27]. As early detectors were noisy and unreliable, several methods search for the optimal association in an offline or batch fashion, often posed as a network flow optimization problem [41, 48, 58, 8, 7].

Alternatively, tracking can be posed as a maximum-a-posteriori (MAP) estimation problem by seeking an optimal set of tracks as a conditional distribution of sequential track states. Several methods perform inference using conditional random fields (CRFs) [38, 12, 40], Markov chain Monte Carlo (MCMC) [39] or a variational expectation-maximization [1, 2, 3]. These methods in general, use hand-crafted descriptors for the appearance model, such as color histograms [38, 11], optical flow based descriptors [12] and/or motion models [31, 40] as association cues. Therefore typically only a few parameters are trainable and are commonly learned using grid/random search or tree of parzen window estimators [5, 40]. In the case of CRF-based methods, the weights can be trained using structured SVM [53, 55].

Deep Multi-Object Tracking. Recent data-driven trends in MOT leverage representational power of deep neural networks. Xiang *et al*. [56] learn track birth/death/association policy by modeling them as Markov Decision Processes (MDP). As the standard evaluation measures [6] are not differentiable, they learn the policy by reinforcement learning.

Several existing methods train parts of their tracking methods using losses, not directly related to tracking evaluation measures [6]. Kim *et al*. [25] leverages pre-learned CNN features or a bilinear LSTM [26] to learn the long-term appearance model. Both are incorporated into (Multiple Hypothesis Tracking) MHT framework [42]. Other methods [19, 28, 54, 50] learn identity-preserving embeddings for data association using deep neural networks, trained using contrastive [20], triplet [47] or quadruplet loss [50]. At inference time, these are used for computing data association affinities. Approaches by [14, 59] learn the appearance model of individual targets using an ensemble of single-object trackers that share a convolutional backbone. A spatiotemporal mechanism (learned online using a cross-entropy loss) guides the online appearance adaptation and prevents drifts. All these methods are only partially trained, and sometimes in various stages. Moreover, it is unclear how to train these methods to maximize established tracking metrics.

Most similar to our objective, Wang *et al*. [55] propose a framework for learning parameters of linear cost association functions, suitable for network flow optimization [58] based multi-object trackers. They train parameters using structured SVM. Similar to our method, they devise a loss function, that resembles MOTA: the intra-frame loss penalizes false positives (FP) and missed targets while the inter-frame component of the loss penalizes false associations, ID switches, and missed associations. However, their loss is not differentiable and is only suitable for training parameters within the proposed min-cost flow framework. Chu *et al*. [13] propose an end-to-end training framework that jointly learns feature, affinity and multi-dimensional assignment. However, their losses are not directly based on MOTA and MOTP. Schulter *et al*. [48] parameterize (arbitrary) cost functions with neural networks and train them end-to-end by optimizing them with respect to the min-flow training objective. Different from [48], our approach goes beyond learning the association function, and can be used by any learnable tracking method.

Bergmann *et al*. [4] propose a tracking-by-regression approach to MOT. The method is trained for the object detection task using a smooth L_1 loss for the bounding box regressor. Empirically, their method is able to regress bounding boxes in high-frame rate video sequences with no significant camera motion. Apart from the track birth and death management, this approach is fully trainable, and thus it is a perfect method for demonstrating the merit of our training framework. Training this approach on a sequence-level data using our proposed loss further improves the performance and establishes a new state of the art on the MOTChallenge benchmark [30].

3. Overview and Notation

The objective of any MOT method is to predict tracks in a video sequence. Each track $\mathbf{X}^i$ is associated with an identity i, and consists of L_i image bounding boxes $\mathbf{x}_{t_l}^i \in \mathbb{R}^4$ (2D location and size), $l = 1 \ldots, L_i$. The task of a multi-object tracker is to accurately estimate the bounding boxes for all identities through time.

At evaluation time, the standard metrics operate frame-by-frame. At frame t, the N_t predicted bounding boxes, $\mathbf{x}_t^{i_1}, \ldots, \mathbf{x}_t^{i_{N_t}}$ must be compared to the M_t ground-truth objects, $\mathbf{y}_t^{j_1}, \ldots, \mathbf{y}_t^{j_{M_t}}$. We first need to compute the correspondence between predicted bounding boxes and ground-truth objects. This is a non-trivial problem as multiple ground-truth boxes may overlap and thus can fit to several track hypotheses. In the following we will omit temporal index t to ease the reading. All expressions will be evaluated with respect to time index t unless specified otherwise.

The standard metrics, proposed in [6], tackle this association problem using bi-partite matching. First, a prediction-to-ground-truth distance matrix $\mathbf{D} \in \mathbb{R}^{N \times M}$,[1] $d_{nm} \in [0, 1]$ is computed. For vision-based tracking, an intersection-over-union (IoU) based distance is commonly used. Then, the optimal prediction-to-ground-truth assignment binary matrix is obtained by solving the following integer program using the Hungarian algorithm (HA) [27]:

$$\mathbf{A}^* = \underset{\mathbf{A} \in \{0,1\}^{N \times M}}{\operatorname{argmin}} \sum_{n,m} d_{nm} a_{nm}, \quad \text{s.t.} \sum_m a_{nm} \leq 1, \forall n;$$
$$\sum_n a_{nm} \leq 1, \forall m; \quad \sum_{m,n} a_{nm} = \min\{N, M\}.$$

By solving this integer program we obtain a mutually consistent association between ground-truth objects and track predictions. The constraints ensure that all rows and columns of the assignment should sum to 1, thus avoiding multiple assignments between the two sets. After finding the optimal association, $\mathbf{A}^*$, we can compute the MOTA and MOTP measures using $\mathbf{A}^*$ and $\mathbf{D}$:[2]

$$\text{MOTA} = 1 - \frac{\sum_t (\text{FP}_t + \text{FN}_t + \text{IDS}_t)}{\sum_t M_t}, \tag{1}$$

$$\text{MOTP} = \frac{\sum_t \sum_{n,m} d_{tnm} a^*_{tnm}}{\sum_t |\,\text{TP}_t\,|}, \tag{2}$$

where a^*_{tnm} is the (n, m)-th entry of $\mathbf{A}^*$ at time t. The true positives (TP) correspond to the number of matched predicted tracks and false positives (FP) correspond to the number of non-matched predicted tracks. False negatives (FN) denote the number of ground-truth objects without a match. Finally, to compute ID switches (IDS) we need to keep track of past-frame assignments. Whenever the track assigned to a ground truth object changes, we increase the number of IDS and update the assignment structure.

As these evaluation measures are not differentiable, existing strategies only optimize the trackers' hyper-parameters (using, *e.g.* random or grid search) that maximize MOTA or MOTP or a combination of both. In their current version, MOTA and MOTP cannot be directly used for tracker optimization with gradient descent techniques.

[1] The distance matrix $\mathbf{D}$ is considered without those objects/tracks that are thresholded-out, *i.e.*, too far from any possible assignment.

[2] Accounting also for the objects/tracks that were left out.

4. DeepMOT

The first step to compute the CLEAR-MOT [6] tracking evaluation measures is to perform bi-partite matching between the sets of ground-truth objects and of predicted tracks. Once the correspondence between the two sets is established, we can count the number of TP, FN, and IDS needed to express MOTA and MOTP. As the main contribution of this paper, we propose a differentiable loss inspired by these measures, following the same two-step strategy. We first propose to perform a soft matching between the two sets using a differentiable function, parameterized as a deep neural network. Once we establish the matching, we design a loss, approximating the CLEAR-MOT measures, as a combination of differentiable functions of the (soft) assignment matrix and the distance matrix. Alternative measures such as IDF1 [44] focus on how long the tracker correctly identifies targets instead of how often mismatches occur. However, MOTA and IDF1 have a strong correlation. This is reflected in our results – by optimizing our loss, we also improve the IDF1 measure (see Sec. 5.3). In the following, we discuss both the differentiable matching module (Sec. 4.1) and the differentiable version of the CLEAR-MOT measures [6] (Sec. 4.2).

4.1. Deep Hungarian Net: DHN

In this section, we introduce DHN, a fundamental block in our DeepMOT framework. DHN produces a proxy $\tilde{\mathbf{A}}$ that is differentiable w.r.t. $\mathbf{D}$. Thus DHN provides a bridge to deliver gradient from the loss (to be described later on) to the tracker. We formalize DHN with a non-linear mapping that inputs $\mathbf{D}$ and outputs the proxy soft assignment matrix $\tilde{\mathbf{A}}$. DHN is modeled by a neural network $\tilde{\mathbf{A}} = g(\mathbf{D}, \omega_d)$ with parameters ω_d. Importantly, the DHN mapping must satisfy several properties: (i) the output $\tilde{\mathbf{A}}$ must be a good approximation to the optimal assignment matrix $\mathbf{A}^*$, (ii) this approximation must be differentiable w.r.t. $\mathbf{D}$, (iii) both input and output matrix are of equal, but varying size and (iv) g must take global decisions as the HA does.

While (i) will be achieved by setting an appropriate loss function when training the DHN (see Sec. 5.1), (ii) is ensured by designing DHN as a composite of differentiable functions. The requirements (iii) and (iv) push us to design a network that can process variable (but equal) input and output sizes, where every output neuron has a receptive field

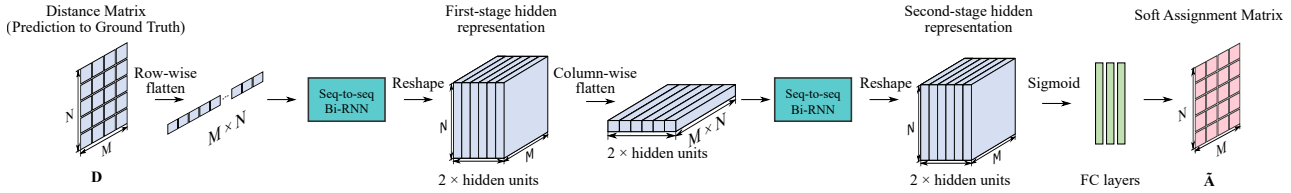

Figure 2. Structure of the proposed Deep Hungarian Net. The row-wise and column-wise flattening are inspired by the original Hungarian algorithm, while the Bi-RNN allows for all decisions to be taken globally, thus is accounting for all input entries.

equals to the entire input. We opt for bi-directional recurrent neural networks (Bi-RNNs). Alternatively, one could consider the use of fully convolutional networks, as these would be able to process variable input/output sizes. However, large assignment problems would lead to partial receptive fields, and therefore, to local assignment decisions.

We outline our proposed architecture in Fig. 2. In order to process a 2D distance matrix $\mathbf{D}$ using RNNs, we perform row-wise (column-wise) flattening of $\mathbf{D}$. This is inspired by the original HA that performs sequentially row-wise and column-wise reductions and zero-entry verifications and fed it to Bi-RNNs (see details below), opening the possibility for $g(\cdot)$ to make global assignment decisions.

Precisely, we perform flattening sequentially, *i.e.*, first row-wise followed by column-wise. The row-wise flattened $\mathbf{D}$ is input to a first Bi-RNN that outputs the first-stage hidden representation of size $N \times M \times 2h$, where h is the size of the Bi-RNN hidden layers. Intuitively the first-stage hidden representation encodes the row-wise intermediate assignments. We then flatten the first-stage hidden representation column-wise, to input to a second Bi-RNN that produces the second-stage hidden representation of size $N \times M \times 2h$. The two Bi-RNNs have the same hidden size, but they do not share weights. Intuitively, the second-stage hidden representation encodes the final assignments. To translate these encodings into the final assignments, we feed the second-stage hidden representation through three fully-connected layers (along the $2h$ dimension, *i.e.*, independently for each element of the original $\mathbf{D}$). Finally, a sigmoid activation produces the optimal $N \times M$ soft-assignment matrix $\tilde{\mathbf{A}}$. Note that in contrast to the binary output of the Hungarian algorithm, DHN outputs a (soft) assignment matrix $\tilde{\mathbf{A}} \in [0,1]^{N \times M}$.

Distance Matrix Computation. The most common metric for measuring the similarity between two bounding boxes is the Intersection-over-Union (IoU). Note that, in principle, the input $\mathbf{D}$ can be any (differentiable) distance function. However, if two bounding boxes have no intersection, the distance $1-\text{IoU}$ will always be a constant value of 1. In that case, the gradient from the loss will be 0, and no information will be back-propagated. For this reason, our distance is an average of the Euclidean center-point distance and the Jaccard distance $\mathcal{J}$ (defined as $1 - \text{IoU}$):

$$d_{nm} = \frac{f(\mathbf{x}^n, \mathbf{y}^m) + \mathcal{J}(\mathbf{x}^n, \mathbf{y}^m)}{2}. \tag{3}$$

f is the Euclidean distance normalized w.r.t. the image size:

$$f(\mathbf{x}^n, \mathbf{y}^m) = \frac{\|c(\mathbf{x}^n) - c(\mathbf{y}^m)\|_2}{\sqrt{H^2 + W^2}}, \tag{4}$$

where function $c(\cdot)$ computes the center point of the bounding box and H and W are the height and the width of the video frame, respectively. Both the normalized Euclidean distance and Jaccard distance have values in the range of $[0, 1]$, so do all entries d_{nm}. Our framework admits any distance that is expressed as a composition of differentiable distance functions. In the experimental section, we demonstrate the benefits of adding a term that measures the cosine distance between two learned appearance embeddings. In the following, we explain how we compute a differentiable proxy of MOTA and MOTP as functions of $\mathbf{D}$ and $\tilde{\mathbf{A}}$.

4.2. Differentiable MOTA and MOTP

In this section, we detail the computation of two components of the proposed DeepMOT loss: differentiable MOTA ($dMOTA$) and MOTP ($dMOTP$). As discussed in Sec. 3, to compute the classic MOTA and MOTP evaluation measures, we first find the optimal matching between predicted tracks and ground-truth objects. Based on $\mathbf{A}^*$, we count FN, FP and IDS. The latter is computed by comparing assignments between the current frame and previous frames. To compute the proposed $dMOTA$ and $dMOTP$, we need to express all these as differentiable functions of $\mathbf{D}$ and $\tilde{\mathbf{A}}$ computed using DHN (see Sec. 4.1).

The operations described in the following are illustrated in Fig. 3. First, we need to count FN and FP. Therefore, we need to obtain a count of non-matched tracks and non-matched ground-truth objects. To this end, we first construct a matrix $\mathbf{C}^r$ by appending a column to $\tilde{\mathbf{A}}$, filled with a threshold value δ (*e.g.*, $\delta = 0.5$), and perform row-wise softmax (Fig. 3a). Analogously, we construct $\mathbf{C}^c$ by appending a row to $\tilde{\mathbf{A}}$ and perform column-wise softmax (Fig. 3b). Then, we can express a soft approximation of the number of FP and FN as:

$$\tilde{\text{FP}} = \sum_n \mathbf{C}^r_{n,M+1}, \quad \tilde{\text{FN}} = \sum_m \mathbf{C}^c_{N+1,m}. \tag{5}$$

Intuitively, if all elements in $\tilde{\mathbf{A}}$ are smaller than the threshold δ, then entries of $\mathbf{C}^r_{n,M+1}$ and $\mathbf{C}^c_{N+1,m}$ will be close to 1, signaling we have a FP or FN. Otherwise, the element with the largest value in each row/column of $\mathbf{C}^r$ and

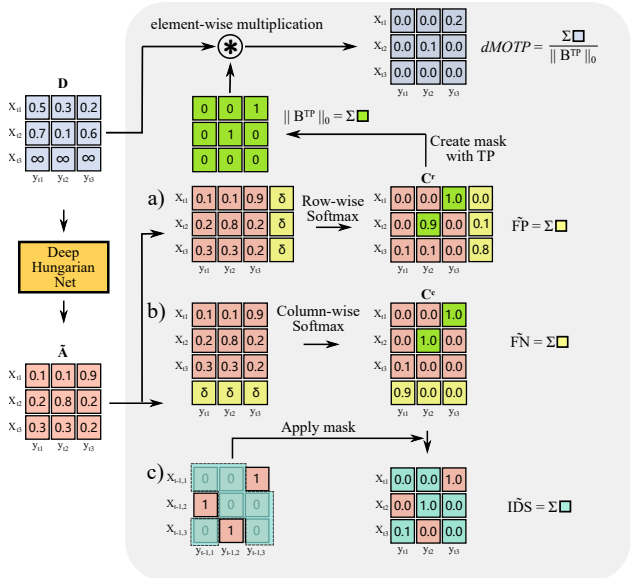

Figure 3. DeepMOT loss: $dMOTP$ (top) is computed as the average distance of matched tracks and $dMOTA$ (bottom) is composed with $\tilde{\text{FP}}$, $\tilde{\text{IDS}}$ and $\tilde{\text{FN}}$.

$\mathbf{C}^c$ (respectively) will be close to 1, signaling that we have a match. Therefore, the sum of the $N+1$-th row of $\mathbf{C}^c$ (Fig. 3b) and of the $M+1$-th column of $\mathbf{C}^r$ (Fig. 3a) provide an soft estimate of the number of FN and the number of FP, respectively. We will refer to these as $\tilde{\text{FN}}$ and $\tilde{\text{FP}}$.

To compute the soft approximations $\tilde{\text{IDS}}$ and $dMOTP$, we additionally need to construct two binary matrices $\mathbf{B}^{\text{TP}}$ and $\mathbf{B}^{\text{TP}}_{\text{-1}}$, whose non-zero entries signal true positives at the current and previous frames respectively. Row indices of these matrices correspond to indices assigned to our tracks and column indices correspond to ground truth object identities. We need to pad $\mathbf{B}^{\text{TP}}_{\text{-1}}$ for element-wise multiplication because the number of tracks and objects varies from frame-to-frame. We do this by filling-in rows and columns of $\mathbf{B}^{\text{TP}}_{\text{-1}}$ to adapt the matrix size for the newly-appeared objects at the current frame by copying their corresponding rows and columns from $\mathbf{B}^{\text{TP}}$. Note that we do not need to modify $\mathbf{B}^{\text{TP}}$ to compensate for newly appearing objects as these do not cause IDS. By such construction, the sum of $\mathbf{C}^c_{1:N,1:M} \odot \overline{\mathbf{B}}^{\text{TP}}_{\text{-1}}$ (where $\overline{\mathbf{B}}$ is the binary complement of $\mathbf{B}$) yields the (approximated) number of IDS (Fig. 3c):

$$\tilde{\text{IDS}} = \| \mathbf{C}^c_{1:N,1:M} \odot \overline{\mathbf{B}}^{\text{TP}}_{\text{-1}} \|_1, \tag{6}$$

where $\|\cdot\|_1$ is the L_1 norm of a flattened matrix. With these ingredients, we can evaluate $dMOTA$:

$$dMOTA = 1 - \frac{\tilde{\text{FP}} + \tilde{\text{FN}} + \gamma\, \tilde{\text{IDS}}}{M}. \tag{7}$$

γ controls the penalty we assign to $\tilde{\text{IDS}}$. Similarly, we can express $dMOTP$ as:

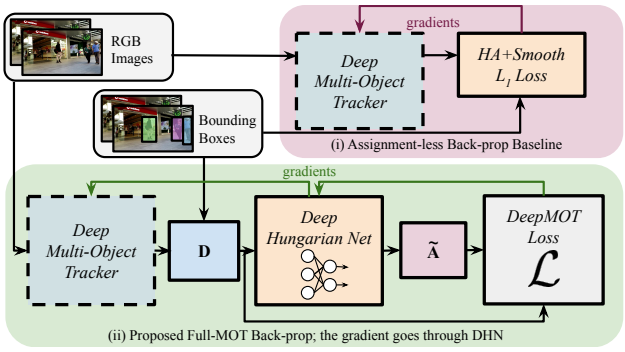

Figure 4. The proposed MOT training strategy (bottom) accounts for the track-to-object assignment problem, solved by the proposed DHN, and approximates the standard MOT losses, as opposed to the classical training strategies (top) using the *non-differentiable* HA.

$$dMOTP = 1 - \frac{\| \mathbf{D} \odot \mathbf{B}^{\text{TP}} \|_1}{\| \mathbf{B}^{\text{TP}} \|_0}. \tag{8}$$

Intuitively, the L_1 norm expresses the distance between the matched tracks and ground-truth objects, and the zero-norm $\|\cdot\|_0$ counts the number of matches. Since we should train the tracker to maximize MOTA and MOTP, we propose the following DeepMOT loss:

$$\mathcal{L}_{\text{DeepMOT}} = (1 - dMOTA) + \lambda(1 - dMOTP), \tag{9}$$

where λ is a loss balancing factor. By minimizing our proposed loss function $\mathcal{L}_{\text{DeepMOT}}$, we are penalizing FP, FN and IDS – all used by the CLEAR-MOT measures [6]. Same as for the standard CLEAR-MOT measures, $dMOTA$, $dMOTP$ must be computed at every time frame t.

4.3. How To Train Your Deep Multi-Object Tracker

The overall tracker training procedure is shown in Fig. 4. We randomly sample a pair of consecutive frames from the training video sequences. These two images together with their ground-truth bounding boxes constitute one training instance. For each such instance, we first initialize the tracks with ground-truth bounding boxes (at time t) and run the forward pass to obtain the track's bounding-box predictions in the following video frame (time $t+1$). To mimic the effect of imperfect detections, we add random perturbations to the ground-truth bounding boxes (see supplementary material for details). We then compute $\mathbf{D}$ and use our proposed DHN to compute $\tilde{\mathbf{A}}$ (Sec. 4.1). Finally, we compute our proxy loss based on $\mathbf{D}$ and $\tilde{\mathbf{A}}$ (Sec. 4.2). This provides us with a gradient that accounts for the assignment, and that is used to update the weights of the tracker.

5. Experimental Evaluation

In this section, we first experimentally verify that our proposed DHN is a good approximation to HA [27] for bi-

partite matching, as required by MOT evaluation measures (Sec. 5.1). To show the merit of the proposed framework, we conduct several experiments on several datasets for evaluating pedestrian tracking performance (Sec. 5.2).

5.1. DHN Implementation Details

In this section, we provide insights into the performance of our differentiable matching module and outline the training and evaluation details.

DHN Training. To train the DHN, we create a data set with pairs of matrices ($\mathbf{D}$ and $\mathbf{A}^*$), separated into 114,483 matrices for training and 17,880 for matrices testing. We generate distance matrices $\mathbf{D}$ using ground-truth bounding boxes and public detections, provided by the MOT challenge datasets [37, 30]. We generate the corresponding assignment matrices $\mathbf{A}^*$ (as labels for training) using HA described in [6]. We pose the DHN training as a 2D binary classification task using the focal loss [33]. We compensate for the class imbalance (between the number of zeros n_0 and ones n_1 in $\mathbf{A}^*$) by weighting the dominant zero-class using $w_0 = n_1/(n_0 + n_1)$. We weight the one-class by $w_1 = 1 - w_0$. We evaluate the performance of DHN by computing the weighted accuracy (WA):

$$\text{WA} = \frac{w_1 n_1^* + w_0 n_0^*}{w_1 n_1 + w_0 n_0}, \tag{10}$$

where n_1^* and n_0^* are the number of true and false positives, respectively. Since the output of the DHN are between 0 and 1, we threshold the output at 0.5. Under these conditions, the network in Fig. 2 scores a WA of 92.88%. In the supplementary material, we provide (i) an ablation study on the choice of recurrent unit, (ii) a discussion of alternative architectures, (iii) an analysis of the impact of the distance matrix size on the matching precision and (iv) we experimentally assess how well the DHN preserves the properties of assignment matrices.

DHN Usage. Once the DHN is trained with the strategy described above, its weights are fixed: they are not updated in any way during the training of the deep trackers.

5.2. Experimental Settings

We demonstrate the practical interest of the proposed framework by assessing the performance of existing (deep) multi-object trackers when trained using the proposed framework on several datasets for pedestrian tracking. We first ablate the loss terms and the tracking architectures. We also evaluate the impact of the framework with respect to other training alternatives. Finally, we establish a new state-of-the-art score on the MOTChallenge benchmark.

Datasets and Evaluation Metrics. We use the MOT15, MOT16, and MOT17 datasets, which provide crowded pedestrian video sequences captured in the real-world outdoor and indoor scenarios. For the ablation study, we divide the training sequences into training and validation. The details of the split can be found in the supplementary material. In addition to the standard MOTP and MOTA measures [6], we report the performance using the IDF1 [44] measure, defined as the ratio of correctly identified detections over the average number of ground-truth objects and object tracks. We also report mostly tracked (MT) and mostly lost (ML) targets, defined as the ratio of ground-truth trajectories that are covered by a track hypothesis more than 80% and less than 20% of their life span respectively.

Tracktor. Tracktor [4] is an adaptation of the Faster RCNN [43] object detector to the MOT task. It uses a region proposal network (RPN) and the classification/regression heads of the detector to (i) detect objects and (ii) to follow the detected targets in the consecutive frames using a bounding box regression head. As most parts of Tracktor are trainable, this makes this method a perfect candidate to demonstrate the benefits of our framework. Note that Tracktor was originally trained only on the MOTChallenge detection dataset and was only applied to video sequences during inference. In the following, we will refer to Tracktor trained in this setting as **Vanilla Base** Tracktor. Thanks to DeepMOT, we can train Tracktor directly on video sequences, optimizing for standard MOT measures. We will refer to this variant as **DeepMOT Base** Tracktor.

Tracktor+ReID. Vanilla Tracktor has no notion of track identity. Therefore [4] proposed to use an externally trained ReID module during inference to mitigate IDS. This external ReID module is a feature extractor with a ResNet-50 backbone, trained using a triplet loss [47] on the MOTChallenge video sequences. We will refer to this variant as **+ReIDext**. Note that this does not give Tracktor any notion of identity during training. This means that the DeepMOT loss which penalizes the number of IDS will have no significant effect on the final performance. For this reason, we propose to replace **ReIDext** with a lightweight ReID head that we can train jointly with Tracktor using DeepMOT. This in turn allows us to utilize $\widetilde{\text{IDS}}$ and to fully optimize performance to all components of CLEAR-MOT measures. We refer to this variant as **+ReIDhead**. It takes the form of a fully-connected layer with 128 units plugged into Tracktor. In the supplementary material we provide details on how we embed the ID information into the distance matrix $\mathbf{D}$.

Even if such a network head has been previously used in [54], it was trained externally using the triplet loss [47]. To the best of our knowledge, we are the first to optimize such an appearance model by directly optimizing the whole network for tracking evaluation measures.

MOT-by-SOT. To demonstrate the generality of our method, we propose two additional simple trainable base-

	Method	MOTA ↑	MOTP ↑	IDF1 ↑	MT ↑	ML ↓	FP ↓	FN ↓	IDS ↓
Van.	Base	59.97	89.50	70.84	35.13	27.66	276	31827	326
	+ReIDext	**60.20**	**89.50**	**71.15**	**35.13**	**27.80**	**276**	**31827**	**152**
DeepMOT	Base	60.43	91.82	71.44	35.41	27.25	218	31545	309
	+ReIDext	60.62	91.82	71.66	35.41	27.39	218	31545	149
	+ReIDhead	**60.66**	**91.82**	**72.32**	**35.41**	**27.25**	**218**	**31545**	**118**

Table 1. Impact of the different ReID strategies for the two training strategies on Tracktor's performance.

Training loss	MOTA ↑	MOTP ↑	IDF1 ↑	MT ↑	ML ↓	FP ↓	FN ↓	IDS ↓
Vanilla	60.20	89.50	71.15	35.13	27.80	276	31827	152
Smooth L_1	60.38	91.81	71.27	34.99	27.25	294	31649	164
$dMOTP$	60.51	91.74	71.75	35.41	**26.83**	291	31574	142
$dMOTA$	60.52	88.31	71.92	35.41	27.39	254	31597	142
$dMOTA$+$dMOTP$-IDS	60.61	**92.03**	72.10	35.41	27.25	222	31579	124
$dMOTA$+$dMOTP$	**60.66**	91.82	**72.32**	**35.41**	27.25	**218**	**31545**	**118**

Table 2. Ablation study on the effect the training loss on Tracktor.

lines to perform MOT by leveraging two existing off-the-shelf (trainable) single-object trackers (SOTs): GOTURN [21] and SiamRPN [32]. During inference we initialize and terminate tracks based on object detections. For each object, the SOTs take a reference image at time $t-1$ of the person and a search region in image t as input. Based on this reference box and search region, the SOTs then regress a bounding box for each object independently.

Track Management. In all cases, we use a simple (non-trainable) track management procedure. We (i) use detector responses to initialize object tracks in regions, not covered by existing tracks (can be either public detections or Faster RCNN detection responses in the case of Tracktor); (ii) we regress tracks from frame $t-1$ to frame t using either a SOT or Tracktor and (iii) we terminate tracks that have no overlap with detections (SOT baseline) or invoke the classification head of Tracktor, that signals whether a track is covering an object or not. As an alternative to direct termination, we can set a track as invisible for K frames.

5.3. Results and Discussion

Beyond Bounding Box Regression. In Tab. 1, we first establish the Vanilla Base Tracktor performance on our validation set and compare it to the DeepMOT Base Tracktor. This experiment (i) validates that our proposed training pipeline based on DHN delivers the gradient to the trackers and improves the overall performance, and (ii) confirms our intuition that training object trackers using a loss that directly correlates with the tracking evaluation measures has a positive impact. Note that the impact on IDS is minimal, which may be on the first sight surprising, as our proposed loss penalizes IDS in addition to FP, FN, and bounding box misalignment.

We study this by first evaluating the impact of applying external ReID module, *i.e.*, **+ReIDext**. As can be seen in Tab. 1, **ReIDext** has a positive impact on the performance, as expected, in terms of MOTA ($+0.23\%$ and $+0.19\%$) and IDS (-174 and -160) compared to **Base** for **Vanilla** and **DeepMOT** training respectively. To further demonstrate the interest of a ReID module, we also report the **+ReIDhead** architecture trained with DeepMOT. Importantly, **+ReIDhead** cannot be trained in the Vanilla setting due to the lack of mechanisms to penalize IDS. Remarkably, **+ReIDhead** trained end-to-end with Tracktor does not only improve over the Base performance (MOTA $+0.23\%$, IDS $\downarrow 191$), but it also outperforms **+ReIDext** (MOTA $\uparrow 0.04$ and IDS $\downarrow 31$). Very importantly, the lightweight ReID head contains a significantly lower number of parameters (≈ 131 K) compared to the external ReID module (≈ 25 M).

Finally, in addition to improve the performance measures for which we optimize Tracktor, DeepMOT consistently improves tracking measures such as IDF1 ($\uparrow 1.17$ improvement of **DeepMOT+ReIDhead** over **Vanilla+ReIDext**). We conclude that (i) training existing trackers using our proposed loss clearly improves the performance and (ii) we can easily extend existing trackers such as Tracktor to go beyond simple bounding box regression and incorporate the appearance module directly into the network. All modules are optimized jointly in a single training.

DeepMOT Loss Ablation. Next, we perform several experiments in which we study the impact of different components of our proposed loss (Eq. 9) to the performance of Tracktor (**DeepMOT+ReIDhead**). We outline our results in Tab. 2. In addition to **Vanilla+ReIDext** (representing the best performance trained in Vanilla settings), we also report results obtained by training the same architecture using only the Smooth L_1 loss (see Fig. 4). We train the regression head with Smooth L_1 loss using a similar training procedure as for DeepMOT (see Sec. 4.3), to regress predicted bounding boxes to the ones at current time step of their associated tracks. This approach is limited in the sense that we cannot (directly) penalize FP, FN and IDS.

The Smooth L_1 training, when compared to Vanilla, has a positive impact on almost all performance measures, except for MT, FP, and IDS. However, both Vanilla and Smooth L_1 are outperformed almost systematically for all performance measures by the various variants of the DeepMOT loss. Remarkably, when using $dMOTA$ term in our loss, we significantly reduce the number of IDS and FP. Training with $dMOTP$ has the highest impact on MOTP, as it is the case when training with Smooth L_1. When only optimizing for $dMOTA$, we have a higher impact on the MOTA and IDF1 measure. Remarkably, when training with ($dMOTA$+$dMOTP$), we obtain a consistent improvement on all tracking evaluation measures with respect to Vanilla and Smooth L_1. Finally, we asses the impact of $\tilde{\text{IDS}}$, by setting the weight γ to 0 (Eq. 7) (line $dMOTA$+$dMOTP$-$\tilde{\text{IDS}}$). In this settings, the trackers exhibits a higher number of IDS compared to using the full loss, confirming that the latter is the best strategy.

	Training	MOTA ↑	MOTP ↑	IDF1 ↑	MT ↑	ML ↓	FP ↓	FN ↓	IDS ↓
GOTURN	Pre-trained	45.99	85.87	49.83	22.27	36.51	2927	39271	1577
	Smooth L_1	52.28	90.56	63.53	**29.46**	**34.58**	2026	36180	472
	DeepMOT	**54.09**	**90.95**	**66.09**	28.63	35.13	**927**	**36019**	**261**
SiamRPN	Pre-trained	55.35	87.15	66.95	33.61	**31.81**	1907	33925	356
	Smooth L_1	56.51	**90.88**	68.38	**33.75**	32.64	925	34151	167
	DeepMOT	**57.16**	89.32	**69.49**	33.47	32.78	**889**	**33667**	**161**
Tracktor	Vanilla	60.20	89.50	71.15	35.13	27.80	276	31827	152
	Smooth L_1	60.38	91.81	71.27	34.99	27.25	294	31649	164
	DeepMOT	**60.66**	**91.82**	**72.32**	**35.41**	**27.25**	**218**	**31545**	**118**

Table 3. DeepMOT vs. Smooth L_1 using MOT-by-SOT baselines and Tracktor.

	Method	MOTA ↑	MOTP ↑	IDF1 ↑	MT ↑	ML ↓	FP ↓	FN ↓	IDS ↓
MOT17	DeepMOT-Tracktor	**53.7**	77.2	53.8	19.4	36.6	**11731**	247447	1947
	Tracktor [4]	53.5	78.0	52.3	19.5	36.6	12201	248047	2072
	DeepMOT-SiamRPN	52.1	**78.1**	47.7	16.7	41.7	12132	255743	2271
	SiamRPN [32]	47.8	76.4	41.4	17.0	41.7	38279	251989	4325
	DeepMOT-GOTURN	48.1	77.9	40.0	13.6	43.5	22497	266515	3792
	GOTURN [21]	38.3	75.1	25.7	9.4	47.1	55381	282670	10328
	eHAF [49]	51.8	77.0	**54.7**	**23.4**	37.9	33212	**236772**	1834
	FWT [22]	51.3	77.0	47.6	21.4	**35.2**	24101	247921	2648
	jCC [24]	51.2	75.9	54.5	20.9	37.0	25937	247822	**1802**
	MOTDT17 [34]	50.9	76.6	52.7	17.5	35.7	24069	250768	2474
	MHT_DAM [25]	50.7	77.5	47.2	20.8	36.9	22875	252889	2314
MOT16	DeepMOT-Tracktor	**54.8**	77.5	**53.4**	**19.1**	**37.0**	**2955**	**78765**	645
	Tracktor [4]	54.4	78.2	52.5	19.0	36.9	3280	79149	682
	DeepMOT-SiamRPN	51.8	78.1	45.5	16.1	45.1	3576	83699	641
	SiamRPN [32]	44.0	76.6	36.6	15.5	45.7	18784	82318	1047
	DeepMOT-GOTURN	47.2	78.0	37.2	13.7	46.1	7230	87781	1206
	GOTURN [21]	37.5	75.4	25.1	8.4	46.5	17746	92867	3277
	HCC [36]	49.3	**79.0**	50.7	17.8	39.9	5333	86795	**391**
	LMP [52]	48.8	**79.0**	51.3	18.2	40.1	6654	86245	481
	GCRA [35]	48.2	77.5	48.6	12.9	41.1	5104	88586	821
	FWT [22]	47.8	75.5	44.3	**19.1**	38.2	8886	85487	852
	MOTDT [34]	47.6	74.8	50.9	15.2	38.3	9253	85431	792

Table 4. We establish a new state-of-the-art on MOT16 and MOT17 public benchmarks by using the proposed DeepMOT.

MOT-by-SOT Ablation. Using DeepMOT, we can turn trainable SOT methods into trainable MOT methods by combining them with the track management mechanism (as explained in Sec. 5.2) and optimize their parameters using our loss. In Tab. 3, we outline the results of the two MOT-by-SOT baselines (GOTURN [21] and SiamRPN [32]). For both, we show the performance when using (i) a pre-trained network, (ii) a network fine-tuned using the Smooth L_1 loss, and (iii) the one trained with DeepMOT.

Based on the results outlined in Tab. 3, we conclude that training using the Smooth L_1 loss improves the MOTA for both SOTs (GOTURN: $+6.29\%$, SiamRPN: $+1.16\%$). Moreover, compared to models trained with Smooth L_1 loss, we further improve MOTA and reduce the number of IDS when we train them using DeepMOT. For GOTURN (SiamRPN), we record a MOTA improvement of 1.81% (0.65%) while reducing the number of IDS by 211 (6). We also outline the improvements comparing **Vanilla+ReIDext** Tracktor trained with Smooth L_1 loss, and **DeepMOT+ReIDhead** Tracktor trained using DeepMOT. These results further validate the merit and generality of our method for training deep multi-object trackers.

MOTChallenge Benchmark Evaluation We evaluate the trackers trained using our framework on the MOTChallenge benchmark (test set) using the best-performing configuration, determined previously using the validation set. During training and inference, we use the camera motion compensation module, as proposed by [4], for the three trained trackers. We discuss the results obtained on MOT16-17. MOT15 results and parameters are in the supplementary.

We follow the standard evaluation practice and compare our models to methods that are officially published on the MOTChallenge benchmark and peer-reviewed. For MOT16 and MOT17, we average the results obtained using the three sets of provided public detections (DPM [18], SDP [16] and Faster R-CNN [43]). As in [4], we use these public detections for track initialization and termination. Importantly, in the case of Tracktor, we do not use the internal detection mechanism of the network, but only public detections.

As can be seen in Tab. 4, DeepMOT-Tracktor establishes a new state-of-the-art on both MOT17 and MOT16. We improve over Tracktor (on MOT17 and MOT16, respectively) in terms of (i) MOTA (0.2% and 0.4%), (ii) IDF1 (1.5% and 0.9%) and (iii) IDS (125 and 37). On both benchmarks, Vanilla Tracktor is the second best-performing method, and our simple SOT-by-MOT baseline DeepMOT-SiamRPN is the third. We observe large improvements over our MOT-by-SOT pre-trained models and models trained using DeepMOT. For GOTURN, we improve MOTA by 9.8% and 9.7% and we significantly reduce the number of IDS by 6536 and 2071, for MOT17 and MOT16 respectively. Similar impact on DeepMOT-SiamRPN is observed.

6. Conclusion

In this paper, we propose an end-to-end MOT training framework, based on a differentiable approximation of HA and CLEAR-MOT metrics. We experimentally demonstrate that our proposed MOT framework improves the performance of existing deep MOT methods. Thanks to our method, we set a new state-of-the-art score on the MOT16 and MOT17 datasets. We believe that our method was the missing block for advancing the progress in the area of end-to-end learning for deep multi-object tracking. We expect that our training module holds the potential to become a building block for training future multi-object trackers.

Acknowledgements

We gratefully acknowledge the mobility grants from the Department for Science and Technology of the French Embassy in Berlin (SST) and the French Institute for Research in Computer Science and Automation (Inria), especially the Perception team. We are grateful to the Dynamic Vision and Learning Group, Technical University of Munich as the host institute, especially Guillem Brasó and Tim Meinhardt for the fruitful discussions. Finally, this research was partially funded by the Humboldt Foundation through the Sofja Kovalevskaja Award.

Supplementary Material

A. Implementation Details

A.1. DHN

For training the DHN, we use the RMSprop optimizer [51] with a learning rate of 0.0003, gradually decreasing by 5% every $20{,}000$ iterations. We train DHN for 20 epochs (6 hours on a Titan XP GPU). For the focal loss, we weight zero-class by $w_0 = n_1/(n_0 + n_1)$ and one-class by $w_1 = 1 - w_0$. Here n_0 is the number of zeros and n_1 the number of ones in $\mathbf{A}^*$. We also use a modulating factor of 2 in the focal loss. Once the DHN training converges, we freeze the DHN weights and keep them fixed when training trackers with DeepMOT.

Datasets. To train the DHN, we generate training pairs as follows. We first compute distance matrices $\mathbf{D}$ using ground-truth labels (bounding boxes) and object detections provided by the MOTChallenge datasets (MOT 15-17) [30, 37]. We augment the data by setting all entries, higher than the randomly (with an uniform distribution ranging from 0 to 1) selected threshold, to a large value to discourage these assignments. This way, we obtain a rich set of various distance matrices. We then compute assignments using the (Hungarian algorithm) HA (variant used in [6]) to get the corresponding (binary) assignment matrices $\mathbf{A}^*$, used as a supervisory signal. In this way, we obtain a dataset of matrix pairs ($\mathbf{D}$ and $\mathbf{A}^*$), separated into $114{,}483$ training and $17{,}880$ testing instances.

A.2. Trackers

Datasets. For training object trackers, we use the MOT17 train set. For the ablation studies, we divide the MOT17 into train/val sets. We split each sequence into three parts: the first, one containing 50% of frames, the second one 25%, and the third 25%. We use the first 50% for training data and the last 25% for validation to make sure there is no overlap between the two. In total, we use $2{,}664$ frames for the train set, containing $35{,}836$ ground-truth bounding boxes and 306 identities. For the validation split, we have $1{,}328$ frames with 200 identities. The public object detections (obtained by DPM [18], SDP [57] and Faster RCNN [43] detectors) from the MOTChallenge are used only during tracking.

Training. We use the Adam optimizer [29] with a learning rate of 0.0001. We train the SOTs for 15 epochs (72h), and we train Tracktor (regression head and ReID head) for 18 epochs (12h) on a Titan XP GPU.

Loss Hyperparameters. When training trackers using our DeepMOT loss, we set the base value of $\delta = 0.5$, and the loss balancing factors of $\lambda = 5, \gamma = 2$, as determined on the validation set.

Training Details. To train object trackers, we randomly select one training instance from the sequence that corresponds to a pair of consecutive frames. Then, we initialize object trackers using ground-truth detections and predict track continuations in the next frame. At each time step, we use track predictions and ground-truth bounding boxes to compute $\mathbf{D}$, which we pass to our DHN and, finally, compute loss and back-propagate the gradients to the tracker.

Data Augmentation. We initialize trackers using ground-truth bounding boxes. To mimic the effects of imperfect object detectors and prevent over-fitting, we perform the following data augmentations during the training:

- We randomly re-scale the bounding boxes with a scaling factor ranging from 0.8 to 1.2.
- We add random vertical and horizontal offset vectors (bounding box width and/or height scaled by a random factor ranging from 0 to 0.25).

Training with the ReIDhead. While training Tracktor with our **ReIDhead**, we make the following changes. Instead of selecting a pair of video frames, we randomly select ten consecutive frames. This is motivated by the implementation of external ReID mechanism in [4], where tracker averages appearance features over ten most recent frames. At each training step, we compute representative embedding by averaging embeddings of the past video frames and use it to compute the cosine distance to the ground-truth object embeddings.

Test-time Track Managment. For the MOT-by-SOT baseline, we use detections from three different detectors (DPM, SDP, and FRCNN) to refine the track predictions. When the IoU between a track prediction and detection is higher than 0.6, we output their average. We also reduce FP in the public detections based on detection scores produced by a Faster RCNN detector. For the birth and death processes, we initialize a new track only when detections appear in 3 consecutive frames, and they have a minimal consecutive IoU overlap of 0.3. Tracks that can not be verified by the detector are marked invisible and are terminated after $K = 60$ frames. For Tracktor, we use the same track management and suppression strategy as proposed in [4].

B. Additional DHN Ablation

We perform DHN ablation using our test split of $17{,}880$ DHN training instances, as explained in Sec. A.1. In addition, we evaluate the generalization of DHN by evaluating performing evaluation using distance matrices, generated during the DeepMOT training process.

Accuracy. We compute the weighted accuracy as (using

the same weighting factors w_1 and w_0 as for the loss):

$$\text{WA} = \frac{w_1 n_1^* + w_0 n_0^*}{w_1 n_1 + w_0 n_0}. \tag{11}$$

Here, n_1^* and n_0^* are the number of true and false positives, respectively.

Validity. The output of the matching algorithm should be a permutation matrix; *i.e.*, there should be at most one assignment per row/column. In the case of the HA, this is explicitly enforced via constraints on the solution. To study how well the predicted (discretized) assignment matrices preserve this property, we count the number of rows and columns by the following criteria:

- **Several Assignments (SA)** counts the number of rows/columns that have more than one assignment (when performing column-wise maximum and row-wise maximum, respectively).
- **Missing Assignments (MA)** counts the number of rows/columns that are not assigned (when performing column-wise maximum and row-wise maximum, respectively) when ground-truth assignment matrix $\mathbf{A}^*$ has an assignment or inversely, no assignment in $\mathbf{A}^*$ while $\bar{\mathbf{A}}$ (see below) has an assignment in the corresponding rows/columns.

Discretization. To perform the evaluation, we first need to discretize the soft assignment matrix $\tilde{\mathbf{A}}$, predicted by our DHN to obtain a discrete assignment matrix $\bar{\mathbf{A}}$. There are two possibilities.

(i) For each row of $\bar{\mathbf{A}}$, we set the entry of $\bar{\mathbf{A}}$ corresponding to the largest value of the row to 1 (as long as it exceeds 0.5) and the remaining values are set to 0. We refer to this variant as *row-wise maximum*.

(ii) Analogously, we can perform *column-wise maximum* by processing columns instead of rows.

DHN variants. We compare three different DHN architectures:

(i) Sequential DHN (**seq**, see Fig. 5),

(ii) Parallel DHN (**paral**, see Fig. 6),

(iii) 1D Convolutional DHN (**1d_conv**, see Fig. 7).

The recurrent unit of the two recurrent architectures, **seq** and **paral**, is also ablated between long-short term memory units (**lstm**) [23] and gated recurrent units (**gru**) [10].

From Tab. 5, we see that the proposed sequential DHN (**seq_gru**) obtains the highest WA (92.88% for row-wise maximum and 93.49% for column-wise maximum) compared to others. Compared to the 1D convolutional DHN

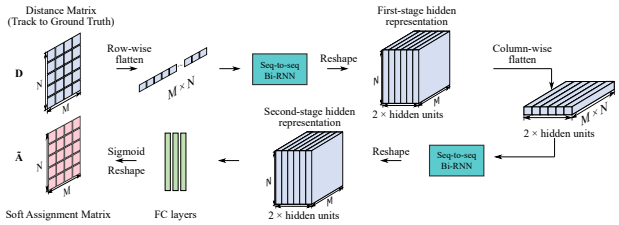

Figure 5. Sequential DHN: Structure of the proposed Deep Hungarian Net. The row-wise and column-wise flattening are inspired by the original Hungarian algorithm, while the Bi-RNN allows for all decisions to be taken globally, thus is accounting for all input entries.

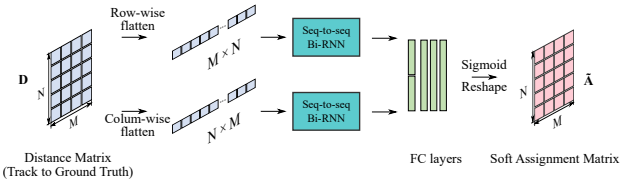

Figure 6. Parallel DHN variant: (i) We perform row-wise and the column-wise flattening of $\mathbf{D}$. (ii) We process the flattened vectors using two different Bi-RNNs. (iii) They then are respectively passed to an FC layer for reducing the number of channels and are concatenated along the channel dimension. (iv) After two FC layers we reshape the vector and apply the sigmoid activation.

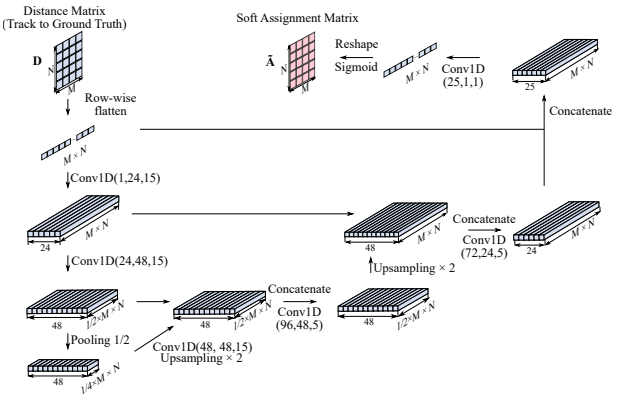

Figure 7. 1D convolutional DHN: Our 1D convolutional DHN variant is inspired by the U-Net [45]. The encoder consists of two 1D-convolution layers of shapes $[1, 24, 15]$ and $[24, 48, 15]$ ([#input channels, #output channels, kernel size]). The decoder consists of two 1D convolutional layers of shapes $[96, 48, 5]$ and $[72, 24, 5]$. Finally, we apply an 1D convolution and a sigmoid activation to produce $\tilde{\mathbf{A}}$.

variant (WA of 56.43% and 56.18% for row-wise and column-wise maximum, respectively), Bi-RNN shows the advantage of its global view due to the receptive field, equal to the entire input. For the sequential DHN setting, we observe in Tab. 5 that **gru** units consistently outperform **lstm** units with WA +9.22% (row-wise maximum) and +6.42% (column-wise maximum). Finally, the proposed sequential DHN is more accurate compared to the parallel variant of DHN (+3.32% for row-wise and +2.48% for column-

Discretization	Network	WA % (↑)	MA% (↓)	SA% (↓)
Row-wise maximum	**seq_gru (proposed)**	**92.88**	**4.79**	3.39
	seq_lstm	83.66	13.79	5.98
	paral_gru	89.56	8.21	4.99
	paral_lstm	88.93	8.67	5.38
	1d_conv	56.43	35.06	**2.78**
Column-wise maximum	**seq_gru (proposed)**	**93.49**	**6.41**	26.57
	seq_lstm	87.07	13.54	47.04
	paral_gru	91.01	7.98	46.25
	paral_lstm	90.50	8.60	47.43
	1d_conv	56.18	79.54	**7.73**

Table 5. Evaluation results: comparison of different network structures and settings in terms of WA, MA and SA on the DHN test set.

Discretization	Network	WA % (↑)	MA% (↓)	SA% (↓)
Row-wise maximum	**seq_gru (proposed)**	**92.71**	**13.17**	9.70
	seq_lstm	91.64	14.55	10.37
	paral_gru	86.84	23.50	17.15
	paral_lstm	71.58	42.48	22.62
	1d_conv	83.12	32.73	**5.73**
Column-wise maximum	**seq_gru (proposed)**	**92.36**	**12.21**	3.69
	seq_lstm	91.93	13.15	4.71
	paral_gru	87.24	20.56	16.67
	paral_lstm	72.58	39.55	23.16
	1d_conv	82.74	32.94	**1.11**

Table 6. Evaluation results. Comparison of different network structures and settings in terms of WA, MA and SA on distance matrices during training.

wise maximum). As for the validity, the proposed **seq_gru** commits the least missing assignments (MA) (4.79% and 6.41% for row-wise and column-wise maximum, respectively), and commits only a few SA compared to other variants.

DHN is a key component of our proposed DeepMOT training framework. To evaluate how well DHN performs during training as a proxy to deliver gradients from the DeepMOT loss to the tracker, we conduct the following experiment. We evaluate DHN using distance matrices $\mathbf{D}$, collected during the DeepMOT training process. As can be seen in Tab. 6, the proposed sequential DHN (**seq_gru**) outperforms the others variants, with a WA of 92.71% for row-wise and 92.36% for column-wise maximum. For validity, it also attains the lowest MA: 13.17% (row) and 12.21% (column). The SA is kept at a low level with 9.70% and 3.69% for row-wise and column-wise maximum discretizations, respectively. Based on these results, we conclude that (i) our proposed DHN generalizes well to matrices, used to train our trackers, and (ii) it produces outputs that closely resemble valid permutation matrices.

Matrix Size. To provide further insights into DHN, we study the impact of the distance matrix size on the assignment accuracy. We visualize the relation between WA and the input matrix size in Fig. 8. For validation, we generate

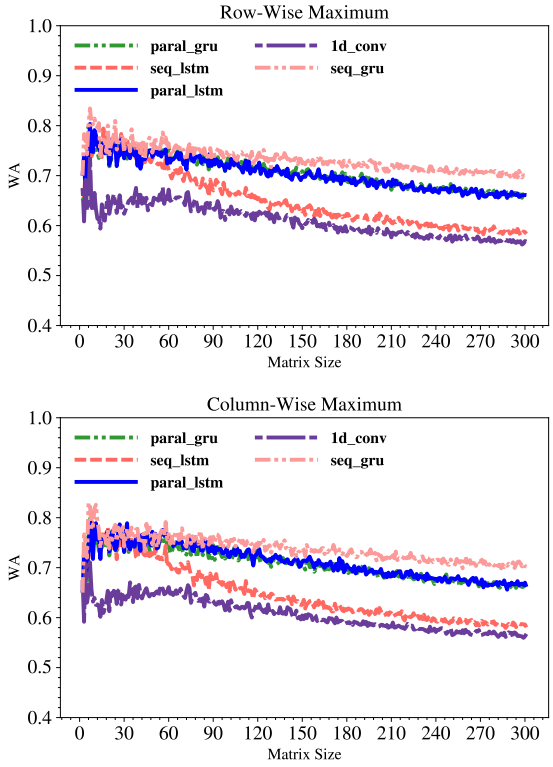

Figure 8. Evaluation of performance of DHN and its variants on $\mathbf{D}$ of different sizes.

square matrices with sizes ranging from $[2, 300]$. Precisely, we generate $\mathbf{D}$ with a uniform distribution $[0, 1)$ and use the Hungarian algorithm implementation from [6] to generate assignment matrices $\mathbf{A}^*$. For each size, we evaluate 10 matrices, which gives us 2,990 matrices in total. As can be seen in Fig. 8, (i) the proposed **seq_gru** consistently outperforms the alternatives. (ii) The assignment accuracy of DHN and its variants decreases with the growth of the matrix size. Moreover, we observe a performance drop for very small matrices (*i.e.*, $\mathrm{M} = \mathrm{N} \leqslant 6$). This may be due to the imbalance with respect to the matrix size during the training.

C. Training Gradient Visualization

The negative gradient should reflect the direction that minimizes the loss. In in Fig. 9 we plot the negative gradient of different terms that constitute our DeepMOT loss w.r.t. the coordinates of each predicted bounding box to demonstrate visually the effectiveness of our DeepMOT. In this example, we manually generated the cases which contain the FP, FN or IDS. We observe that the negative gradient does encourage the tracks' bounding boxes to be close to those of their associated objects during the training.

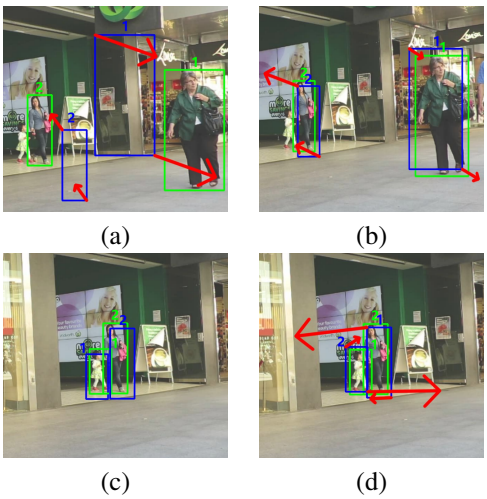

Figure 9. Visualization of negative gradients (direction and magnitude) from different terms in the proposed DeepMOT loss: (a) FP and FN (b) MOTP (c-d) IDS (compare (c) $t-1$ with (d) t). The predicted bounding-boxes are shown in blue, the ground-truth are shown in green and the gradient direction is visualized using red arrows.

	Method	MOTA ↑	MOTP ↑	IDF1 ↑	MT ↑	ML ↓	FP ↓	FN ↓	IDS ↓
2D MOT 2015	DeepMOT-Tracktor	**44.1**	**75.3**	46.0	17.2	26.6	6085	26917	1347
	Tracktor [4]	**44.1**	75.0	46.7	18.0	**26.2**	6477	**26577**	1318
	DeepMOT-SiamRPN	33.3	74.6	32.7	9.3	43.7	7825	32211	919
	SiamRPN [32]	31.0	73.9	30.7	12.6	41.7	10241	31099	1062
	DeepMOT-GOTURN	29.8	**75.3**	27.7	4.0	66.6	3630	38964	524
	GOTURN [21]	23.9	72.8	22.3	3.6	66.4	7021	38750	965
	AP_HWDPL_p [9]	38.5	72.6	**47.1**	8.7	37.4	**4005**	33203	586
	AMIR15 [46]	37.6	71.7	46.0	15.8	26.8	7933	29397	1026
	JointMC [24]	35.6	71.9	45.1	**23.2**	39.3	10580	28508	457
	RAR15pub [17]	35.1	70.9	45.4	13.0	42.3	6771	32717	**381**

Table 7. Results on MOTChallenge MOT15 benchmark.

D. MOT15 Results

We summarize the results we obtain on MOT15 dataset in Tab. 7. Our key observations are:

(i) For the MOT-by-SOT baseline, we significantly improve over the trainable baselines (SiamRPN and GOTURN). DeepMOT-SiamRPN increases MOTA for $+2.3\%$, MOTP for $+0.7\%$ and IDF1 for $+2.0\%$. Remarkably, DeepMOT-SiamRPN suppresses $2{,}416$ FP and 143 IDS. We observe similar performance gains for DeepMOT-GOTURN.

(ii) DeepMOT-Tracktor obtains results, comparative to the vanilla Tracktor [4]. Different from MOT16 and MOT17 datasets, we observe no improvements in terms of MOTA, which we believe is due to the fact that labels in MOT15 are very noisy, and vanilla Tracktor already achieves impressive performance. Still, we increase MOTP for 0.3% and reduce FP for 392.

References

[1] Sileye Ba, Xavier Alameda-Pineda, Alessio Xompero, and Radu Horaud. An on-line variational bayesian model for multi-person tracking from cluttered scenes. *CVIU*, 153:64–76, 2016. 2

[2] Yutong Ban, Xavier Alameda-Pineda, Laurent Girin, and Radu Horaud. Variational bayesian inference for audio-visual tracking of multiple speakers. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019. 2

[3] Yutong Ban, Sileye Ba, Xavier Alameda-Pineda, and Radu Horaud. Tracking multiple persons based on a variational bayesian model. *ECCV*, 2016. 2

[4] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixé. Tracking without bells and whistles. *ICCV*, 2019. 1, 2, 6, 8, 9, 12

[5] James Bergstra, Daniel Yamins, and David D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. *ICML*, 2013. 2

[6] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: The clear mot metrics. *JIVP*, 2008:1:1–1:10, 2008. 1, 2, 3, 5, 6, 9, 11

[7] William Brendel, Mohamed R. Amer, and Sinisa Todorovic. Multi object tracking as maximum weight independent set. *CVPR*, 2011. 1, 2

[8] Asad A. Butt and Robert T. Collins. Multi-target tracking by lagrangian relaxation to min-cost network flow. *CVPR*, 2013. 1, 2

[9] Long Chen, Haizhou Ai, Chong Shang, Zijie Zhuang, and Bo Bai. Online multi-object tracking with convolutional neural networks. *ICIP*, 2017. 12

[10] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 10

[11] Hyunggi Cho, Young-Woo Seo, BVK Vijaya Kumar, and Ragunathan Raj Rajkumar. A multi-sensor fusion system for moving object detection and tracking in urban driving environments. *ICRA*, 2014. 2

[12] Wongun Choi. Near-online multi-target tracking with aggregated local flow descriptor. *ICCV*, 2015. 2

[13] Peng Chu and Haibin Ling. Famnet: Joint learning of feature, affinity and multi-dimensional assignment for online multiple object tracking. *ICCV*, 2019. 2

[14] Qi Chu, Wanli Ouyang, Hongsheng Li, Xiaogang Wang, Bin Liu, and Nenghai Yu. Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism. *ICCV*, 2017. 1, 2

[15] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. *CVPR*, 2005. 2

[16] Piotr Dollár, Ron Appel, Serge Belongie, and Pietro Perona. Fast feature pyramids for object detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(8):1532–1545, 2014. 8

[17] Kuan Fang, Yu Xiang, Xiaocheng Li, and Silvio Savarese. Recurrent autoregressive networks for online multi-object tracking. *WACV*, 2018. 12

[18] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. *CVPR*, 2008. 2, 8, 9

[19] Chuang Gan, Hang Zhao, Peihao Chen, David Cox, and Antonio Torralba. Self-supervised moving vehicle tracking with stereo sound. *ICCV*, 2019. 2

[20] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. *CVPR*, 2006. 2

[21] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. *ECCV*, 2016. 7, 8, 12

[22] Roberto Henschel, Laura Leal-Taixe, Daniel Cremers, and Bodo Rosenhahn. Improvements to frank-wolfe optimization for multi-detector multi-object tracking. *arXiv preprint arXiv:1705.08314*, 2017. 8

[23] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 10

[24] Margret Keuper, Siyu Tang, Bjorn Andres, Thomas Brox, and Bernt Schiele. Motion segmentation & multiple object tracking by correlation co-clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2018. 8, 12

[25] Chanho Kim, Fuxin Li, Arridhana Ciptadi, and James M. Rehg. Multiple hypothesis tracking revisited. *ICCV*, 2015. 2, 8

[26] Chanho Kim, Fuxin Li, and James M. Rehg. Multi-object tracking with neural gating using bilinear lstm. *ECCV*, 2018. 2

[27] Harold William Kuhn and Bryn Yaw. The hungarian method for the assignment problem. *Naval research logistics quarterly*, pages 83–97, 1955. 1, 2, 3, 5

[28] Laura Leal-Taixé, Cristian Canton-Ferrer, and Konrad Schindler. Learning by tracking: Siamese cnn for robust target association. *CVPR Workshops*, 2016. 1, 2

[29] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. ICLR, 2015. 9

[30] Laura Leal-Taixé, Anton Milan, Ian Reid, Stefan Roth, and Konrad Schindler. MOTChallenge 2015: Towards a benchmark for multi-target tracking. *arXiv preprint arXiv:1504.01942*, 2015. 2, 6, 9

[31] Bastian Leibe, Konrad Schindler, Nico Cornelis, and Luc Van Gool. Coupled object detection and tracking from static cameras and moving vehicles. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(10):1683–1698, 2008. 2

[32] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. *CVPR*, 2018. 7, 8, 12

[33] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. *ICCV*, 2017. 6

[34] Chen Long, Ai Haizhou, Zhuang Zijie, and Shang Chong. Real-time multiple people tracking with deeply learned candidate selection and person re-identification. *ICME*, 2018. 8

[35] Cong Ma, Changshui Yang, Fan Yang, Yueqing Zhuang, Ziwei Zhang, Huizhu Jia, and Xiaodong Xie. Trajectory factory: Tracklet cleaving and re-connection by deep siamese bi-gru for multiple object tracking. *ICME*, 2018. 8

[36] Liqian Ma, Siyu Tang, Michael J. Black, and Luc Van Gool. Customized multi-person tracker. *ACCV*, 2018. 8

[37] Anton Milan, Laura. Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. MOT16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016. 2, 6, 9

[38] Anton Milan, Stefan Roth, and Konrad Schindler. Continuous energy minimization for multitarget tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(1):58–72, 2014. 2

[39] Songhwai Oh, Stuart Russell, and Shankar Sastry. Markov chain monte carlo data association for multi-target tracking. *IEEE Trans. Autom. Control*, 54(3):481–497, 2009. 2

[40] Aljoša Ošep, Wolfgang Mehner, Markus Mathias, and Bastian Leibe. Combined image- and world-space tracking in traffic scenes. *ICRA*, 2017. 2

[41] Hamed Pirsiavash, Deva Ramanan, and Charles C.Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. *CVPR*, 2011. 1, 2

[42] Donald B. Reid. An algorithm for tracking multiple targets. *IEEE Trans. Autom. Control*, 24(6):843–854, 1979. 1, 2

[43] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *NIPS*, 2015. 6, 8, 9

[44] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. *ECCV*, 2016. 3, 6

[45] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *MICCAI*, pages 234–241. Springer, 2015. 10

[46] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. *ICCV*, 2017. 12

[47] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *CVPR*, 2015. 1, 2, 6

[48] Samuel Schulter, Paul Vernaza, Wongun Choi, and Manmohan Krishna Chandraker. Deep network flow for multi-object tracking. *CVPR*, 2017. 1, 2

[49] Hao Sheng, Yang Zhang, Jiahui Chen, Zhang Xiong, and Jun Zhang. Heterogeneous association graph fusion for target association in multiple object tracking. *IEEE Trans. Circuits Syst. Video Technol.*, 2018. 8

[50] Jeany Son, Mooyeol Baek, Minsu Cho, and Bohyung Han. Multi-object tracking with quadruplet convolutional neural networks. *CVPR*, 2017. 1, 2

[51] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 2012. 9

[52] Siyu Tang, Mykhaylo Andriluka, Bjoern Andres, and Bernt Schiele. Multiple people tracking by lifted multicut and person re-identification. *CVPR*, 2017. 8

[53] Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. *NIPS*, 2003. 2

[54] Paul Voigtlaender, Michael Krause, Aljosa Osep, Jonathon Luiten, B.B.G Sekar, Andreas Geiger, and Bastian Leibe. MOTS: Multi-object tracking and segmentation. *CVPR*, 2019. 1, 2, 6

[55] Shaofei Wang and Charless C. Fowlkes. Learning optimal parameters for multi-target tracking with contextual interactions. *IJCV*, 122(3):484–501, 2016. 2

[56] Yu Xiang, Wongun Choi, Yuanqing Lin, and Silvio Savarese. Data-driven 3d voxel patterns for object category recognition. *CVPR*, 2015. 2

[57] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. *CVPR*, 2016. 9

[58] Li Zhang, Li Yuan, and Ramakant Nevatia. Global data association for multi-object tracking using network flows. *CVPR*, 2008. 1, 2

[59] Ji Zhu, Hua Yang, Nian Liu, Minyoung Kim, Wenjun Zhang, and Ming-Hsuan Yang. Online multi-object tracking with dual matching attention networks. *ECCV*, 2018. 1, 2