Deep Learning in Video Multi-Object Tracking: A Survey

Gioele Ciaparrone[1,2]**, Francisco Luque Sánchez**[2]**, Siham Tabik**[2]**, Luigi Troiano**[3]**, Roberto Tagliaferri**[1]**, Francisco Herrera**[2]

[1]Department of Management and Innovation Systems, University of Salerno, 84084 Fisciano (SA), Italy
[2]Andalusian Research Institute in Data Science and Computational Intelligence, University of Granada, 18071 Granada, Spain
[3]Department of Engineering, University of Sannio, 82100 Benevento, Italy
{gciaparrone,robtag}@unisa.it, {fluque,herrera}@decsai.ugr.es, siham@ugr.es, troiano@unisannio.it

August 1, 2019

Abstract

The problem of Multiple Object Tracking (MOT) consists in following the trajectory of different objects in a sequence, usually a video. In recent years, with the rise of Deep Learning, the algorithms that provide a solution to this problem have benefited from the representational power of deep models. This paper provides a comprehensive survey on works that employ Deep Learning models to solve the task of MOT on single-camera videos. Four main steps in MOT algorithms are identified, and an in-depth review of how Deep Learning was employed in each one of these stages is presented. A complete experimental comparison of the presented works on the three MOTChallenge datasets is also provided, identifying a number of similarities among the top-performing methods and presenting some possible future research directions.

Keywords Multiple Object Tracking · Deep Learning · Video Tracking · Computer Vision · Convolutional Neural Networks · LSTM · Reinforcement Learning

1 Introduction

Multiple Object Tracking (MOT), also called Multi-Target Tracking (MTT), is a computer vision task that aims to analyze videos in order to identify and track objects belonging to one or more categories, such as pedestrians, cars, animals and inanimate objects, without any prior knowledge about the appearance and number of targets. Differently from object detection algorithms, whose output is a collection of rectangular bounding boxes identified by their coordinates, height and width, MOT algorithms also associate a target ID to each box (known as a *detection*), in order to distinguish among intra-class objects. An example of the output of a MOT algorithm is illustrated in figure 1. The MOT task plays an important role in computer vision: from video surveillance to autonomous cars, from action recognition to crowd behaviour analysis, many of these problems would benefit from a high-quality tracking algorithm.

While in Single Object Tracking (SOT) the appearance of the target is known a priori, in MOT a detection step is necessary to identify the targets, that can leave or enter the scene. The main difficulty in tracking multiple targets simultaneously stems from the various occlusions and interactions between objects, that can sometimes also have similar appearance. Thus, simply applying SOT models directly to solve MOT leads to poor results, often incurring in target drift and numerous ID switch errors, as such models usually struggle in distinguishing between similar looking intra-class objects. A series of algorithms specifically tuned to multi-target tracking have then been developed in recent years to address these issues, together with a number of benchmark datasets and competitions to ease the comparisons between the different methods.

Recently, more and more of such algorithms have started exploiting the representational power of deep learning (DL). The strength of Deep Neural Networks (DNN) resides in their ability to learn rich representations and to extract complex and abstract features from their input. Convolutional neural networks (CNN) currently constitute the state-of-the-art in spatial pattern extraction, and are employed in tasks such as image classification [1, 2, 3] or object detection [4, 5, 6], while recurrent neural networks (RNN) like the Long Short-Term Memory (LSTM) are used to process sequential data, like audio signals, temporal series and text [7, 8, 9, 10]. Since DL methods have been able to reach top performance in many of those tasks, we are now progressively seeing them used in most of the top performing MOT algorithms, aiding to solve some of the subtasks in which the problem is divided.

This work presents a survey of algorithms that make use of the capabilities of deep learning models to perform Multiple Object Tracking, focusing on the different approaches used for the various components of a MOT algorithm and putting them in the context of each of the proposed methods. While the MOT task can be applied to both 2D and 3D data, and to both single-camera and multi-camera scenarios, in this survey we focus on 2D data extracted from videos recorded by a single camera.

Some reviews and surveys have been published on the subject of MOT. Their main contributions and limitations are the following:
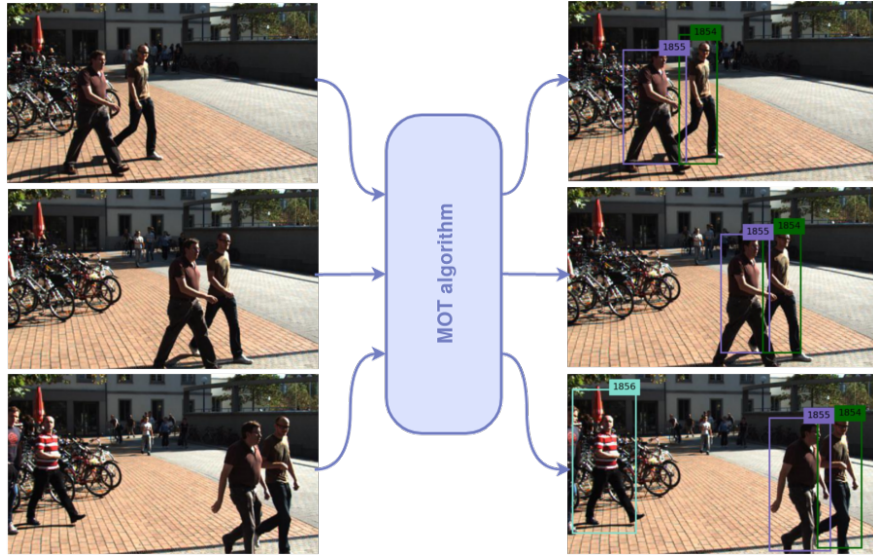

Figure 1: An illustration of the output of a MOT algorithm. Each output bounding box has a number that identifies a specific person in the video.

- Luo et al. [11] presented the first comprehensive review to focus specifically on MOT, in particular on pedestrian tracking. They provided a unified formulation of the MOT problem and described the main techniques used in the key steps of a MOT system. They presented deep learning as one of the future research directions, since at the time it had only been employed by very few algorithms.
- Camplani et al. [12] presented a survey on Multiple Pedestrian Tracking, but they focused on RGB-D data, while our focus is on 2D RGB images, without additional inputs. Moreover, their review does not cover deep learning based algorithms.
- Emami et al. [13] proposed a formulation of single and multi-sensor tracking tasks as a Multidimensional Assignment Problem (MDAP). They also presented a few approaches that employed deep learning in tracking problems, but it wasn't the focus of their paper and they didn't provide any experimental comparison among such methods.
- Leal-Taixé et al. [14] presented an analysis of the results obtained by algorithms on the MOT15 [15] and MOT16 [16] datasets, providing a summary of the trending lines of research and statistics about the results. They found that after 2015, methods have been shifting from trying to find better optimization algorithms for the association problem to focusing on improving the affinity models, and they predict that many more approaches would tackle this issue by using deep learning. However, this work also did not focus on deep learning, and it does not cover more recent MOT algorithms, published in the last years.

In this paper, based on the discussed limitations, our aim is to provide a survey with the following main contributions:

- We provide the first comprehensive survey on the use of Deep Learning in Multiple Object Tracking, focusing on 2D data extracted from single-camera videos, including recent works that have not been covered by past surveys and reviews. The use of DL in MOT is in fact recent, and many approaches have been published in the last three years.
- We identify four common steps in MOT algorithms and describe the different DL models and approaches employed in each of those steps, including the algorithmic context in which they are used. The techniques utilized by each analyzed work are also summarized in a table, together with links to the available source code, to serve as a quick reference for future research.
- We collect experimental results on the most commonly used MOT datasets to perform a numerical comparison among them, also identifying the main trends in the best performing algorithms and possible future directions of research.

The survey is further organized in this manner. We first describe the general structure of MOT algorithms and the most commonly used metrics and datasets in section 2. Section 3 explores the various DL-based models and algorithms in each of the four identified steps of a MOT algorithm. Section 4 presents a numerical comparison among the presented algorithms and identifies common trends and patterns in current approaches, as well as some limitations and possible future research directions. Finally, section 5 summarizes the findings of the previous sections and presents some final remarks.

2 MOT: algorithms, metrics and datasets

In this section, a general description about the problem of MOT is provided. The main characteristics and common steps of MOT algorithms are identified and described in section 2.1. The metrics that are usually employed to evaluate the performance of the models are discussed in section 2.2, while the most important benchmark datasets are presented in section 2.3.

2.1 Introduction to MOT algorithms

The standard approach employed in MOT algorithms is *tracking-by-detection*: a set of detections (i.e. bounding boxes identifying the targets in the image) are extracted from the video frames and are used to guide the tracking process, usually by associating them together in order to assign the same ID to bounding boxes that contain the same target. For this reason, many MOT algorithms formulate the task as an assignment problem. Modern detection frameworks [4, 17, 18, 5, 6] ensure a good detection quality, and the majority of MOT methods (with some exceptions, as we will see) have been focusing on improving the association; indeed, many MOT datasets provide a standard set of detections that can be used by the algorithms (that can thus skip the detection stage) in order to exclusively compare their performances on the quality of the association algorithm, since the detector performance can heavily affect the tracking results.

MOT algorithms can also be divided into batch and online methods. Batch tracking algorithms are allowed to use future information (i.e. from future frames) when trying to determine the object identities in a certain frame. They often exploit global information and thus result in better tracking quality. Online tracking algorithms, on the contrary, can only use present and past information to make predictions about the current frame. This is a requirement in some scenarios, like autonomous driving and robot navigation. Compared to batch methods, online methods tend to perform worse, since they cannot fix past errors using future information. It is important to note that while a real-time algorithm is required to run in an online fashion, not every online method necessarily runs in real-time; quite often, in fact, with very few exceptions, online algorithms are still too slow to be employed in a real-time environment, especially when exploiting deep learning algorithms, that are often computationally intensive.

Despite the huge variety of approaches presented in the literature, the vast majority of MOT algorithms share part or all of the following steps (summarized in figure 2):

- Detection stage: an object detection algorithm analyzes each input frame to identify objects belonging to the target class(es) using bounding boxes, also known as 'detections' in the context of MOT;

- Feature extraction/motion prediction stage: one or more feature extraction algorithms analyze the detections and/or the tracklets to extract appearance, motion and/or interaction features. Optionally, a motion predictor predicts the next position of each tracked target;

- Affinity stage: features and motion predictions are used to compute a similarity/distance score between pairs of detections and/or tracklets;

- Association stage: the similarity/distance measures are used to associate detections and tracklets belonging to the same target by assigning the same ID to detections that identify the same target.
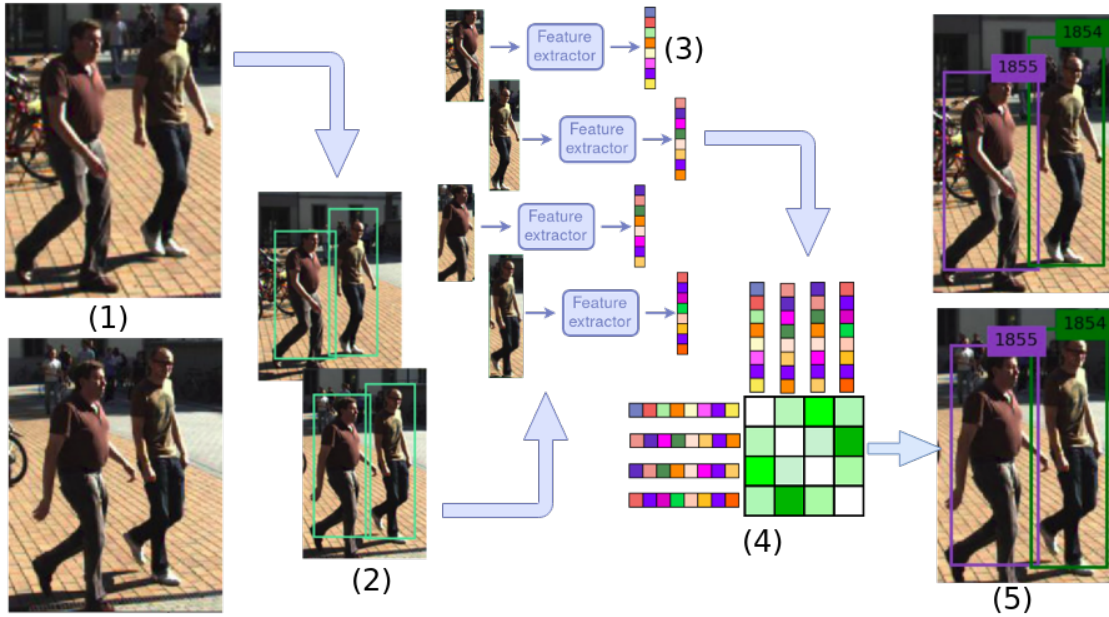

Figure 2: Usual workflow of a MOT algorithm: given the raw frames of a video (1), an object detector is run to obtain the bounding boxes of the objects (2). Then, for every detected object, different features are computed, usually visual and motion ones (3). After that, an affinity computation step calculates the probability of two objects belonging to the same target (4), and finally an association step assigns a numerical ID to each object (5).

While these stages can be performed sequentially in the order presented here (often once per frame for online methods and once for the whole video for batch methods), there are many algorithms that merge some of these steps together, or intertwine them, or even perform them multiple times using different techniques (e.g. in algorithms that work in two phases). Moreover, some methods do not directly associate detections together, but use them to refine trajectory predictions and to manage initialization and termination of new tracks; nonetheless, many of the presented steps can often still be identified even in such cases, as we will see.

2.2 Metrics

In order to provide a common experimental setup where algorithms can be fairly tested and compared, a group of metrics have been *de facto* established as standard, and they are used in almost every work. The most relevant ones are metrics defined by Wu and Nevatia [19], the so-called CLEAR MOT metrics [20], and recently the ID metrics [21]. These sets of metrics aim to reflect the overall performance of the tested models, and point out the possible drawbacks of each one. Therefore, those metrics are defined as follows:

Classical metrics

These metrics, defined by Wu and Nevatia [19], highlight the different types of errors a MOT algorithm can make. In order to show those problems, the following values are computed:

- *Mostly Tracked* (MT) trajectories: number of ground-truth trajectories that are correctly tracked in at least 80% of the frames.
- *Fragments*: trajectory hypotheses which cover at most 80% of a ground truth trajectory. Observe that a true trajectory can be covered by more than one fragment.
- *Mostly Lost* (ML) trajectories: number of ground-truth trajectories that are correctly tracked in less than 20% of the frames.
- *False trajectories*: predicted trajectories which do not correspond to a real object (i.e. to a ground truth trajectory).
- *ID switches*: number of times when the object is correctly tracked, but the associated ID for the object is mistakenly changed.

CLEAR MOT metrics

The CLEAR MOT metrics were developed for the *Classification of Events, Activities and Relationships* (CLEAR) workshops held in 2006 [22] and 2007 [23]. The workshops were jointly organized by the the European CHIL project, the U.S. VACE project, and the National Institute of Standards and Technology (NIST). Those metrics are MOTA (Multiple Object Tracking Accuracy) and MOTP (Multiple Object Tracking Precision). They serve as a summary of other simpler metrics which compose them. We will explain the simpler metrics at first and build the complex ones over them. A detailed description on how to match the real objects (ground truth) with the tracker hypothesis can be found in [20], as it is not trivial how to consider when a hypothesis is related to an object, and it depends on the precise tracking task to be evaluated. In our case, as we are focusing on 2D tracking with single camera, the most used metric to decide whether an object and a prediction are related or not is Intersection over Union (IoU) of bounding boxes, as it was the measure established in the presentation paper of MOT15 dataset [15]. Specifically, the mapping between ground truth and hypotheses is established as follows: if the ground truth object o_i and the hypothesis h_j are matched in frame $t-1$, and in frame t the $IoU(o_i, h_j) \geq 0.5$, then o_i and h_j are matched in that frame, even if there exists another hypothesis h_k such that $IoU(o_i, h_j) < IoU(o_i, h_k)$, considering the continuity constraint. After the matching from previous frames has been performed, the remaining objects are tried to be matched with the remaining hypotheses, still using a 0.5 IoU threshold. The ground truth bounding boxes that cannot be associated with a hypothesis are counted as *false negatives* (FN), and the hypotheses that cannot be associated with a real bounding box are marked as *false positives* (FP). Also, every time a ground truth object tracking is interrupted and later resumed is counted as a *fragmentation*, while every time a tracked ground truth object ID is incorrectly changed during the tracking duration is counted as an *ID switch*. Then, the simple metrics computed are the following:

- FP: the number of false positives in the whole video;
- FN: the number of false negatives in the whole video;
- Fragm: the total number of fragmentations;
- IDSW: the total number of ID switches.

The MOTA score is then defined as follows:

$$MOTA = 1 - \frac{(FN + FP + IDSW)}{GT} \qquad \in (-\infty, 1]$$

where GT is the number of ground truth boxes. It is important to note that the score can be negative, as the algorithm can commit a number of errors greater than the number of ground truth boxes. Usually, instead of reporting MOTA, it is

common to report the percentage MOTA, which is just the previous expression expressed as a percentage. On the other hand, MOTP is computed as:

$$MOTP = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t}$$

where c_t denotes the number of matches in frame t, and $d_{t,i}$ is the bounding box overlap between the hypothesis i with its assigned ground truth object. It is important to note that this metric takes few information about tracking into account, and rather focuses on the quality of the detections.

ID scores

The main problem of MOTA score is that it takes into account the number of times a tracker makes an incorrect decision, such as an ID switch, but in some scenarios (e.g. airport security) one could be more interested in rewarding a tracker that can follow an object for the longest time possible, in order to not lose its position. Because of that, in [21] a couple of alternative new metrics are defined, that are supposed to complement the information given by the CLEAR MOT metrics. Instead of matching ground truth and detections frame by frame, the mapping is performed globally, and the trajectory hypothesis assigned to a given ground truth trajectory is the one that maximizes the number of frames correctly classified for the ground truth. In order to solve that problem, a bipartite graph is constructed, and the minimum cost solution for that problem is taken as the problem solution. For the bipartite graph, the sets of vertices are defined as follows: the first set of vertices, V_T, has a so-called regular node for each true trajectory, and a false positive node for each computed trajectory. The second set, V_C, has a regular node for each computed trajectory and a false negative for each true one. The costs of the edges are set in order to count the number of false negative and false positive frames in case that edge were chosen (more information can be found in [21]). After the association is performed, there are four different possible pairs, attending to the nature of the involved nodes. If a regular node from V_T is matched with a regular node of V_C (i.e. a true trajectory is matched with a computed trajectory), a *true positive ID* is counted. Every false positive from V_T matched with a regular node from V_C counts as a *false positive ID*. Every regular node from V_T matched with a false negative from V_C counts as a *false negative ID*, and finally, every false positive matched with a false negative counts as a *true negative ID*. Afterwards, three scores are calculated. IDTP is the sum of the weights of the edges selected as *true positive ID* matches (it can be seen as the percentage of detections correctly assigned in the whole video). IDFN is the sum of weights from the selected *false negative ID* edges, and IDFP is the sum of weights from the selected *false positive ID* edges. With these three basic measures, another three measures are computed:

- Identification precision: $IDP = \frac{IDTP}{IDTP+IDFP}$
- Identification recall: $IDR = \frac{IDTP}{IDTP+IDFN}$
- Identification F1: $IDF1 = \frac{2}{\frac{1}{IDP}+\frac{1}{IDR}} = \frac{2IDTP}{2IDTP+IDFP+IDFN}$

Usually, the reported metrics in almost every piece of work are the CLEAR MOT metrics, mostly tracked trajectories (MT), mostly lost trajectories (ML) and IDF1, since this metrics are the ones shown in MOTChallenge leaderboards (see section 2.3 for details). Additionally, the number of frames per second (FPS) the tracker can process is often reported, and is also included in the leaderboards. However, we find this metric difficult to compare among different algorithms, since some of the methods include the detection phase while others skip that computation. Also, the dependency on the hardware employed is relevant in terms of speed.

2.3 Benchmark datasets

In the past few years, a number of datasets for MOT have been published. In this section we are going to describe the most important ones, starting from a general description of the MOTChallenge benchmark, then focusing on its datasets, and finally describing KITTI and other less commonly used MOT datasets.

MOTChallenge. MOTChallenge[1] is the most commonly used benchmark for multiple object tracking. It provides, among others, some of largest datasets for pedestrian tracking that are currently publicly available. For each dataset, the ground truth for the training split, and detections for both training and test splits are provided. The reason why MOTChallenge datasets frequently provide detections (often referred to as *public detections*, as opposed to the *private detections*, that are obtained by the algorithm authors by using a detector of their own) is that the detection quality has a big impact on the final performance of the tracker, but the detection part of the algorithms is often independent

---

[1]https://motchallenge.net/

from the tracking part and usually uses already existing models; providing public detections that every model can use makes the comparison of the tracking algorithms easier, since the detection quality is factored out from the performance computation and trackers start on a common ground. The evaluation of an algorithm on the test dataset is done by submitting the results to a test server. The MOTChallenge website contains a leaderboard for each of the datasets, showing in separate pages models using the publicly provided detections and the ones using private detections. Online methods are also marked as so. MOTA is the primary evaluation score for the MOTChallenge, but many other metrics are shown, including all the ones presented in section 2.2. As we will see, since the vast majority of MOT algorithms that use deep learning focus on pedestrians, the MOTChallenge datasets are the most widely used, as they are the most comprehensive ones currently available, providing more data to train deep models.

MOT15. The first MOTChallenge dataset is 2D MOT 2015[2] [15] (often just called MOT15). It contains a series of 22 videos (11 for training and 11 for testing), collected from older datasets, with a variety of characteristics (fixed and moving cameras, different environments and lighting conditions, and so on) so that the models would need to generalize better in order to obtain good results on it. In total, it contains 11283 frames at various resolutions, with 1221 different identities and 101345 boxes. The provided detections were obtained using the ACF detector [24].

MOT16/17. A new version of the dataset was presented in 2016, called MOT16[3] [16]. This time, the ground truth was made from scratch, so that it was consistent throughout the dataset. The videos are also more challenging, since they have a higher pedestrian density. A total of 14 videos are included in the set (7 for training and 7 for testing), with public detections obtained using the Deformable Part-based Model (DPM) v5 [25, 26], that they found to obtain better performance in detecting pedestrians on the dataset when compared to other models. This time the dataset includes 11235 frames with 1342 identities and 292733 boxes in total. The MOT17 dataset[4] includes the same videos as MOT16, but with more accurate ground truth and with three sets of detections for each video: one from Faster R-CNN [4], one from DPM and one from the Scale-Dependent Pooling detector (SDP) [27]. The trackers would then have to prove to be versatile and robust enough to get a good performance using different detection qualities.

MOT19. Very recently, a new version of the dataset for the CVPR 2019 Tracking Challenge[5] has been released, containing 8 videos (4 for training, 4 for testing) with extremely high pedestrian density, reaching up to 245 pedestrians per frame on average in the most crowded video. The dataset contains 13410 frames with 6869 tracks and a total of 2259143 boxes, much more than the previous datasets. While submissions for this dataset have only been allowed for a limited amount of time, this data will be the basis for the release of MOT19 in late 2019 [28].

KITTI. While the MOTChallenge datasets focus on pedestrian tracking, the KITTI tracking benchmark[6] [29, 30] allows for tracking of both people and vehicles. The dataset was collected by driving a car around a city and it was released in 2012. It consists of 21 training videos and 29 test ones, with a total of about 19000 frames (32 minutes). It includes detections obtained using the DPM[7] and RegionLets[8] [31] detectors, as well as stereo and laser information; however, as explained, in this survey we are only going to focus on models using 2D images. The CLEAR MOT metrics, MT, ML, ID switches and fragmentations are used to evaluate the methods. It is possible to submit results only for pedestrians or only for cars, and two different leaderboards are maintained for the two classes.

Other datasets. Besides the previously described datasets, there is a number of older, and now less frequently used, ones. Among those we can find the UA-DETRAC tracking benchmark[9] [32], that focuses on vehicles tracked from traffic cameras, and the TUD[10] [33] and PETS2009[11] [34] datasets, that both focus on pedestrians. Many of their videos are now part of the MOTChallenge datasets.

3 Deep learning in MOT

As this survey focuses on the use of deep learning in the MOT task, we organize this section into five subsections. Each of the first four subsections provides a review on how deep learning is exploited in each one of the four MOT stages

---

[2]Dataset: https://motchallenge.net/data/2D_MOT_2015/, leaderboard: https://motchallenge.net/results/2D_MOT_2015/.

[3]Dataset: https://motchallenge.net/data/MOT16/, leaderboard: https://motchallenge.net/results/MOT16/.

[4]Dataset: https://motchallenge.net/data/MOT17/, leaderboard: https://motchallenge.net/results/MOT17/.

[5]https://motchallenge.net/workshops/bmtt2019/tracking.html

[6]http://www.cvlibs.net/datasets/kitti/eval_tracking.php

[7]The website says the detections were obtained using a model based on a latent SVM, or L-SVM. That model is now known as Deformable Parts Model (DPM).

[8]http://www.xiaoyumu.com/project/detection

[9]https://detrac-db.rit.albany.edu/Tracking

[10]https://www.d2.mpi-inf.mpg.de/node/428

[11]http://www.cvg.reading.ac.uk/PETS2009/a.html

defined previously[12]. Subsection 3.4, besides presenting the use of deep learning in the association process, will also include its use in the overall track management process (e.g. initialization/termination of tracks), since it is strictly linked to the association step. Subsection 3.5 will finally describe uses of deep learning in MOT that do not fit into the four-step scheme.

We have included a summary table in Appendix A that shows the main techniques used in each of the four steps in each paper presented in this survey. The mode of operation (batch vs. online) is indicated and a link to the source code or to other provided material is also included (when available).

3.1 DL in detection step

While many works have used as input to their algorithms dataset-provided detections generated by various detectors (for example Aggregated Channel Features [24] for MOT15 [15] or Deformable Parts Model [25] for MOT16 [16]), there have also been algorithms that integrated a custom detection step, that often contributed to improve the overall tracking performance by enhancing the detection quality.

As we will see, most of the algorithms that employed custom detections made use of Faster R-CNN and its variants (section 3.1.1) or SSD (section 3.1.2), but approaches that used different models also exist (section 3.1.3). Despite the vast majority of algorithms utilized deep learning models to extract rectangular bounding boxes, a few works made a different use of deep networks in the detection step: these works are the focus of section 3.1.4.
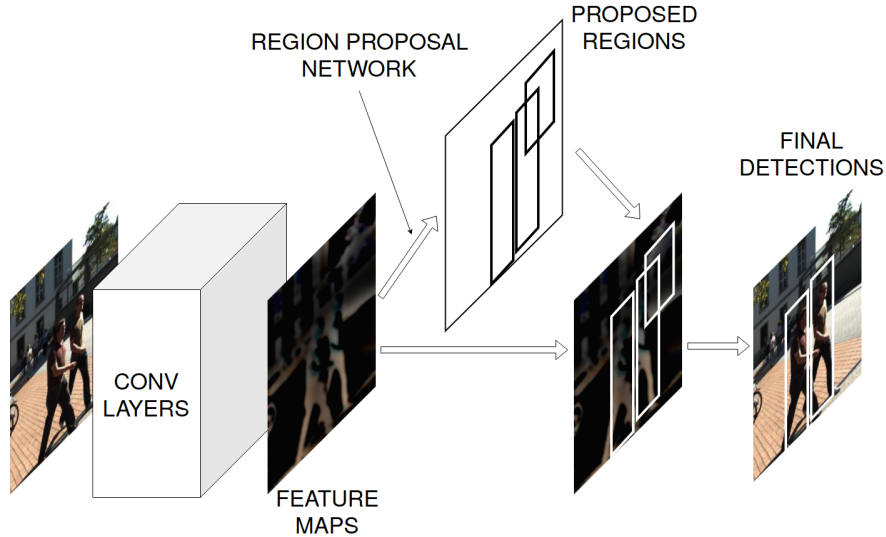

Figure 3: Example of a deep learning based detector (Faster R-CNN architecture [4])

3.1.1 Faster R-CNN

The Simple Online and Realtime Tracking (SORT) algorithm [35] has been one of the first MOT pipelines to leverage convolutional neural networks for the detection of pedestrians. Bewley et al. showed that replacing detections obtained using Aggregated Channel Features (ACF) [24] with detections computed by Faster R-CNN [4] (illustrated in figure 3) could improve the MOTA score by 18.9% (absolute change) on the MOT15 dataset [15]. They used a relatively simple approach that consisted in predicting object motion using the Kalman filter [36] and then associating the detections together with the help of the Hungarian algorithm [37], using intersection-over-union (IoU) distances to compute the cost matrix. At the time of publishing, SORT was ranked as the best-performing open source algorithm on the MOT15 dataset.

---

[12]Note that the classification of the models should not be considered as a strict categorization, since it's not rare that one of them has been used for multiple purposes and drawing a line is sometimes difficult. For example, some deep learning models, Siamese networks in particular, are often trained to output an affinity score, but at inference time they are only used to extract 'association features', and a simple hardcoded distance measure is then used instead to compute the affinities. In those cases, we decided to consider the network as performing feature extraction, since the similarity measure is not directly learned. However, those models could have also been considered to use deep learning for affinity computation.

Yu et al. reached the same conclusions in [38] using a modified Faster R-CNN, that included skip-pooling [39] and multi-region features [40] and that was fine-tuned on multiple pedestrian detection datasets. With this architecture they were able to improve the performance of the algorithm they proposed (see section 3.2.2) by more than 30% (absolute change, measured in MOTA), reaching state-of-the-art performance on the MOT16 dataset [16]. They also showed that having higher-quality detections reduces the need of complex tracking algorithms while still obtaining similar results: this is because the MOTA score is heavily influenced by the amount of false positives and false negatives, and using accurate detections is an effective way of reducing both. The detections computed by [38] on the MOT16 dataset have also been made available to the public[13] and many MOT algorithms have since exploited them [41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51].

In the following years, other works have taken advantage of the detection accuracy of Faster R-CNN, that has since been applied as part of MOT algorithms to detect athletes [52], cells [53] and pigs [54]. Moreover, an adaptation of Faster R-CNN that adds a segmentation branch, Mask R-CNN [17], has been used for example by Zhou et al. [55] both to detect and to track pedestrians,

3.1.2 SSD

The SSD [5] detector is another commonly used network in the detection step. In particular, Zhang et al. [54] compared it with Faster R-CNN and R-FCN [18] in their pig tracking pipeline, showing that it worked better on their dataset. They employed a Discriminative Correlation Filters (DCF) based online tracking method [56] with the use of HOG [57] and Colour Names [58] features to predict the position of so-called *tag-boxes*, small regions around the center of each animal. The Hungarian algorithm was used for the association between tracked tag-boxes and detections, and in the case of tracking failure the output of the DCF tracker was used to refine the bounding boxes. Lu et al. [59] also used SSD, but in this case to detect a variety of object classes to track (people, animals, cars, etc., see section 3.2.4).

Some works have tried to refine the detections obtained with SSD by taking into account the information obtained in other steps of the tracking algorithm. Kieritz et al. [60], in their joint detection and tracking framework, used the affinity scores computed between tracks and detections to replace the standard Non-Maximum Suppression (NMS) step included in the SSD network with a version that refines detection confidence scores based on their correspondence to tracked targets.

Zhao et al. [61] instead employed the SSD detector to search for pedestrians and vehicles in a scene, but they used a CNN-based Correlation Filter (CCF) to allow SSD to generate more accurate bounding boxes. The CCF exploited PCA-compressed [62] CNN features to predict the position of a target in the subsequent frame; the predicted position was then used to crop a ROI (Region Of Interest) around it, that was given as input to SSD. In that way, the network was able to compute small detections using deeper layers, that extract more valuable semantic information and that are thus known to produce more accurate bounding boxes and less false negatives. The algorithm then combined these detections with the ones obtained on the full image with a NMS step and then association between tracks and detections was performed using the Hungarian algorithm, with a cost matrix that took into account geometric (IoU) and appearance (Average Peak-to-Correlation Energy - APCE [63]) cues. APCE was also used for an object re-identification (ReID) step, to recover from occlusions. The authors showed that training a detector with multi-scale augmentation could lead to much better performance in tracking and the algorithm reached accuracy comparable to state-of-the-art online algorithms on KITTI and MOT15.

3.1.3 Other detectors

Among the other CNN models used as detectors in MOT, we can mention the YOLO series of detectors [64, 6, 65]; in particular, YOLOv2 has been used by Kim et al. [66] also to detect pedestrians. Sharma et al. [67] used instead a Recurrent Rolling Convolution (RRC) CNN [68] and a SubCNN [69] to detect vehicles in videos recorded on a moving camera in the context of autonomous driving (see section 3.2.4). Pernici et al. [70] used the Tiny CNN detector [71] in their face tracking algorithm, obtaining a better performance when compared to the Deformable Parts Model detector (DPM) [25], that does not use deep learning techniques.

3.1.4 Other uses of CNNs in the detection step

Sometimes CNNs have been employed in the MOT detection step for uses other than directly computing object bounding boxes.

For example, CNNs have been exploited to reduce false positives in [72], where vehicle detections were obtained with a modified version of the ViBe algorithm [73] that performed background subtraction on the input. These detections were

---

[13]https://drive.google.com/file/d/0B5ACiy41McAHMjczS2p0dFg3emM/view

first given as input to a SVM [74] and, in case the SVM was not confident enough to either discard or confirm them, a Faster-CNN based network [75] would then be used to decide whether to keep or discard each of them. In this way, only a few objects would have to be analyzed by the CNN, making the detection step faster.

Bullinger et al. explored a different approach in [76], where instead of computing classical bounding boxes in the detection step, a Multi-task Network Cascade [77] was instead employed to obtain instance-aware semantic segmentation maps. The authors argue that since the 2D shape of instances, differently from rectangular bounding boxes, do not contain background structures or parts of other objects, optical flow based tracking algorithms would perform better, especially when the target position in the image is also subject to camera motion in addition to the object's own motion. After obtaining the segmentation maps for the various instances present in the current frame, an optical flow method ([78, 79, 80]) was applied to predict the position and shape of each instance in the next frame. An affinity matrix between predicted and detected instances was then computed and given as input to the Hungarian algorithm for association. While the method obtained slightly lower MOTA score on the whole MOT15 dataset when compared to SORT, the authors showed that it performed better on videos with moving camera.

3.2 DL in feature extraction and motion prediction

The feature extraction phase is the preferred one for the employment of deep learning models, due to their strong representational power that makes them good at extracting meaningful high-level features. The most typical approach in this area is the use of CNNs to extract visual features, as it is commented in section 3.2.2. Instead of using classical CNN models, another recurrent idea consists in training them as Siamese CNNs, using contrastive loss functions, in order to find the set of features that best distinguish between subjects. Those approaches are explained in section 3.2.3. Furthermore, some authors explored the capabilities of CNNs to predict object motion inside correlation filter based algorithms: these are commented in section 3.2.5. Finally, other types of deep learning models have been employed, usually including them in more complex systems, combining deep features with classical ones. They are explained in sections 3.2.4 (specifically for visual features) and 3.2.6 (for approaches that don't fit in the other categories).

3.2.1 Autoencoders: first usage of DL in a MOT pipeline

To the best of our knowledge, the first approach using deep learning in MOT was presented by Wang et al. [81] in 2014. They proposed a network of autoencoders stacked in two layers that were used to refine visual features extracted from natural scenes [82]. After the extraction step, affinity computation was performed using a SVM, and the association task was formulated as a minimum spanning tree problem. They showed that feature refinement greatly improved the model performance. However, the dataset on which the algorithm was tested is not commonly used and results are hardly comparable to other methods.

3.2.2 CNNs as visual feature extractors

The most widely used methods for feature extraction are based on subtle modifications of convolutional neural networks. One of the first uses of these models can be found in [83]. Here, Kim et al. incorporated visual features into a classical algorithm, called Multiple Hypothesis Tracking, using a pretrained CNN that extracted 4096 visual features from the detections, that were later reduced to 256 using PCA. This modification improved the MOTA score on MOT15 by more than 3 points. By the time that paper was submitted, it was the top ranked algorithm on that dataset. Yu el al. [38] used a modified version of GoogLeNet [2], pretrained on a custom re-identification dataset, built by combining classical person identification datasets (PRW [84], Market-1501 [85], VIPeR [86], CUHK03 [87]). Visual features were combined with spatial ones, extracted with a Kalman filter, and then an affinity matrix was computed.

Other examples of the use of CNNs for feature extraction can be found in [88], where a custom CNN was used to extract appearance features in a Multiple Hypothesis Tracking framework, in [89], whose tracker employed a pretrained region-based CNN [90], or in [91], where a CNN extracted visual features from fish heads, later combined with motion prediction from a Kalman Filter.

The SORT algorithm [35], presented in section 3.1.1, was later refined with deep features, and this new version was called DeepSORT [41]. This model incorporated visual information extracted by a custom residual CNN [92]. The CNN provided a normalized vector with 128 features as output, and the cosine distance between those vectors was added to the affinity scores used in SORT. The experiments showed that this modification overcame the main drawback of the SORT algorithm, which was a high number of ID switches.

Mahmoudi el al. [42] also incorporated CNN extracted visual features along with dynamic and position features, and then solved the association problem via Hungarian algorithm. In [93], a ResNet-50 [3] pretrained on ImageNet was used as visual feature extractor. An extensive explanation of how a CNN can be used to distinguish pedestrians can be

found in [94]. In their model, Bae et al. combined the output of the CNN with shape and motion models, and computed an aggregated affinity score for each pair of detections; the association problem was then solved by the Hungarian algorithm. Again, Ullah et al. [95] applied an off-the-shelf version of GoogLeNet [2] for feature extraction. Fang et al. [96] selected as visual features the output of a hidden convolutional layer of an Inception CNN [97]. Fu et al. [98] employed the DeepSORT feature extractor, and measured the correlation of features using a discriminative correlation filter. Afterwards, the matching score was combined with a spatio-temporal relation score, and the final score was used as a likelihood in a Gaussian Mixture Probability Hypothesis Density filter [99]. The authors in [100] used a fine-tuned GoogLeNet on the ILSVRC CLS-LOC [101] dataset for pedestrians recognition. In [70], the authors reused the visual features extracted by the CNN-based detector, and the association was performed using a Reverse Nearest Neighbor technique [102]. Sheng et al. [103] employed the convolutional part of GoogLeNet to extract appearance features, using the cosine distance between them to compute an affinity score between pairs of detections, and merging that information with motion prediction in order to compute an overall affinity which serves as edge cost in a graph problem. Chen et al. [104] utilized the convolutional part of ResNet to build a custom model, stacking a LSTM cell on top of the convolutions, in order to compute simultaneously a similarity score and a bounding box regression.

In [53], the model learned to distinguish fast moving cells from slow moving cells. After the classification was computed, slow cells were associated using only motion features, since they were almost still, while fast cells were associated using both motion features and visual features extracted by a Fast R-CNN based on VGG-16 [1], specifically fine-tuned for the cell classification task. Moreover, the proposed model included a tracking optimization step, where false negatives and false positives were reduced by combining possible tracklets that were mistakenly interrupted.

Ran et al. [52] proposed a combination of a classical CNN for visual features extraction and AlphaPose CNN for pose estimation. The output of these two networks was then fed into a LSTM model together with the tracklet information history to compute a similarity, as it is explained in section 3.3.1.

An interesting employment of CNNs in feature extraction can be found in [51]. The authors used a pose detector, called DeepCut [105], that was a modification of Fast R-CNN; its output consisted in score maps predicting the presence of fourteen body parts. These were combined with the cropped images of detected pedestrians and fed into a CNN. A more detailed explanation of the algorithm is available in section 3.3.6.

3.2.3 Siamese networks
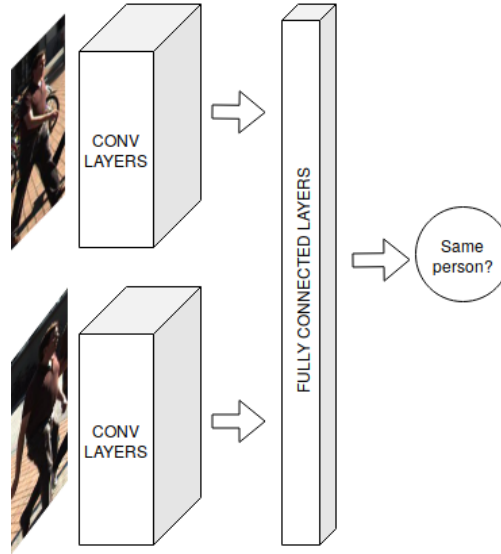
Figure 4: Example of a Siamese CNN architecture. For feature extraction, the network is trained as a Siamese CNN, but at inference time the output probability is discarded, and the last fully connected layer is used as feature vector for a single candidate. When the network is used for affinity computation, the whole structure is preserved during inference.

Another recurrent idea is training CNNs with loss functions that combine information from different images, in order to learn the set of features that best differentiates examples of different objects. These networks are usually called Siamese networks (an example of the architecture is shown in figure 4). Kim et al. [106] proposed a Siamese network [107] which was trained using a contrastive loss. The network took two images, their IoU score and their area ratio

as input, and produced a contrastive loss as output. After the net was trained, the layer that computed the contrastive loss was removed, and the last layer was used as a feature vector for the input image. The similarity score was later computed by combining the Euclidean distance between feature vectors, the IoU score and the area ratio between bounding boxes. The association step was solved using a custom greedy algorithm. Wang et al. [108] also proposed a Siamese network which took two image patches and computed a similarity score between them. The score at test time was computed comparing the visual features extracted by the network for the two images, and including temporally constrained information. The distance employed as similarity score was a Mahalanobis distance with a weight matrix, also learned by the model.

Zhang et al. [109] proposed a loss function called SymTriplet loss. According to their explanation, during the training phase three CNNs with shared weights were used, and the loss function combined the information extracted from two images belonging to the same object (positive pair) and from an image of a different one (two negative pairs). The SymTriplet loss decreased when the distance between the feature vectors of the positive pair was small, and increased when the negative pairs' features were close. Optimizing that function resulted in very similar feature vectors for images of the same object, while producing different vectors for different objects, with a larger distance between them. The dataset on which the tracking algorithm was tested was made of chapters from TV series and music videos from YouTube. Since the videos included different shots, the problem was divided into two stages. First, data association between frames in the same shot were performed. The affinity score in that case was a combination between the Euclidean distance of the feature vectors from the detections, temporal and kinematic information. Afterwards, tracklets were linked across shots, using a Hierarchical Agglomerative Clustering algorithm working over the appearance features.

Leal-Taixé et al. [110] proposed a Siamese CNN which received two stacked images as an input, and output the probability of both images belonging to the same person. They trained the network with this output so that it learned the most representative features to distinguish subjects. Afterwards, the output layer was removed and the features extracted by the last hidden layer were used as input for a Gradient Boosting model, together with contextual information, in order to get an affinity score between detections. Then, the association step was solved using Linear Programming [111].

Son et al. [112] proposed a new CNN architecture, called Quad-CNN. This model received as input four image patches, where the first three of them were from the same person, but in increasing time order, and the last one from another person. The network was trained using a custom loss, combining information about temporal distances between detections, extracted visual features, and bounding box positions. During the test phase, the network took two detections, and predicted the probability that both detections belonged to the same person, using the learned embedding.

In [55] a Siamese network based on Mask R-CNN [17] was built. After the Mask R-CNN had produced the mask for each detection, three examples were fed into the shallow Siamese net, two from the same object (positive pair) and one from another object (negative pair), again, and a triplet loss was used for training. After the training phase, the output layer was removed, and a 128-d vector was extracted from the last hidden layer. The appearance similarity was then computed using the cosine distance. That similarity was further combined with a motion consistency, which consisted on a score based on the predicted position of the object, assuming linear motion, and with a spatial potential, which was a more complex motion model. The association problem was then solved with a power iteration over a 3-d tensor of computed similarities.

Maksai et al. [113] directly used the 128-d feature vector extracted by the ReID triplet CNN proposed in [114], and combined it with other appearance-based features (as an alternative to an appearance-less version of the algorithm). Those features were further processed by a bidirectional LSTM. In [115] a similar approach was followed, with a so-called Spatial Attention Network (SAN). The SAN was a Siamese CNN, which used a pretrained ResNet-50 as base model. That net was truncated so that only the convolutional layers were employed. Then, a Spatial Attention Map was extracted from the last convolutional layers of the model: it represented a measure of the importance of different parts in the bounding box, in order to exclude background and other targets from the extracted features. The features were in fact weighted by this map, acting as a mask. The masked features from both detections were then merged into a fully connected layer which computed the similarity between them. During training, the network was also set to output a classification score, because the authors observed that jointly optimizing classification and affinity computation tasks resulted in a better performance in the latter. The affinity information was further fed into a bidirectional LSTM, as in the previous example. Both will be further discussed in section 3.3. Ma et al. [116] also trained a Siamese CNN in order to extract visual features from tracked pedestrians in their model, which is explained in detail in section 3.4.1.

In [117], Zhou et al. proposed a visual displacement CNN, which learned to predict the next position of an object depending on previous positions of the objects, and the influence that an object had over other objects in the scene. That CNN was then used to predict the location of objects in the next frame, taking as input their past trajectories. The

network was also capable of extracting visual information from the predicted positions and the actual detections, in order to compute a similarity score, as it is explained in section 3.3.6.

Chen et al. [118] proposed a two-steps algorithm which employed GoogLeNet trained with triplet loss for feature extraction. In the first step, the model used a R-FCN to predict possible detection candidates using information from the existing tracklets. Then, those detections were combined with the actual detections and NMS was performed. Afterwards, using the customly trained GoogLeNet model, they extracted visual features from the detections, and solved the association problem with a hierarchical association algorithm. When their paper was published, the algorithm was ranked on top among online methods in the MOT16 dataset.

Lee et al. [119] recently explored an interesting approach, combining pyramid and Siamese networks together. Their model, called Feature Pyramid Siamese Network, employed a backbone network (they studied the performance using SqueezeNet [120] and GoogLeNet [2], but the backbone network can be changed), which extracted visual features from two different images using the same parameters. Afterwards, some of the hidden feature maps from the network were extracted and given to the Feature Pyramid Siamese Network. The network then employed an upsampling and merging strategy to create a feature vector for every stage of the pyramid. Deeper layers were merged with shallower ones in order to enrich the simpler features with more complex ones. Afterwards, affinity score computation was performed, as explained in section 3.3.7.

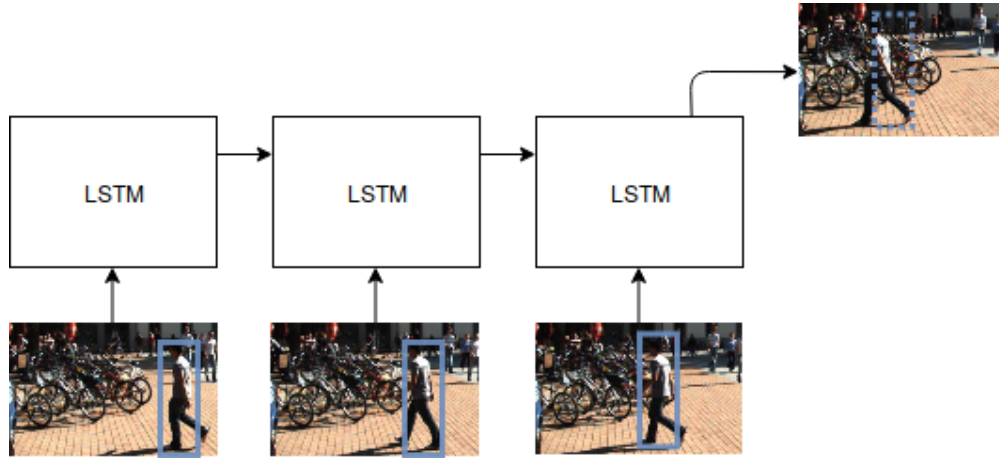### 3.2.4 More complex approaches for visual feature extraction

Figure 5: Typical usage of LSTM for motion prediction. A group of bounding boxes are fed into the network, and the produced output is the predicted bounding box in the next frame

More complex approaches have also been proposed. Lu et al. [59] employed the class predicted by the SSD in the detection step as a feature, and combined it with an image descriptor extracted with RoI pooling for each detection. Afterwards, the extracted features were used as input for a LSTM network, which learned to compute association features for the detections. Those features were later used for affinity computation, using the cosine distance between them.

In [121], the shallower layers of GoogLeNet were employed to learn a dictionary of features of the tracked objects. In order to learn the dictionary, the algorithm randomly selected objects in the first 100 frames of the video. The model extracted the feature maps in the first seven layers of the network. Then a dimensionality reduction was performed using Orthogonal Matching Pursuit (OPM) [122] over the features extracted from the objects, and the learned representation was used as a dictionary. During the test phase, the OPM representation was computed for every detected object in the scene, and compared with the dictionary in order to construct a cost matrix, combining visual and motion information extracted by a Kalman filter. Finally, the association was performed using the Hungarian algorithm.

LSTMs are sometimes employed for motion prediction, in order to learn more complex, non-linear motion models from the data. In figure 5 a scheme of the typical use of LSTMs for motion prediction is shown. An example of this use of recurrent networks is shown by Sadeghian et al. [123], who proposed a model that employed three different RNNs to compute various types of features, not only motion ones, for each detection. The first RNN was employed to extract appearance features. The input of this RNN was a visual features vector extracted by a VGG CNN [1], pretrained specifically for person re-identification. The second RNN was a LSTM trained to predict the motion model for every tracked object. In this case, the output of the LSTM was the velocity vector of each object. The last RNN was trained to

learn the interactions between different objects on the scene, since the position of some objects could be influenced by the behavior of surrounding items. The affinity computation was then performed by another LSTM, taking the information of the other RNNs as input.

In [124], a model of stacked CNNs was proposed. The first section of the model consisted of a pretrained shared CNN which extracted common features for every object in the scene. That CNN was not updated online. Then, a RoI pooling was applied and the RoI features for every candidate were extracted. Afterwards, for every tracked candidate a new specific CNN was instantiated and trained online. Those CNNs extracted both the visibility map and the spatial attention map for its candidate. Finally, after the refined features were extracted, the probability of each new image belonging to every already tracked object was computed, and the association step was finally performed using a greedy algorithm.

Sharma et al. [67] designed a set of cost functions to compute similarity between detections of vehicles. Those costs combined appearance features, extracted by a CNN, with 3D shape and position features assuming an environment with a moving camera. The defined costs were a 3D-2D cost, were the estimated 3D projection of the bounding box on the previous frame was compared with the 2D bounding box on the new frame, a 3D-3D cost, were the 3D projection of the previous bounding box was overlapped with the 3D projection of the current bounding box, an appearance cost, were the euclidean distance of the extracted visual features was computed, and a shape and pose cost, were the rough shape and position of the object in the bounding boxes were computed and compared. Note that while 3D projections were inferred, the input was still 2D images. After every cost was computed, the final pairwise cost between detections in two subsequent frames was a linear combination of the former costs. The final association problem was solved using the Hungarian algorithm.

Kim et al. [66] employed the information extracted by the YOLOv2 CNN object detector to build a random ferns classifier [125]. The algorithm worked in two steps. In the first step, a so-called teacher-RF was trained in order to differentiate pedestrians from non-pedestrians. After the teacher-RF was trained, for every tracked object, a random ferns classifier was constructed. Those classifiers were called student-RF, and they were smaller than the teacher-RF. They were specialized in distinguishing their tracked object from the rest of the objects in the scene. The decision of having a small random ferns classifier for every object was taken in order to reduce the computational complexity of the overall model, so that it could work in real time.

In [126] the number of affinity computations that the model must compute was reduced by estimating first the position of objects in subsequent frames, using a Hidden Markov Model [127]. Then, the feature extraction was performed using a pretrained CNN. After the visual features were extracted, the affinity computation was only computed between feasible pairs, that is, between detections close enough to the HMM prediction to be considered as the same object. The affinity score was obtained using a mutual information function between visual features. When the affinity scores were computed, a dynamic programming algorithm was used to associate detections.

3.2.5 CNNs for motion prediction: correlation filters

Wang et al. [128] studied the employment of a correlation filter [129], whose output is a response map for the tracked object. That map was an estimation of the new position of the object in the next frame. Such affinity was further combined with an optical flow affinity, computed using the Lucas-Kanade algorithm [130], a motion affinity calculated with a Kalman filter, and a scale affinity, represented by a ratio involving height and width of the bounding boxes. The affinity between two detections was computed as a linear combination of the previous scores. There was also a step of false detections removal, using a SVM classifier, and missing detections handling, using for that task the response map calculated in the previous steps. If an object was mistakenly lost and then re-identified, that step could fix the mistake and reconnect the broken tracklet.

In [61], a correlation filter was also employed to predict the position of the object in subsequent frames. The filter received as input the appearance features extracted by a CNN, previously reduced using PCA, and produced a response map of the predicted position for the object in the next frame as output. The predicted position was later used to compute a similarity score, combining the IoU between prediction and detections, and the APCE score of the response map. After the cost matrix was constructed, computing said score for every pair of detections between frames, the assignment problem was solved using the Hungarian algorithm.

3.2.6 Other approaches

Rosello et al. [131] explored a completely different approach, using a reinforcement learning framework to train a set of agents that helped in the feature extraction step. The algorithm was based solely on motion features, without any visual information employed. The motion model was learned using a Kalman filter, whose behavior was managed by an agent, using one agent for each tracked object. The agent learned to decide which action should the Kalman filter take, between a set of actions that included ignoring the prediction, ignoring the new measure, using both information pieces,

starting or stopping a track. The authors claimed that their algorithm could solve the tracking task even in non-visual scenarios, in contrast with classical algorithms whose performance was deeply influenced by visual features. However, the experimental results on MOT15 are not reliable and cannot be compared with other models because the model was tested on the training set.

Another algorithm that relied solely on motion features was the one proposed in [132]. Babaee et al. presented a LSTM which learned to predict the new position and size of the bounding box for every object in the scene, using information about position and velocity in previous frames. Using the IoU between the predicted bounding box and the real detection, an affinity measure was computed, and the tracks were associated using a custom greedy algorithm. The pipeline was applied on tracking results obtained by other algorithms, in order to handle occlusions, and the authors showed that their method could effectively reduce the number of ID switches.

3.3 DL in affinity computation

While many works compute affinity between tracklets and detections (or tracklets and other tracklets) by using some distance measure over features extracted by a CNN, there are also algorithms that use deep learning models to directly output an affinity score, without having to specify an explicit distance metric between the features. This section focuses on such works.

In particular, we are first going to describe algorithms that used recurrent neural networks, starting from standard LSTMs (section 3.3.1) and then describing uses of Siamese LSTMs (section 3.3.2) and Bidirectional LSTMs (section 3.3.3). A particular use of LSTM-computed affinities in the context of Multiple Hypothesis Tracking (MHT) frameworks is presented in section 3.3.4; finally, a few works that employed different kinds of recurrent network for affinity computations are presented in section 3.3.5.

In the second part of this section we are going to explore instead the uses of CNNs in affinity computation (section 3.3.6), including the algorithms that used the output of Siamese CNNs directly as an affinity score (section 3.3.7), instead of relying on distance measures over feature vectors like in section 3.2.3.

3.3.1 Recurrent neural networks and LSTMs

One of the first works to use deep network to directly compute an affinity is [133], where Milan et al. proposed an end-to-end learning approach for online MOT. A recurrent neural network (RNN) based model was used as the main tracker, mimicking a Bayesian filter algorithm, consisting of three blocks: the first was a motion prediction block, that learned a motion model that took as input the state of the target in the past frames (i.e. the old bounding box locations and sizes) and predicted the target state in the next frame without accounting for the detections; the second block refined the state prediction using the detections in the new frame and an association vector containing the probability of associating the target with all such detections (it is evident how this can be considered an affinity score); the third block managed the birth and death of tracks, as it used the previous collected information to predict the probability of existence of the track in the new frame[14]. The association vector was computed using a LSTM-based network, that used the Euclidean distance between the predicted state of the target and the states of the detections in the new frame as input features (besides the hidden state and the cell state, as any standard LSTM). The networks were trained separately using 100K 20-frame long synthetically generated sequences. While the algorithm performed favorably to other techniques, like the combination of a Kalman filter with the Hungarian algorithm, the results on the MOT15 test set did not quite reach top accuracy; however, the algorithm was able to run much faster than other algorithms ($\sim$ 165 FPS) and did not use any kind of appearance features, leaving room for future improvements.

Among the other works that later used LSTMs there is [123], that used a LSTM with a fully-connected (FC) layer to fuse features extracted by 3 other LSTMs (as already explained in section 3.2.4) and output an affinity score[15]. The overall algorithm is similar to the Markov Decision Processes (MDP) based framework presented in [134]: a single object tracker (SOT) is used to track targets independently; when a target gets occluded, the SOT is stopped and a bipartite graph is built that uses the affinities computed by the LSTM as edge costs and the association problem is then solved with the help of the Hungarian algorithm. The authors showed that using both a combination of all the 3 feature extractors and an LSTM rather than a plain FC layer led to consistently better performance on a MOT15 validation

---

[14]To smooth the existence probability predictions and avoid deleting tracks of temporarily occluded objects, the difference between the new and the old existence probability was also output so that it could be minimized during training.

[15]The paper seems to imply that while the LSTM is trained to predict an affinity score, only the affinity features are extracted and are then used to replace the handcrafted features used in the MDP paper. The algorithm presented in the MDP paper, though, adds on top of those features another FC layer, trained with reinforcement learning to classify the track/detection pair as belonging to the same identity or not. Thus we can consider the overall affinity computation as performed by a deep learning model.

set. The algorithm also reached state-of-the-art MOTA scores on both MOT15 and MOT16 test sets at the time of publication, confirming the validity of the approach.

Another approach using multiple LSTMs is [52], where Ran et al. proposed a Pose-based Triple Stream Network, that computed an affinity combining 3 other affinities output by 3 LSTMs: one for appearance similarity, using CNN features and pose information extracted with AlphaPose [135], one for motion similarity, using pose joints velocity, and one for interaction similarity, using an interaction grid. A custom tracking algorithm is then used to associate the detections. The comparison with other state-of-the-art MOT algorithms on their proprietary Volleyball dataset for athlete tracking was favourable.

3.3.2 Siamese LSTMs

Liang et al. [136] also used multiple LSTMs to model various features, but they proceeded in a different way. Since extracting appearance features with CNNs is computationally expensive, they proceeded with a so-called *pre-association step*, that used a SVM to predict the association probability between tracklets and detections. The SVM took as input position and velocity similarity scores, computed using two LSTMs for position and velocity prediction. The pre-association step then consisted in discarding the detections with low SVM affinity scores. After this step, an actual association step was performed by using VGG-16 features given as input to a Siamese LSTM, that predicted an affinity score between the tracklet and the detections. Association was performed in a greedy manner, associating the detection with the highest score to the tracklet. Testing was done on the MOT17 datasets, and the results were in line with the top-performing algorithms.

Wan et al. [43] also used a Siamese LSTM in their algorithm, that was also composed of two steps. In the first step, short reliable tracklets were built by using Hungarian algorithm with affinity measures computed using the IoU between detections and the predicted target positions (obtained with Kalman filter or Lucas-Kanade optical flow). The second step also used the Hungarian algorithm to join the tracklets, but this time the affinity was computed using a Siamese LSTM framework that used motion features concatenated to appearance features extracted by a CNN (like in [137]), pre-trained on the CUHK03 Re-ID dataset.

3.3.3 Bidirectional LSTMs

A different usage of LSTMs in the affinity computation phase was presented by Zhu et al. [115]. They used a so-called Temporal Attention Network (TAN) to compute attention coefficients to weigh the features extracted by the Spatial Attention Network (SAN) (see section 3.2.3) to give less importance to noisy observations. A bidirectional LSTM was employed to this end. The whole network (called Dual Matching Attention Network) was used to recover from occlusions when a modified version of the Efficient Convolution Operators tracker (ECO) [56], trained exploiting hard example mining, failed to detect a target. The algorithm obtained results comparable to online state-of-the-art methods on MOT16 and MOT17 according to various metrics (MOTA, IDF1, number of ID switches).

Yoon et al. [138] also used a Bidirectional LSTM to compute affinities, on top of some FC layers that encoded non-appearance features (only bounding box coordinates and detection confidences). The association was solved using the Hungarian algorithm. They trained the network on the Stanford Drone Dataset (SDD) [139] and evaluated it on both SDD and MOT15. They reached comparable results with top algorithms that did not use visual cues, but the performance was still worse than appearance-based methods.

3.3.4 Uses of LSTMs in MHT frameworks

In Multiple Hypothesis Tracking approaches, a tree of potential track hypotheses for each candidate target is first built. Then the likelihood of each track is computed and the combination of tracks that has the highest likelihood is chosen as a solution. Various deep learning algorithms have also been employed to enhance MHT based approaches.

Kim et al. [93] proposed the use of a so-called *Bilinear LSTM* network as the gating step of the MHT-DAM [83] algorithm, that is, the affinity score computed by the LSTM was used to decide whether to prune or not a certain branch of the hypotheses tree. The LSTM cell had a modified forward pass (inspired by the online recursive least squares estimator proposed in [83]) and took as input the appearance features of a tracklet in the past frames, extracted with a ResNet-50 CNN. The output of the LSTM cell was a feature matrix representing the historical appearance of the tracklet, and such matrix was then multiplied by the vector with the appearance features of the detection that needed to be compared with the tracklet. FC layers on top of that finally computed the affinity score between the tracklet and the detection. The authors claimed that such a modified LSTM is able to store longer-term appearance models than classical LSTMs. They also proposed to add a motion modeling classical LSTM to compute historical motion features (using the normalized bounding box coordinates and sizes), that were then concatenated to the appearance features before proceeding with the FC layers and the final softmax that output the affinity score. The two LSTMs were first

trained separately and then fine-tuned jointly. The training data was also augmented including localization errors and missing detections, to resemble real-world data more closely. They used MOT15, MOT17, ETH, KITTI and other minor datasets for training and they evaluated the model on MOT16 and MOT17. They showed that their model is sensitive to the quality of detections, as they improved on the MOTA performance of MHT-DAM when using the public Faster R-CNN and SDP detections, while they performed worse than it on the public DPM detections. Anyway, they seemed to get a higher IDF1 score regardless of the detections used, and their overall results reflected that, since they got the highest IDF1 of all the methods using MHT-based algorithms. However, the tracking quality, as measured both in MOTA and in IDF1, was still lower than other state-of-the-art algorithms.

A similar use of a RNN has been recently presented by Maksai et al. [113], who also employed a LSTM to compute tracklet scores in a variation of the MHT algorithm, that grows and prunes tracklets iteratively and then tries to select the set of tracklets that maximizes said score.[16] The goal of their work was to solve two frequent problems in training recurrent networks for multiple object tracking: the *loss-evaluation mismatch*, that arises when a network is trained by optimizing a loss that is not well-aligned to the evaluation metric used at inference time (e.g. classification score vs. MOTA); the *exposure bias*, that is present when the model is not exposed to its own errors during the training process. In order to solve the first problem, they introduced a novel way to score tracklets (using a RNN) that is a direct proxy of the IDF1 metric and does not use the ground truth; the network can then be trained to optimize such metric. The second problem was solved instead by adding to the training set of the network the tracklets computed using the current version of the network, together with hard example mining and random tracklet merging during training; in this way, the training set distribution should be more similar to the inference-time input data distribution. The network used was a Bidirectional LSTM, on top of an embedding layer that took as input various features. The authors presented a version of the algorithm using only geometric features and a version that instead used appearance-based features, that performed better. Lots of ablation studies were run, and various alternative approaches were tested. The final algorithm was able to reach top-performance on various MOT datasets (MOT15, MOT17, DukeMTMC [21]) when considering the IDF1 metric, even though it didn't excel in MOTA.

Among the other approaches in the MHT family using RNNs we can also find [104], where Chen et al. used a so-called *Recurrent Metric Network* (RMNet) to compute appearance affinity between tracklet hypotheses and detections (together with a motion-based affinity) in their Batch Multi-Hypothesis Tracking strategy. The RMNet is an LSTM that takes as input appearance features of the detection sequence under consideration, extracted with a ResNet CNN, and outputs a similarity score together with bounding box regression parameters. A dual-threshold approach in gating and forming hypothesis was used, and a re-find reward was employed to encourage recovery from occlusions. The hypotheses were selected by casting the problem as a binary linear programming one, solved using *lpsolve*. Kalman filter was finally used to smooth the trajectories. Evaluation was performed on MOT15, PETS2009 [34], TUD [140] and KITTI, obtaining better results on the IDF1 metric, that gives more weight to people re-identification, than on MOTA.

3.3.5 Other recurrent networks

Fang et al. [96] used instead Gated Recurrent Units (GRUs) [141] inside their Recurrent Autoregressive Network (RAN) framework for pedestrian tracking. The GRUs were used to estimate the parameters of autoregressive models, one for motion and the other for appearance for each tracked target, that computed the probability of observing a given detection motion/appearance based on the tracklet's past motion/appearance features. The two probabilities, that can be easily seen as a kind of affinity measure, were then multiplied together to obtain a final association probability, used to solve a bipartite matching problem for association between tracklets and detections following the algorithm in [134]. The RAN training step was formulated as a maximum likelihood estimation problem.

Kieritz et al. [60] used a recurrent 2-hidden-layer multi-layer perceptron (MLP) to compute an appearance affinity score between a detection and a tracklet. Such affinity was then given as input to another MLP, together with track and detection confidence scores, to predict an aggregate affinity score (called *association metric*). Such score was finally used by the Hungarian algorithm to perform association. The method reached top performance on the UA-DETRAC dataset [32], but the performance on MOT16 was not very good when compared with other algorithms using private detections.

3.3.6 CNNs for affinity computation

Other algorithms used instead CNNs to compute some kind of similarity score. Tang et al. [51] tested the use of 4 different CNNs to compute an affinity score between nodes in a graph, with the association task being formulated as a minimum cost lifted multicut problem [142]: it can be seen as a graph clustering problem, where each output

---

[16]While it is not an explicit affinity measure, it can still be seen as an evaluation of the effect of merging two tracklets and thus plays a similar role (i.e. taking decisions about associating tracklets) as other affinities presented in this section.

cluster represents a single tracked object. The costs associated to the edges accounted for the similarity between two detections. Such similarity was a combination of person re-identification confidence, deep correspondence matching and spatio-temporal relations. To compute the person re-identification affinity, various architectures were tested (after being trained on a dataset of 2511 identities extracted from MOT15, MOT16, CUHK03, Market-1501 datasets), but the best performing one was the novel StackNetPose. It incorporated body part information extracted using the DeepCut body part detector [105] (see section 3.2.2). The 14 score maps for the body parts of two images were stacked together with the two images themselves to produce a 20-channel input. The network followed the VGG-16 architecture and output an affinity score between the two input identities. Differently from Siamese CNNs, the pair of images were able to 'communicate' in the early stages of the network. The authors showed that the StackNetPose network performed better in the person re-identification task, and thus they used it to compute the ReID affinity. The combined affinity score was computed by multiplying a weight vector (learned with logistic regression, and dependent on the time interval between the two detections) with a 14-d vector containing ReID affinity, DeepMatching-based affinity [143], a spatio-temporal affinity score, the minimum of the two detection confidences and quadratic terms with all the pairwise combinations of the previously mentioned terms. The authors showed that combining all these features produced better results, and together with the improvement in framing the problem as a minimum cost lifted multicut problem (solved heuristically using the algorithm proposed in [144]), they managed to reach state-of-the-art performance (measured in MOTA score) on the MOT16 dataset at the time of publishing.

Another approach using CNNs was presented in [145], where Chen et al. used a Particle Filter [146] to predict target motion, weighting the importance of each particle using a modified Faster R-CNN network. Such model was trained to predict the probability that the bounding box contains an object, but it was also augmented with a target-specific branch, that took as input features from lower layers of the CNN and merged them with the target historical features to predict the probability of the two objects being the same. The difference with the previous approaches is that here the affinity is computed between the sampled particles and the tracked target, instead of being computed between targets and detections. The detections that did not overlap with the tracked objects were instead used to initialize new tracks or retrieve missing objects. Despite being an online tracking algorithm, it was able to reach top performance on MOT15 at the time of publishing, both when using public detections and when using private ones (obtained from [147]).

Zhou et al. [117] used a visual-similarity CNN, similar to the ResNet-101 based visual-displacement CNN presented in section 3.2.3, that outputs affinity scores between the detections and the tracklet boxes predicted by the Deep Continuous Conditional Random Fields. This visual affinity score was merged with a spatial similarity using IoU, and then the detection with the highest score was associated to each tracklet; in case of conflicts, the Hungarian algorithm was employed. The method reached results comparable to state-of-the-art online MOT algorithms on MOT15 and MOT16 in terms of MOTA score.

3.3.7 Siamese CNNs

Siamese CNNs are also a common approach used in affinity computation. An example of Siamese CNN is shown in figure 4. The approaches presented here decided to directly use the output of the Siamese CNN as an affinity, instead of employing classical distances between feature vectors extracted from the penultimate layer of the network, like the algorithms presented in section 3.2.3. For example, Ma et al. [148] used one to compute affinities between tracklets in a two-step algorithm. They chose to apply hierarchical correlation clustering, solving two successive lifted multicut problems: local data association and global data association. In the local data association step temporally-close detections were joined together by using the robust similarity measure presented in [149], that uses DeepMatching and detection confidences to compute an affinity score between detections. In this step, only edges between close detections were inserted into the graph. The multicut problem was solved with the heuristic algorithm proposed in [144]. In the global data association step, local tracks that were split by long-term occlusion needed to be joined together, and a fully-connected graph with all the tracklets was then built. The Siamese CNN was used to compute the affinities that would serve as edge costs in the graph. The architecture was based on GoogLeNet [2] and it was pretrained on ImageNet. The net was then trained on the Market-1501 ReID dataset and then fine-tuned on the MOT15 and MOT16 training sequences. Besides the verification layer, that output a similarity score between the two images, two classification layers were added to the network only during training to classify the identity of each training image; this was shown to improve the network performance in computing the affinity score. This so-called 'generic' ReID net was also fine-tuned on each test sequence in an unsupervised manner, without using any ground truth information, to adapt the net to the illumination conditions, resolution, camera angle, etc. of each particular sequence. This was done by sampling positive and negative detection pairs by looking at the tracklets built in the local data association step. The effectiveness of the algorithm was proven by the results obtained on MOT16, where it is at the time of writing the best performing method with a published paper, with a 49.3 MOTA score.

As explained in section 3.2.3, Lee et al. [119] used a Feature Pyramid Siamese Network to extract appearance features. When employing this kind of network in the MOT problem, a vector of motion features was concatenated to the

appearance features and 3 fully-connected layers were then added on top to predict an affinity score between a track and a detection; the network was trained end-to-end. Detections were then associated iteratively, starting from the pairs with highest affinity scores and stopping when the score got below a threshold. The method obtained top performance results among the online algorithms on the MOT17 dataset at the time of publishing.

3.4 DL in Association/Tracking step

Some works, albeit not as many as for the other steps in the pipeline, have used deep learning models to improve the association process performed by classical algorithms, like the Hungarian algorithm, or to manage the track status (e.g. by deciding to start or terminate a track). We are going to present them in this section, including the use of RNNs (section 3.4.1), deep multi-layer perceptrons (section 3.4.2) and deep reinforcement learning agents (section 3.4.3).

3.4.1 Recurrent neural networks

A first example of algorithms employing DL to manage the track status is the one presented by Milan et al. in [133], already described in section 3.3.1, that used a RNN to predict the probability of existence of a track in each frame, thus helping with the decision of when to initiate or terminate the tracks.

Ma et al. [116] used a bidirectional GRU RNN to decide where to split tracklets. The algorithm proceeded in three main stages: a tracklet generation step, that included a NMS step to remove redundant detections and then employed the Hungarian algorithm with appearance and motion affinity together to form high-confidence tracklets; then, a tracklet cleaving step was performed: since a tracklet might contain an ID switch error due to occlusions, this step aimed to split the tracklets at the point where the ID switch happened, in order to obtain two separate tracklets that contained the same identity; finally, a tracklet reconnection step was employed, using a customized association algorithm that made use of features extracted by a Siamese bidirectional GRU. The gaps within the newly-formed tracklets were then filled with polynomial curve fitting. The cleaving step was performed with a bidirectional GRU RNN, that used features extracted by a Wide Residual Network CNN [92]. The GRU output a pair of feature vectors for each frame (one for each direction of the GRU); then the distance between pairs of such feature vectors was computed and a distance vector was obtained. The highest value in this vector indicated where to split the tracklet, provided that the score was higher than a threshold. The reconnection GRU was similar, but it had an additional FC layer on top of the GRU and a temporal pooling layer to extract a feature vector representing the whole tracklet; the distance between the features of the two tracklets was then used to decide which tracklets to reconnect. The algorithm reached results comparable to state-of-the-art on the MOT16 dataset.

3.4.2 Deep Multi-Layer Perceptron

Despite not being a very common approach, deep multi-layer perceptrons (MLP) have also been employed to guide the tracking process. For example, Kieritz et al. [60] used a MLP with two hidden layers to compute track confidence scores, taking as input the track score at the previous step and various information about the last associated detection (like association score and detection confidence). This confidence score was then used to manage the termination of tracks: they decided in fact to keep a fixed number of targets through time, replacing with new tracks the older ones that had the lowest confidence scores. The rest of the algorithm has been explained in section 3.3.5.

3.4.3 Deep Reinforcement Learning agents

Some works have used Deep Reinforcement Learning (RL) agents to take decisions in the tracking process. Rosello et al. [131], as explained in section 3.2.6, used multiple deep RL agents to manage the various tracked targets, deciding when to start and stop tracks and influencing the operation of the Kalman filter. The agent was modeled with a MLP with 3 hidden layers.

Ren et al. [150] also used multiple deep RL agents in a collaborative environment to manage the association task. The algorithm was mainly composed of two parts: a prediction network and a decision network. The prediction network was a CNN that was learned to predict the movement of the target in the new frame looking at the target and at the new image, and also using the recent tracklet trajectory. The decision network was instead a collaborative system that consisted of multiple agents (one for each tracked target) and the environment. Each agent took decisions based on the information about themselves, the neighbours and the environment; the interactions between the agents and the environment were exploited by maximizing a shared utility function: the agents thus did not operate independently from each other. Every agent/object was represented by a trajectory, its appearance features (extracted using MDNet [151]) and its current position. The environment was represented by the detections in the new frame. The detection network took as input, for each target, its predicted location in the new frame (output by the prediction network), the nearest target and the nearest detection, and based on various factors, such as the detection reliability and the target occlusion

status, took one among various actions: updating the track and its appearance features using both the prediction and the detection, ignoring the detection and only using the prediction to update the track, detecting an occlusion of the tracked target, deleting the track. The agents were modelled using 3 FC layers on top of the feature extraction part of the MDNet. Various ablation studies showed the effectiveness of using the prediction and detection networks instead of linear motion models and Hungarian algorithm, respectively, and the method obtained very good results on the MOT15 and MOT16 datasets, reaching state-of-the-art performance among online methods, despite suffering from a relatively high number of ID switches.

3.5 Other uses of DL in MOT

In this section we will present other interesting uses of deep learning models that don't neatly fit into one of the four common steps of a multiple object tracking algorithm. For this reason, such works have not been included in table 7, but are summarized instead in table 1.

	Detection	Description	Mode	Source and data
[152]	N/A	They integrate a bounding box regression step in various existing MOT algorithms. The regression is done using Deep Reinforcement Learning using CNN features.	N/A	
[153]	Public	An ensemble of 2 CNNs, color histograms and a KLT motion detector are used to compute likelihoods for a Markov Chain Monte Carlo sampling; the position sampling was used to form short tracklets. A Changing Point Detection algorithm was employed to merge and delete tracklets.	Online	
[154]	CNN	Multi-Bernoulli Filter with a novel Interactive Likelihood, computed using a CNN.	Online	
[155]	Public	Body detections are refined using head detections obtained with a CNN [156]. A modified version of the Frank-Wolfe algorithm is used to solve a correlation clustering problem for association, using spatial and temporal costs.	Batch	
[157]	Public	Modified MDNet CNN with target-specific branches to compute affinities between targets and candidates extracted with Gaussian sampling. Combination of appearance and motion features to reduce ID Switches.	Online	
[158]	Public	CNN to extract app features and LSTM to extract motiono features. The LSTM is part of a BF-Net, that performs Bayesian filtering and uses the output from Hungarian algorithm for track refinement.	Online	
[159]	Public	PafNet and PartNet CNNs to distinguish targets from background and among themselves. KCF SOT tracker is used. SVM+Hungarian algorithm for error recovering. CNN trained with RL for model updating.	Online	

Table 1: Information summary about methods using DL that don't fit the 4-step scheme.

One example is [152], where Jiang et al. use a Deep RL agent to perform bounding box regression after the use of one of many MOT algorithms. The procedure is in fact completely independent from the tracking algorithm employed, and can be used a posteriori to increase the accuracy of the model. A VGG-16 CNN was used to extract appearance features from the region enclosed by the bounding box, then those features were concatenated to a vector representing the history of the last 10 actions taken by the agent. Finally a Q-network [160] made of 3 fully-connected layers was used to predict one among 13 possible actions, that included motion and scaling of the bounding box and a termination action, to signal the completion of the regression. The use of this bounding box regression technique on various state-of-the-art MOT algorithms allowed an improvement between 2 and 7 absolute MOTA points on the MOT15 dataset, reaching top score among public detections methods. The authors also showed that their regression approach had better results than using conventional methods, such as the bounding box regression computed by a Faster R-CNN model.

Lee et al. [153] proposed a multi-class multi-object tracker that used an ensemble of detectors, including CNN models like VGG-16 and ResNet, to compute the likelihood of each target being at a certain location in the next frame. A

Markov Chain Monte Carlo sampling from a distribution that was influenced by said likelihoods was used to predict the next position for each target, and together with an estimation of track birth and death probabilities, short track segments were built. Finally, a changing point detection [161] algorithm was employed to detect abrupt changes in stationary time series representing track segments; this was done in order to detect track drift, to remove unstable track segments and to combine the segments together. The algorithm reached results comparable to state-of-the-art MOT methods using private detections.

Hoak et al. [154] proposed a 5-layer custom CNN network, trained on the Caltech pedestrian detection dataset [162], to compute the likelihood of a target being at a certain location in the image. They used a multi-Bernoulli filter (implemented using the particle filter algorithm presented in [163]), and a novel Interactive Likelihood (ILH) was computed for each particle, in order to weigh them based on their distance from particles belonging to other targets; this was done to prevent the algorithm from sampling from areas that belong to different objects. The algorithm obtained good results on the VSPETS 2003 INMOVE soccer dataset[17] and the AFL dataset [164].

Henschel et al. [155] used head detections, extracted with a CNN [156], in addition to the usual body detections to perform pedestrian tracking. The presence/absence of a head and its position relative to the bounding box can help determine if a bounding box is a true or a false positive. The association problem was modelled as a correlation clustering problem on graphs, that the authors solved with a modified version of the Frank-Wolfe algorithm [165]; the association costs were computed as a combination of spatial and temporal costs: the spatial costs were the distance and the angle between the detected and the predicted head positions; the temporal costs were computed using the correspondences between pixels between the two frames, obtained using DeepMatching [79]. The algorithm reached top MOTA score on MOT17 and second-best score on MOT16 at the time of publication.

Gan et al. [157] employed a modified MDNet [151] in their online pedestrian tracking framework. Besides 3 shared convolutional layers, common to all the targets, each target also had 3 specific FC layers, that were updated online to capture the appearance change of the target. A set of box candidates, including detections intersecting the last bounding box of the target and a set of boxes sampled from a Gaussian distribution with parameters estimated using a linear motion model, were given as input to the network, that output a confidence score for each of them. The candidate with the highest score was considered the optimal estimated target location. To reduce the number of ID switch errors, the algorithms tried to find the past trajectory that was most similar to the estimated box, using another affinity measure between the pairs; such affinity was computed using appearance and motion cues, together with the tracklet confidence score and a collision factor. Detections were also used to initialize new tracklets and to fix the motion prediction errors when occlusions happened.

Xiang et al. [158] used a MetricNet to track pedestrians. The model unified an affinity model with trajectory estimation, done with a Bayesian filter. An appearance model, made of a VGG-16 CNN trained for person re-identification on various datasets, extracted features and performed bounding box regression; the motion model instead consisted of two parts: an LSTM-based feature extractor, that took as input the trajectory's past coordinates, and a so-called *BF-Net* on top, made of various FC layers, that combined the features extracted by the LSTM and a detection box (chosen by the Hungarian algorithm) to perform the Bayesian filtering step and output the new position of the target. The MetricNet was trained using a triplet loss, similar to other models presented in the previous sections. The algorithm obtained the best and second-best results among online methods on MOT16 and MOT15, respectively.

Finally, Chu et al. [159] used three different CNNs in their algorithm. The first one, called PafNet [166], was used to distinguish the background from the tracked objects. The second one, called PartNet [167], was employed to distinguish among the different targets. The third CNN, made of one convolutional layer and one FC layer, was instead used to decide whether to refresh the tracking model or not. The overall algorithm worked as follows: for every tracked target in the past frame, two score maps were computed in the current one, using PafNet and PartNet. Then, using the Kernel Correlation Filter tracker [168], a new position for the object was predicted. Moreover, after a certain number of frames, a so-called detection verification step was performed: the detections output by a detector (in their experiments, they chose to use the public detections provided with the dataset) were assigned to the tracked targets by solving a graph multicut problem. Targets that were not associated to a detection for a certain number of frames were terminated. Then, the third CNN was employed to check if the associated detection box was better than the predicted one. If so, the KCF model parameters were updated to reflect the change in the object characteristics. Such CNN used the maps extracted by PafNet, and was trained using reinforcement learning. Unassociated detections were then employed to recover from target occlusion, using a SVM classifier and the Hungarian algorithm. Finally, the remaining unassociated detections were used to initialize new targets. The algorithm was evaluated both on MOT15 and MOT16 datasets, reaching top performance overall on the first one, and top performance among online methods on the second.

---

[17]ftp://ftp.cs.rdg.ac.uk/pub/VS-PETS/

4 Analysis and comparisons

This section presents a comparison between all the works that have tested their algorithm on one of the MOTChallenge datasets. We will only focus on the MOTChallenge datasets since for other datasets there aren't enough relevant papers using deep learning to perform a meaningful analysis.

We first describe the setup of the experimental analysis, including the considered metrics and the organization of the tables in section 4.1. Section 4.2 will then present the actual results and considerations derived from the analysis.

4.1 Setup and organization

For a fair comparison, we only show results reported on the whole test sets. Some of the discussed papers report their results using subsets of the test set, or validation datasets extracted from the training splits of the MOTChallenge datasets. These results are discarded as they are not comparable with the others. Moreover, the reported results are divided into algorithms that use public detections and algorithms that use private detections, since the different quality of the detections has a big impact on performance. The results are further split into online and batch methods, since the online methods are at a disadvantage, being only able to access present and past information to assign IDs in each frame.

For each algorithm we indicate the year of the referenced published paper, their mode of operation (batch vs. online); the MOTA, MOTP, IDF1, Mostly Tracked (MT) and Mostly Lost (ML) metrics, expressed in percentages; the absolute number of false positives (FP), false negatives (FN), ID switches (IDS) and fragmentations (Frag); the speed of the algorithm expressed in frames per second (Hz). For each metric, an arrow pointing up ($\uparrow$) indicates that a higher score is better, while an arrow pointing down ($\downarrow$) indicates the opposite. The metrics shown here are the same that can be found on the public leaderboards on the MOTChallenge website. The numerical results presented in the referenced works have been integrated with data from the MOTChallenge leaderboards.

Attending to the classification presented before, a table for each of the combinations dataset/detection source is shown. Tables 2 and 3 show results on MOT15 using public and private detections respectively; tables 4 and 5 do the same on MOT16; finally, table 6 shows results on MOT17, who currently only has published algorithms that use public detections. Each table groups online and batch methods separately, and for each group the papers are sorted by year, and then by ascending MOTA score if the papers are from the same year, since it is the main metric considered in the MOTChallenge datasets[18]. If a work presents multiple results on the same dataset, using the same set of detections and the same mode of operation, we only show the result with the highest MOTA. The best performance for each metric is highlighted in **bold**, while the best performance among papers operating in the same mode (batch/online) is underlined. It is important to note though that comparisons on the Hz metric may not be reliable since the performance is usually reported only for the tracking part of the algorithms, without the detection step and sometimes without including the runtime of deep learning models, that are usually the most computational intensive part of the algorithms presented in this survey; moreover, the algorithms were run on widely different hardware.

4.2 Discussion of the results

General observations

As expected, the best performing algorithms on each dataset use private detections, confirming the fact that the detection quality dominates the overall performance of the tracker: 56.5% MOTA vs. 42.3% for MOT15 and 71.0% vs. 49.3% for MOT16. Moreover, on MOT16 and MOT17 the batch algorithms slightly outperform the online ones, even though the online methods are progressively getting closer to the performance of the batch ones. In fact, on MOT15 the best reported algorithm that uses deep learning runs in an online fashion. However, this can be an effect of a greater focus on developing online methods, which is a trend in the MOT deep learning research community.

Another interesting thing to notice is that since the MOTA score is basically a normalized sum of FPs, FNs and ID switches, and since the number of FNs is usually at least an order of magnitude higher than the FPs and two order of magnitudes higher than the ID switches, the methods that manage to strongly reduce the number of FNs are the ones that obtain the best performances. We can in fact observe a strong correlation between MOTA and number of FNs, in accordance to what was found in [14]: MOTA and FN values are linked by a Pearson correlation coefficient of -0.95 on MOT15, -0.98 on MOT16 and -0.95 on MOT17. So, while there have been limited improvements in the reduction of FNs using public detections, the most effective way is still building and training a custom detector; the halving of the

---

[18]If not differently specified, when we use in this section expressions such as "best performing" or similar, we are always referring to a higher MOTA score, since it's the main evaluation metric used in the MOTChallenge benchmark.

	Year	Mode	MOTA↑	MOTP↑	IDF1↑	MT↑	ML↓	FP↓	FN↓	IDS↓	Frag↓	Hz↑
[83]	2015		32.4	71.8	45.3	16.0	43.8	9064	32 060	435	826	0.7
[133]	2017		19.0	71.0	17.1	5.5	45.6	11 578	36 706	1490	2081	165.2
[128]	2017		31.6	71.8		10.1	46.3			491	994	
[94]	2017		32.8	70.7	38.8	9.7	42.2	4983	35 690	614	1583	2.3
[124]	2017		34.3	70.5	48.3	11.4	43.4	5154	34 848	**348**	1463	0.5
[89]	2017		35.0	72.6	47.7	11.4	42.2	8455	31 140	358	1267	4.6
[123]	2017		37.6	71.7	46.0	15.8	26.8	7933	**29 397**	1026	2024	1.0
[145]	2017	Online	38.5	72.6	47.1	8.7	37.4	**4005**	33 204	586	1263	6.7
[117]	2018		33.6	70.9	39.1	10.4	37.6	5917	34 002	866	1566	0.1
[96]	2018		35.1	70.9	45.4	13.0	42.3	6771	32 717	381	1523	5.4
[150]	2018		37.1	71.0		14.0	31.3	7036	30 440			
[152]	2018		**42.3**		47.7	13.6	39.7					3.1
[138]	2019		22.5	70.9	25.9	6.4	61.9	7346	39 092	1159	1538	**172.8**
[158]	2019		37.1	72.5	**48.4**	12.6	39.7	8305	29 732	580	1193	1.0
[159]	2019		38.9	70.6	44.5	**16.6**	31.5	7321	29 501	720	1440	0.3
[110]	2016		29.0	71.2	34.3	8.5	48.4	5160	37 798	639	1316	52.8
[108]	2016		29.6	71.8	36.8	11.2	44.0	7786	34 733	712	943	1.7
[112]	2017	Batch	33.8	73.4	40.4	12.9	36.9	7898	32 061	703	1430	3.7
[113]	2018		22.2	71.1	27.2	3.1	61.6	5591	41 531	700	1240	8.9
[104]	2019		28.1	**74.3**	38.7			6733	36 952	477	**790**	16.9

Table 2: Experimental results of MOT algorithms using deep learning and public detections on MOT15 dataset.

	Year	Mode	MOTA↑	MOTP↑	IDF1↑	MT↑	ML↓	FP↓	FN↓	IDS↓	Frag↓	Hz↑
[35]	2016		33.4	72.1	40.4	11.7	30.9	7318	32 615	1001	1764	**260.0**
[76]	2017		32.1	70.9		13.2	30.1	6551	33 473	1687	2471	
[94]	2017	Online	51.3	74.2	54.1	36.3	22.2	7110	22 271	544	**1335**	1.3
[145]	2017		53.0	**75.5**	52.2	29.1	20.2	**5159**	22 984	708	1476	6.7
[61]	2018		32.7			38.9	26.2	19.6				11.1
[96]	2018		**56.5**	73.0	**61.3**	**45.1**	**14.6**	9386	**16 921**	**428**	1364	5.1

Table 3: Experimental results of MOT algorithms using deep learning and private detections on MOT15 dataset.

number of FNs is in fact the main reason why private detectors have lead to better tracking performances, being able to identify previously uncovered targets.

Best approaches in the four MOT steps

Speaking of private detections, the tables show that the best performing detectors are currently Faster R-CNN and its variants. In fact, the algorithm presented in [38], that uses a modified Faster R-CNN, has held its top ranking position among online methods on MOT16 for 3 years, and many of the other top-performing MOT16 algorithms have employed the same detections.

Regarding feature extraction, all the top performing methods on the three considered datasets have used a CNN to extract appearance features, where GoogLeNet is the most common one. Methods that do not exploit appearance (either extracted with deep or classical methods) tend to perform worse. However, visual features are not enough: many of the best algorithms also employ other types of features to compute affinity, especially motion ones. In fact, algorithms like LSTMs and Kalman Filters are often employed to predict the position of the target in the next frame and this often helps in improving the quality of the association. Various Bayesian filter algorithms, such as particle filter and hypotheses density filter, are also used to predict target motion, and they benefit from the use of deep models [158, 145, 98].

While deep learning plays an important role in detection and feature extraction, the use of deep networks to learn affinity functions is less ubiquitous and has not yet been proven to be essential for a good MOT algorithm, although some works have already taken advantage of it. Very few works have instead explored the use deep learning models to guide the association process, and this could be a interesting research direction for future approaches.

	Year	Mode	MOTA ↑	MOTP ↑	IDF1 ↑	MT ↑	ML ↓	FP ↓	FN ↓	IDS ↓	Frag ↓	Hz ↑
[106]	2016		35.3	75.2		7.4	51.1	5592	110 778	1598	5153	7.9
[147]	2016		38.8	75.1		7.9	49.1	8114	102 452	965	1657	11.8
[94]	2017		43.9	74.7	45.1	10.7	44.4	6450	95 175	676	1795	0.5
[124]	2017		46.0	74.9	50.0	14.6	43.6	6895	91 117	473	1422	0.2
[123]	2017		47.2	75.8	46.3	14.0	41.6	**2681**	92 856	774	1675	1.0
[157]	2018		44.2	78.3		15.2	45.7	7912	93 215	560	1212	
[117]	2018	Online	44.8	75.6	39.7	14.1	42.3	5613	94 125	968	1378	0.1
[96]	2018		45.9	74.8	48.8	13.2	41.9	6871	91 173	648	1992	0.9
[115]	2018		46.1	73.8	**54.8**	17.4	42.7	7909	89 874	532	1616	0.3
[150]	2018		47.3	74.6		17.4	39.9	6375	88 543			
[118]	2018		47.6	74.8	50.9	15.2	38.3	9253	85 431	792	1858	**20.6**
[158]	2019		48.3	76.7	50.9	15.4	40.1	2706	91 047	543	896	0.5
[159]	2019		48.8	75.7	47.2	15.8	38.1	5875	86 567	906	1116	0.1
[112]	2017		44.1	76.4	38.3	14.6	44.9	6388	94 775	745	1096	1.8
[88]	2017		45.3	75.9	47.9	17.0	39.9	11 122	87 890	639	946	1.8
[51]	2017		48.8	**79.0**		18.2	40.1	6654	86 245	481	595	0.5
[93]	2018		42.1		47.8	14.9	44.4	11 637	93 172	753	1156	1.8
[132]	2018	Batch	46.9	76.4	46.8	16.1	43.2	6257	91 669	549	757	
[103]	2018		47.2	75.7	52.4	18.6	42.8	12 586	83 107	542	787	0.5
[100]	2018		47.5		43.6	**19.4**	**36.9**	13 002	**81 762**	1035	1408	0.8
[155]	2018		47.8	75.5	44.3	19.1	38.2	8886	85 487	852	1534	0.6
[116]	2018		48.2	77.5	48.6	12.9	41.1	5104	88 586	821	1117	2.8
[148]	2018		**49.3**	**79.0**	50.7	17.8	39.9	5333	86 795	**391**	**535**	0.8

Table 4: Experimental results of MOT algorithms using deep learning and public detections on MOT16 dataset.

	Year	Mode	MOTA ↑	MOTP ↑	IDF1 ↑	MT ↑	ML ↓	FP ↓	FN ↓	IDS ↓	Frag ↓	Hz ↑
[38]	2016		66.1	79.5	65.1	34.0	20.8	5061	55 914	805	3093	9.9
[41]	2017		61.4	79.1	62.2	32.8	**18.2**	12 852	56 668	781	2008	17.4
[60]	2018		39.1			11.1	41.1	9411	99 727	1906		4.5
[44]	2018	Online	55.0	76.7		20.4	24.5	15 766	65 297	1024	1594	16.9
[43]	2018		62.6	78.3		32.7	21.1	10 604	56 182	1389	1534	
[96]	2018		63.0	78.8	63.8	39.9	22.1	13 663	53 248	482	1251	1.6
[55]	2018		64.8	78.6	**73.5**	40.6	22.0	13 470	49 927	794	1050	**39.4**
[42]	2019		65.2	78.4	62.2	32.4	21.3	6578	55 896	946	2283	11.2
[153]	2016		62.4	78.3	51.6	31.5	24.2	9855	57 257	1394	1318	34.9
[38]	2016	Batch	68.2	79.4	60.0	41.0	19.0	11 479	45 605	933	1093	0.7
[51]	2017		**71.0**	**80.2**	70.1	**46.9**	21.9	7880	**44 564**	**434**	**587**	0.5
[132]	2018		58.1	77.2	47.4	23.1	33.3	**4883**	70 207	1624	2539	

Table 5: Experimental results of MOT algorithms using deep learning and private detections on MOT16 dataset.

Other trends in top-performing algorithms

Some other trends can be identified among current top ranked methods. For example, a successful approach in online methods is the use of Single Object Tracking algorithms, properly modified in order to solve the MOT task. Some of the top performing online algorithms on the 3 datasets have in fact employed a SOT tracker augmented with deep learning techniques to recover from occlusions or to refresh the target models [159, 115, 123]. Interestingly, to the best of our knowledge, no adapted SOT algorithm has been used to perform tracking with private detections. As we have already observed, the use of private detections reduces the number of completely uncovered targets; since SOT trackers don't usually need detections to keep following a target once it's been identified, the reduction in uncovered targets might translate in a much lower number of lost tracks, that in turn would enhance the overall performance of tracker. The application of a SOT tracker on private detections could thus be a good research direction to try to further improve

	Year	Mode	MOTA ↑	MOTP ↑	IDF1 ↑	MT ↑	ML ↓	FP ↓	FN ↓	IDS ↓	Frag ↓	Hz ↑
[157]	2018		44.9	**78.9**		13.8	44.2	**22 085**	287 267	1537	3295	
[98]	2018		46.5	77.2		16.9	37.2	23 859	272 430	5649	9298	1.6
[115]	2018	Online	48.2	75.7	55.7	19.3	38.3	26 218	263 608	2194	5378	0.3
[118]	2018		50.9	76.6	52.7	17.5	35.7	24 069	250 768	2474	5317	**18.3**
[119]	2019		44.9	76.6	48.4	16.5	35.8	33 757	269 952	7136	14 491	10.1
[113]	2018		44.2	76.4	**57.2**	16.1	44.3	29 473	283 611	**1529**	**2644**	4.8
[93]	2018		47.5		51.9	18.2	41.7	25 981	268 042	2069	3124	1.9
[136]	2018	Batch	50.3		47.9	21.8	36.2	22 204	249 342	3243	3155	1.9
[155]	2018		51.3	77.0	47.6	21.4	**35.2**	24 101	247 921	2648	4279	0.2
[103]	2018		**51.8**	77.0	54.7	**23.4**	37.9	33 212	**236 772**	1834	2739	0.7

Table 6: Experimental results of MOT algorithms using deep learning and public detections on MOT17 dataset.

the results on the MOTChallenge datasets. A batch method could also exploit a SOT tracker to look at past frames in order to recover missed detections before the target was first identified by the detector.

While many methods perform association by formulating the task as a graph optimization problem, batch methods benefit in particular from this, since they can perform global optimization on them. For example, the minimum cost lifted multicut problem has reached top performance on MOT16, helped by CNN-computed affinities [148, 51], while heterogeneous association graph fusion and correlation clustering are used on the two top MOT17 methods [103, 155].

Finally, it can be noticed that the accuracy of bounding boxes radically affects the final performance of the algorithms. In fact, the top ranked tracker on MOT15 [89] obtained a relatively high MOTA score by just performing a bounding box regression step on the output of a previous state-of-the-art algorithm [89] using a deep RL agent. Developing an effective bounding box regressor to be incorporated in future MOT algorithms could be an interesting research direction that has not yet been explored thoroughly. Moreover, instead of relying on a single frame to fix the boxes, that could make them enclose the wrong target in case of an occlusion, batch methods could also try to exploit future and past target appearance to more accurately regress the bounding boxes around the right target.

5 Conclusion and future directions

We have presented a comprehensive description of all MOT algorithms employing deep learning techniques, focusing on single-camera videos and 2D data. Four main steps have been shown to characterize a generic MOT pipeline: detection, feature extraction, affinity computation, association. The use of deep learning in each of these four steps has been explored. While most of the approaches have focused on the first two, some applications of deep learning to learn affinity functions are also present, but only very few approaches use deep learning to directly guide the association algorithm. A numerical comparison of the results on the MOTChallenge datasets has also been provided, showing that, despite the wide variety of approaches, some commonalities can be found among the presented methods:

- **detection quality is important**: the amount of false negatives still dominates the MOTA score. While deep learning has allowed for some improvement in this regard for algorithms employing public detections, the use of higher quality detections is still the most effective way to reduce false negatives. Thus, a careful use of deep learning in the detection step can considerably improve the performance of a tracking algorithm;

- **CNNs are essential in feature extraction**: the use of appearance features is also fundamental for a good tracker and CNNs are particularly effective at extracting them. Moreover, strong trackers tend to use them in conjunction with motion features, that can be computed using LSTMs, Kalman filter or other Bayesian filters;

- **SOT trackers and global graph optimization work**: the adaptation of SOT trackers to the MOT task, with the help of deep learning, has recently produced good-performing online trackers; batch methods have instead benefited from the integration of deep models in global graph optimization algorithms.

As deep learning has been introduced only recently in the field of MOT, a number of promising future research directions have also been identified:

- **applying DL to track different targets**: most of DL-based MOT algorithms have focused on pedestrian tracking. Since different types of targets pose different challenges, possible improvements in tracking vehicles, animals, or other objects with the use of deep networks should be investigated;

- **applying DL to guide association**: the use of deep learning to guide the association algorithm and to directly perform tracking is still in its infancy: more research is needed in this direction to understand if deep algorithms can be useful in this step too;

- **combining SOT trackers with private detections**: a possible way to reduce the number of lost tracks, and thus reduce the false negatives, could be the combination of SOT trackers with private detections, especially in a batch setting, where it would be possible to recover past detections that were previously missed;

- **investigating bounding box regression**: the use of bounding box regression has been shown to be a promising step in obtaining a higher MOTA score, but this has not yet been explored in detail and further improvements should be investigated, e.g. the use of past and future information to guide the regression.

Finally, as very few of the presented algorithms have provided public access to their source code, we would like to encourage future researchers to publish their code in order to allow for better reproducibility of their results and benefit the whole research community.

References

[1] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[2] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[5] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[6] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[7] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth annual conference of the international speech communication association*, 2014.

[8] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.

[9] Yuchen Fan, Yao Qian, Feng-Long Xie, and Frank K Soong. Tts synthesis with bidirectional lstm based recurrent neural networks. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[10] Erik Marchi, Giacomo Ferroni, Florian Eyben, Leonardo Gabrielli, Stefano Squartini, and Björn Schuller. Multi-resolution linear prediction based features for audio onset detection with bidirectional lstm neural networks. In *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 2164–2168. IEEE, 2014.

[11] Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, Xiaowei Zhao, and Tae-Kyun Kim. Multiple object tracking: A literature review. *arXiv preprint arXiv:1409.7618*, 2014.

[12] Massimo Camplani, Adeline Paiement, Majid Mirmehdi, Dima Damen, Sion Hannuna, Tilo Burghardt, and Lili Tao. Multiple human tracking in rgb-depth data: a survey. *IET computer vision*, 11(4):265–285, 2016.

[13] Patrick Emami, Panos M Pardalos, Lily Elefteriadou, and Sanjay Ranka. Machine learning methods for solving assignment problems in multi-target tracking. *arXiv preprint arXiv:1802.06897*, 2018.

[14] Laura Leal-Taixé, Anton Milan, Konrad Schindler, Daniel Cremers, Ian Reid, and Stefan Roth. Tracking the trackers: An analysis of the state of the art in multiple object tracking. *arXiv preprint arXiv:1704.02781*, 2017.

[15] 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.

[16] 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.

[17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[18] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.

[19] Bo Wu and Ram Nevatia. Tracking of multiple, partially occluded humans based on static body part detection. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 951–958. IEEE, 2006.

[20] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *Journal on Image and Video Processing*, 2008:1, 2008.

[21] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision*, pages 17–35. Springer, 2016.

[22] Rainer Stiefelhagen and John Garofolo. *Multimodal Technologies for Perception of Humans: First International Evaluation Workshop on Classification of Events, Activities and Relationships, CLEAR 2006, Southampton, UK, April 6-7, 2006, Revised Selected Papers*, volume 4122. Springer, 2007.

[23] Rainer Stiefelhagen, Rachel Bowers, and Jonathan Fiscus. *Multimodal Technologies for Perception of Humans: International Evaluation Workshops CLEAR 2007 and RT 2007, Baltimore, MD, USA, May 8-11, 2007, Revised Selected Papers*, volume 4625. Springer, 2008.

[24] Piotr Dollár, Ron Appel, Serge Belongie, and Pietro Perona. Fast feature pyramids for object detection. *IEEE transactions on pattern analysis and machine intelligence*, 36(8):1532–1545, 2014.

[25] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2009.

[26] Ross B. Girshick, Pedro F. Felzenszwalb, and David McAllester. Discriminatively trained deformable part models, release 5. http://people.cs.uchicago.edu/~rbg/latent-release5/, 2012.

[27] Fan Yang, Wongun Choi, and Yuanqing Lin. Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2129–2137, 2016.

[28] Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, Javen Shi, Daniel Cremers, Ian Reid, Stefan Roth, Konrad Schindler, and Laura Leal-Taixe. Cvpr19 tracking and detection challenge: How crowded can it get?, 2019.

[29] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

[30] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[31] Xiaoyu Wang, Ming Yang, Shenghuo Zhu, and Yuanqing Lin. Regionlets for generic object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 17–24, 2013.

[32] Longyin Wen, Dawei Du, Zhaowei Cai, Zhen Lei, Ming-Ching Chang, Honggang Qi, Jongwoo Lim, Ming-Hsuan Yang, and Siwei Lyu. Ua-detrac: A new benchmark and protocol for multi-object detection and tracking. *arXiv preprint arXiv:1511.04136*, 2015.

[33] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. Monocular 3d pose estimation and tracking by detection. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 623–630. IEEE, 2010.

[34] James Ferryman and Ali Shahrokni. Pets2009: Dataset and challenge. In *2009 Twelfth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pages 1–6. IEEE, 2009.

[35] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[36] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[37] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[38] Fengwei Yu, Wenbo Li, Quanquan Li, Yu Liu, Xiaohua Shi, and Junjie Yan. Poi: Multiple object tracking with high performance detection and appearance feature. In *European Conference on Computer Vision*, pages 36–42. Springer, 2016.

[39] Sean Bell, C Lawrence Zitnick, Kavita Bala, and Ross Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2874–2883, 2016.

[40] Spyros Gidaris and Nikos Komodakis. Object detection via a multi-region and semantic segmentation-aware cnn model. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1134–1142, 2015.

[41] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[42] Nima Mahmoudi, Seyed Mohammad Ahadi, and Mohammad Rahmati. Multi-target tracking using cnn-based features: Cnnmtt. *Multimedia Tools and Applications*, 78(6):7077–7096, 2019.

[43] Xingyu Wan, Jinjun Wang, and Sanping Zhou. An online and flexible multi-object tracking framework using long short-term memory. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1230–1238, 2018.

[44] Takayuki Ujiie, Masayuki Hiromoto, and Takashi Sato. Interpolation-based object detection using motion vectors for embedded real-time tracking systems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 616–624, 2018.

[45] Qizheng He, Jianan Wu, Gang Yu, and Chi Zhang. Sot for mot. *arXiv preprint arXiv:1712.01059*, 2017.

[46] Minghua Li, Zhengxi Liu, Yunyu Xiong, and Zheng Li. Multi-person tracking by discriminative affinity model and hierarchical association. In *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, pages 1741–1745. IEEE, 2017.

[47] Wenbo Li, Ming-Ching Chang, and Siwei Lyu. Who did what at where and when: Simultaneous multi-person tracking and activity recognition. *arXiv preprint arXiv:1807.01253*, 2018.

[48] Felipe Jorquera, Sergio Hernández, and Diego Vergara. Probability hypothesis density filter using determinantal point processes for multi object tracking. *Computer Vision and Image Understanding*, 2019.

[49] Zhao Zhong, Zichen Yang, Weitao Feng, Wei Wu, Yangyang Hu, and Cheng-lin Liu. Decision controller for object tracking with deep reinforcement learning. *IEEE Access*, 2019.

[50] Weigang Lu, Zhiping Zhou, Lijuan Zhang, and Guoqiang Zheng. Multi-target tracking by non-linear motion patterns based on hierarchical network flows. *Multimedia Systems*, pages 1–12, 2019.

[51] Siyu Tang, Mykhaylo Andriluka, Bjoern Andres, and Bernt Schiele. Multiple people tracking by lifted multicut and person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3539–3548, 2017.

[52] Nan Ran, Longteng Kong, Yunhong Wang, and Qingjie Liu. A robust multi-athlete tracking algorithm by exploiting discriminant features and long-term dependencies. In *International Conference on Multimedia Modeling*, pages 411–423. Springer, 2019.

[53] Haigen Hu, Lili Zhou, Qiu Guan, Qianwei Zhou, and Shengyong Chen. An automatic tracking method for multiple cells based on multi-feature fusion. *IEEE Access*, 6:69782–69793, 2018.

[54] Lei Zhang, Helen Gray, Xujiong Ye, Lisa Collins, and Nigel Allinson. Automatic individual pig detection and tracking in pig farms. *Sensors*, 19(5):1188, 2019.

[55] Zongwei Zhou, Junliang Xing, Mengdan Zhang, and Weiming Hu. Online multi-target tracking with tensor-based high-order graph matching. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 1809–1814. IEEE, 2018.

[56] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Eco: efficient convolution operators for tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6638–6646, 2017.

[57] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *international Conference on computer vision & Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE Computer Society, 2005.

[58] Joost Van De Weijer, Cordelia Schmid, Jakob Verbeek, and Diane Larlus. Learning color names for real-world applications. *IEEE Transactions on Image Processing*, 18(7):1512–1523, 2009.

[59] Yongyi Lu, Cewu Lu, and Chi-Keung Tang. Online video object detection using association lstm. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2344–2352, 2017.

[60] Hilke Kieritz, Wolfgang Hubner, and Michael Arens. Joint detection and online multi-object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1459–1467, 2018.

[61] Dawei Zhao, Hao Fu, Liang Xiao, Tao Wu, and Bin Dai. Multi-object tracking with correlation filter for autonomous vehicle. *Sensors*, 18(7):2004, 2018.

[62] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[63] Mengmeng Wang, Yong Liu, and Zeyi Huang. Large margin object tracking with circulant feature maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4021–4029, 2017.

[64] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[65] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[66] Sang Jun Kim, Jae-Yeal Nam, and Byoung Chul Ko. Online tracker optimization for multi-pedestrian tracking using a moving vehicle camera. *IEEE Access*, 6:48675–48687, 2018.

[67] Sarthak Sharma, Junaid Ahmed Ansari, J Krishna Murthy, and K Madhava Krishna. Beyond pixels: Leveraging geometry and shape cues for online multi-object tracking. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3508–3515. IEEE, 2018.

[68] Jimmy Ren, Xiaohao Chen, Jianbo Liu, Wenxiu Sun, Jiahao Pang, Qiong Yan, Yu-Wing Tai, and Li Xu. Accurate single stage detector using recurrent rolling convolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5420–5428, 2017.

[69] Yu Xiang, Wongun Choi, Yuanqing Lin, and Silvio Savarese. Subcategory-aware convolutional neural networks for object proposals and detection. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 924–933. IEEE, 2017.

[70] Federico Pernici, Federico Bartoli, Matteo Bruni, and Alberto Del Bimbo. Memory based online learning of deep representations from video streams. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2324–2334, 2018.

[71] Peiyun Hu and Deva Ramanan. Finding tiny faces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 951–959, 2017.

[72] Weidong Min, Mengdan Fan, Xiaoguang Guo, and Qing Han. A new approach to track multiple vehicles with the combination of robust detection and two classifiers. *IEEE Transactions on Intelligent Transportation Systems*, 19(1):174–186, 2018.

[73] Olivier Barnich and Marc Van Droogenbroeck. Vibe: A universal background subtraction algorithm for video sequences. *IEEE Transactions on Image processing*, 20(6):1709–1724, 2011.

[74] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[75] Shaoyong Yu, Yun Wu, Wei Li, Zhijun Song, and Wenhua Zeng. A model for fine-grained vehicle classification based on deep learning. *Neurocomputing*, 257:97–103, 2017.

[76] Sebastian Bullinger, Christoph Bodensteiner, and Michael Arens. Instance flow based online multiple object tracking. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 785–789. IEEE, 2017.

[77] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158, 2016.

[78] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on Image analysis*, pages 363–370. Springer, 2003.

[79] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Deepmatching: Hierarchical deformable dense matching. *International Journal of Computer Vision*, 120(3):300–323, 2016.

[80] Yinlin Hu, Rui Song, and Yunsong Li. Efficient coarse-to-fine patchmatch for large displacement optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5704–5712, 2016.

[81] Li Wang, Nam Trung Pham, Tian-Tsong Ng, Gang Wang, Kap Luk Chan, and Karianto Leman. Learning deep features for multiple object tracking by using a multi-task learning strategy. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 838–842. IEEE, 2014.

[82] Charles Cadieu and Bruno A Olshausen. Learning transformational invariants from natural movies. In *Advances in neural information processing systems*, pages 209–216, 2009.

[83] Chanho Kim, Fuxin Li, Arridhana Ciptadi, and James M Rehg. Multiple hypothesis tracking revisited. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4696–4704, 2015.

[84] Liang Zheng, Hengheng Zhang, Shaoyan Sun, Manmohan Chandraker, Yi Yang, and Qi Tian. Person re-identification in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1367–1376, 2017.

[85] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1116–1124, 2015.

[86] Douglas Gray and Hai Tao. Viewpoint invariant pedestrian recognition with an ensemble of localized features. In *European conference on computer vision*, pages 262–275. Springer, 2008.

[87] Wei Li, Rui Zhao, Tong Xiao, and Xiaogang Wang. Deepreid: Deep filter pairing neural network for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 152–159, 2014.

[88] Jiahui Chen, Hao Sheng, Yang Zhang, and Zhang Xiong. Enhancing detection model for multiple hypothesis tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 18–27, 2017.

[89] Min Yang, Yuwei Wu, and Yunde Jia. A hybrid data association framework for robust online multi-object tracking. *IEEE Transactions on Image Processing*, 26(12):5667–5679, 2017.

[90] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158, 2015.

[91] Shuo Hong Wang, Jing Wen Zhao, and Yan Qiu Chen. Robust tracking of fish schools using cnn for head identification. *Multimedia Tools and Applications*, 76(22):23679–23697, 2017.

[92] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[93] Chanho Kim, Fuxin Li, and James M Rehg. Multi-object tracking with neural gating using bilinear lstm. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 200–215, 2018.

[94] Seung-Hwan Bae and Kuk-Jin Yoon. Confidence-based data association and discriminative deep appearance learning for robust online multi-object tracking. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):595–610, 2017.

[95] Mohib Ullah and Faouzi Alaya Cheikh. Deep feature based end-to-end transportation network for multi-target tracking. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 3738–3742. IEEE, 2018.

[96] Kuan Fang, Yu Xiang, Xiaocheng Li, and Silvio Savarese. Recurrent autoregressive networks for online multi-object tracking. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 466–475. IEEE, 2018.

[97] Tong Xiao, Hongsheng Li, Wanli Ouyang, and Xiaogang Wang. Learning deep feature representations with domain guided dropout for person re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1249–1258, 2016.

[98] Zeyu Fu, Federico Angelini, Syed Mohsen Naqvi, and Jonathon A Chambers. Gm-phd filter based online multiple human tracking using deep discriminative correlation matching. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4299–4303. IEEE, 2018.

[99] B-N Vo and W-K Ma. The gaussian mixture probability hypothesis density filter. *IEEE Transactions on signal processing*, 54(11):4091–4104, 2006.

[100] Longyin Wen, Dawei Du, Shengkun Li, Xiao Bian, and Siwei Lyu. Learning non-uniform hypergraph for multi-object tracking. *Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.

[101] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[102] Flip Korn and Suresh Muthukrishnan. Influence sets based on reverse nearest neighbor queries. *ACM Sigmod Record*, 29(2):201–212, 2000.

[103] Hao Sheng, Yang Zhang, Jiahui Chen, Zhang Xiong, and Jun Zhang. Heterogeneous association graph fusion for target association in multiple object tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.

[104] Longtao Chen, Xiaojiang Peng, and Mingwu Ren. Recurrent metric networks and batch multiple hypothesis for multi-object tracking. *IEEE Access*, 7:3093–3105, 2019.

[105] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter V Gehler, and Bernt Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4929–4937, 2016.

[106] Minyoung Kim, Stefano Alletto, and Luca Rigazio. Similarity mapping with enhanced siamese network for multi-object tracking. In *Machine Learning for Intelligent Transportation Systems (MLITS), 2016 NIPS Workshop*, 2016.

[107] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744, 1994.

[108] Bing Wang, Li Wang, Bing Shuai, Zhen Zuo, Ting Liu, Kap Luk Chan, and Gang Wang. Joint learning of convolutional neural networks and temporally constrained metrics for tracklet association. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8, 2016.

[109] Shun Zhang, Yihong Gong, Jia-Bin Huang, Jongwoo Lim, Jinjun Wang, Narendra Ahuja, and Ming-Hsuan Yang. Tracking persons-of-interest via adaptive discriminative features. In *European conference on computer vision*, pages 415–433. Springer, 2016.

[110] Laura Leal-Taixé, Cristian Canton-Ferrer, and Konrad Schindler. Learning by tracking: Siamese cnn for robust target association. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 33–40, 2016.

[111] Laura Leal-Taixé, Gerard Pons-Moll, and Bodo Rosenhahn. Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker. In *2011 IEEE international conference on computer vision workshops (ICCV workshops)*, pages 120–127. IEEE, 2011.

[112] Jeany Son, Mooyeol Baek, Minsu Cho, and Bohyung Han. Multi-object tracking with quadruplet convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5620–5629, 2017.

[113] Andrii Maksai and Pascal Fua. Eliminating exposure bias and loss-evaluation mismatch in multiple object tracking. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[114] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.

[115] Ji Zhu, Hua Yang, Nian Liu, Minyoung Kim, Wenjun Zhang, and Ming-Hsuan Yang. Online multi-object tracking with dual matching attention networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 366–382, 2018.

[116] 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. In *2018 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2018.

[117] Hui Zhou, Wanli Ouyang, Jian Cheng, Xiaogang Wang, and Hongsheng Li. Deep continuous conditional random fields with asymmetric inter-object constraints for online multi-object tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.

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

[119] Sangyun Lee and Euntai Kim. Multiple object tracking via feature pyramid siamese networks. *IEEE Access*, 7:8181–8194, 2019.

[120] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[121] Mohib Ullah, Ahmed Kedir Mohammed, Faouzi Alaya Cheikh, and Zhaohui Wang. A hierarchical feature model for multi-target tracking. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 2612–2616. IEEE, 2017.

[122] Stéphane G Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on signal processing*, 41(12):3397–3415, 1993.

[123] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 300–311, 2017.

[124] 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. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4836–4845, 2017.

[125] Mustafa Ozuysal, Michael Calonder, Vincent Lepetit, and Pascal Fua. Fast keypoint recognition using random ferns. *IEEE transactions on pattern analysis and machine intelligence*, 32(3):448–461, 2009.

[126] Mohib Ullah and Faouzi Alaya Cheikh. A directed sparse graphical model for multi-target tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1816–1823, 2018.

[127] Lawrence R Rabiner and Biing-Hwang Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986.

[128] Lu Wang, Lisheng Xu, Min Young Kim, Luca Rigazico, and Ming-Hsuan Yang. Online multiple object tracking via flow and convolutional features. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3630–3634. IEEE, 2017.

[129] Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. Hierarchical convolutional features for visual tracking. In *Proceedings of the IEEE international conference on computer vision*, pages 3074–3082, 2015.

[130] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of Imaging Understanding Workshop*, pages 121–130. Vancouver, British Columbia, 1981.

[131] Pol Rosello and Mykel J Kochenderfer. Multi-agent reinforcement learning for multi-object tracking. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1397–1404. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[132] Maryam Babaee, Zimu Li, and Gerhard Rigoll. Occlusion handling in tracking multiple people using rnn. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 2715–2719. IEEE, 2018.

[133] Anton Milan, S Hamid Rezatofighi, Anthony Dick, Ian Reid, and Konrad Schindler. Online multi-target tracking using recurrent neural networks. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[134] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to track: Online multi-object tracking by decision making. In *Proceedings of the IEEE international conference on computer vision*, pages 4705–4713, 2015.

[135] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. Rmpe: Regional multi-person pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2334–2343, 2017.

[136] Yiming Liang and Yue Zhou. Lstm multiple object tracker combining multiple cues. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 2351–2355. IEEE, 2018.

[137] Sanping Zhou, Jinjun Wang, Deyu Meng, Xiaomeng Xin, Yubing Li, Yihong Gong, and Nanning Zheng. Deep self-paced learning for person re-identification. *Pattern Recognition*, 76:739–751, 2018.

[138] Kwangjin Yoon, Du Yong Kim, Young-Chul Yoon, and Moongu Jeon. Data association for multi-object tracking via deep neural networks. *Sensors*, 19(3):559, 2019.

[139] Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *European conference on computer vision*, pages 549–565. Springer, 2016.

[140] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. People-tracking-by-detection and people-detection-by-tracking. In *2008 IEEE Conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008.

[141] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[142] Bjoern Andres, Andrea Fuksová, and Jan-Hendrik Lange. Lifting of multicuts. *CoRR, abs/1503.03791*, 3, 2015.

[143] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1385–1392, 2013.

[144] Margret Keuper, Evgeny Levinkov, Nicolas Bonneel, Guillaume Lavoué, Thomas Brox, and Bjorn Andres. Efficient decomposition of image and mesh graphs by lifted multicuts. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1751–1759, 2015.

[145] Long Chen, Haizhou Ai, Chong Shang, Zijie Zhuang, and Bo Bai. Online multi-object tracking with convolutional neural networks. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 645–649. IEEE, 2017.

[146] M Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on signal processing*, 50(2):174–188, 2002.

[147] Ricardo Sanchez-Matilla, Fabio Poiesi, and Andrea Cavallaro. Online multi-target tracking with strong and weak detections. In *European Conference on Computer Vision*, pages 84–99. Springer, 2016.

[148] Liqian Ma, Siyu Tang, Michael J. Black, and Luc Van Gool. Customized multi-person tracker. In *Computer Vision – ACCV 2018*. Springer International Publishing, December 2018.

[149] Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. Multi-person tracking by multicut and deep matching. In *European Conference on Computer Vision*, pages 100–111. Springer, 2016.

[150] Liangliang Ren, Jiwen Lu, Zifeng Wang, Qi Tian, and Jie Zhou. Collaborative deep reinforcement learning for multi-object tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 586–602, 2018.

[151] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4293–4302, 2016.

[152] Yifan Jiang, Hyunhak Shin, and Hanseok Ko. Precise regression for bounding box correction for improved tracking based on deep reinforcement learning. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1643–1647. IEEE, 2018.

[153] Byungjae Lee, Enkhbayar Erdenee, Songguo Jin, Mi Young Nam, Young Giu Jung, and Phill Kyu Rhee. Multi-class multi-object tracking using changing point detection. In *European Conference on Computer Vision*, pages 68–83. Springer, 2016.

[154] Anthony Hoak, Henry Medeiros, and Richard Povinelli. Image-based multi-target tracking through multi-bernoulli filtering with interactive likelihoods. *Sensors*, 17(3):501, 2017.

[155] Roberto Henschel, Laura Leal-Taixé, Daniel Cremers, and Bodo Rosenhahn. Fusion of head and full-body detectors for multi-object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1428–1437, 2018.

[156] Russell Stewart, Mykhaylo Andriluka, and Andrew Y Ng. End-to-end people detection in crowded scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2325–2333, 2016.

[157] Weihao Gan, Shuo Wang, Xuejing Lei, Ming-Sui Lee, and C-C Jay Kuo. Online cnn-based multiple object tracking with enhanced model updates and identity association. *Signal Processing: Image Communication*, 66:95–102, 2018.

[158] Jun Xiang, Guoshuai Zhang, and Jianhua Hou. Online multi-object tracking based on feature representation and bayesian filtering within a deep learning architecture. *IEEE Access*, 2019.

[159] Peng Chu, Heng Fan, Chiu C Tan, and Haibin Ling. Online multi-object tracking with instance-aware tracker and dynamic model refreshment. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 161–170. IEEE, 2019.

[160] Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

[161] Jun-ichi Takeuchi and Kenji Yamanishi. A unifying framework for detecting outliers and change points from time series. *IEEE transactions on Knowledge and Data Engineering*, 18(4):482–492, 2006.

[162] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: A benchmark. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 304–311. IEEE, 2009.

[163] Reza Hoseinnezhad, Ba-Ngu Vo, Ba-Tuong Vo, and David Suter. Visual tracking of numerous targets via multi-bernoulli filtering of image data. *Pattern Recognition*, 45(10):3625–3635, 2012.

[164] Anton Milan, Rikke Gade, Anthony Dick, Thomas B Moeslund, and Ian Reid. Improving global multi-target tracking with local updates. In *European Conference on Computer Vision*, pages 174–190. Springer, 2014.

[165] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

[166] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7291–7299, 2017.

[167] Liming Zhao, Xi Li, Yueting Zhuang, and Jingdong Wang. Deeply-learned part-aligned representations for person re-identification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3219–3228, 2017.

[168] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence*, 37(3):583–596, 2014.

[169] Longyin Wen, Wenbo Li, Junjie Yan, Zhen Lei, Dong Yi, and Stan Z Li. Multiple target tracking based on undirected hierarchical relation hypergraph. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1282–1289, 2014.

[170] Zhi-Ming Qian, Xi En Cheng, and Yan Qiu Chen. Automatically detect and track multiple fish swimming in shallow water with frequent occlusion. *PloS one*, 9(9):e106506, 2014.

[171] Hamed Pirsiavash, Deva Ramanan, and Charless C Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *CVPR 2011*, pages 1201–1208. IEEE, 2011.

[172] Steven Gold, Anand Rangarajan, et al. Softmax to softassign: Neural network algorithms for combinatorial optimization. *Journal of Artificial Neural Networks*, 2(4):381–399, 1996.

[173] Chang Huang, Bo Wu, and Ramakant Nevatia. Robust object tracking by hierarchical association of detection responses. In *European Conference on Computer Vision*, pages 788–801. Springer, 2008.

[174] Rodrigo Benenson, Markus Mathias, Radu Timofte, and Luc Van Gool. Pedestrian detection at 100 frames per second. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2903–2910. IEEE, 2012.

A   Appendix

Here we present a table containing a summary of the techniques used by each algorithm presented in this paper. The table follows the order of presentation of the papers. Since we think that the publication of open source code can greatly help the research community, we have also provided links to the source codes for the papers that provide them.

	Detection	Feature extr. / mot. pred.	Affinity / cost computation	Association / Tracking	Mode	Source and data
[35]	Faster R-CNN	Kalman filter	IoU	Hungarian algorithm	O	Source
[38]	Modified Faster R-CNN	Modified GoogLeNet, Kalman filter	Cosine distance + IoU	Hungarian algorithm (online), modified H2T [169] (batch)	O+B	Detections and appearance features
[52]	Faster R-CNN	CNN (app.), AlphaPose CNN, pose joints velocities, interaction grid	Pose-based Triple Stream Network (LSTM-based)	Custom algorithm	O	
[53]	Faster R-CNN	CNN	Euclidean distance, cosine distance	Multifeature fusion re-tracking algorithm	B	
[54]	CNN	HOG + Colour Names	Variation of Discriminative Correlation Filter	Custom algorithm + Hungarian algorithm	O	
[59]	SSD	SSD, LSTM	Cosine similarity	Hungarian algorithm	O	
[60]	SSD	SSD	RNN	Hungarian algorithm, MLP (track scores)	O	
[61]	SSD	SSD + Correlation Filter	IoU + APCE	Hungarian algorithm	O	
[55]	Public / Mask R-CNN	Siamese Mask R-CNN	App. affinity, mot. consistency, spatial structural potential	Tensor-based high-order graph matching	O	Code will be released
[66]	YOLOv2	Tiny Yolo, Particle filter, Random Ferns, KLT	Pairwise overlap ratio, student Random Ferns, Euclidean distance	Greedy bipartite assignment	O	
[67]	RRC or SubCNN	Feature-based odometry, Pose Adjustment CNN, stacked-hourglass CNN	3D-2D cost + 3D-3D cost + appearance, shape and pose costs	Hungarian algorithm	O	Source
[70]	DPM or Tiny (CNN)	DPM or Tiny (CNN)	Implicit in Reverse Nearest Neighbour	Reverse Nearest Neighbour Matching	O	Code will be released
[72]	ViBe + SVM + CNN		IoU	Region Matching algorithm	O	
[76]	Multi-task Network Cascades (CNN)	Optical flow	Overlap of segmentation instances	Hungarian algorithm	O	
[81]	Dalal-Triggs detector	Autoencoders	SVM	Minimum spanning tree	O	

35

	Detection	Feature extr. / mot. pred.	Affinity / cost computation	Association / Tracking	Mode	Source and data
[83]	Public	CNN + PCA	Multi-Output Regularized Least Squares	Variation of Multiple Hypothesis Tracking	O	Source
[88]	Public	CNN, Kalman Filter	Multi-Output Regularized Least Squares + Kalman Filter + detection-scene score	Maximum Weighted Independent Set	B	
[89]	Public	R-CNN	Observation cost + transition cost + birth-death cost	Min-cost multi commodity flow problem, solved with Dantzig-Wolfe decomposition	O	
[91]	DoH [170]	CNN	CNN + Kalman filter	Custom algorithm, SVM	B	
[41]	From [38]	Kalman filter, Wide Residual Net	Mahalanobis dist. (mot.) + cosine distance (app.), IoU	Hungarian algorithm	O	Source
[42]	From [38]	CNN	appearance + motion + dynamic affinity	Hungarian algorithm	O	
[93]	Public	CNN	Bilinear LSTM	Variant of MHT-DAM [83]	B	
[94]	Public / SDP+RPN	CNN	Appearance + motion + shape affinities	Hungarian algorithm	O	Source
[95]	Public	GoogLeNet CNN	App. similarity	Bayesian inference using [171]	B	
[96]	Public / Faster R-CNN	GoogLeNet CNN	Recurrent Autoregressive Networks (GRU-based)	Bipartite graph matching	O	
[98]	Public	CNN	Hybrid Likelihood Function (Discriminative Correlation Filter + Gaussian Mixture Probability Hypothesis Density)	Hungarian algorithm	O	
[100]	Public	CNN	app. + HSV histogram + motion similarities	Pairwise update algorithm + SSVM	B	Will be available at this link
[103]	Public	GoogLeNet CNN, Optical flow	Distance between app. features, common superpixels, optical flow predictions	Multiple Hypotheses Tracking	B	
[104]	Public	CNN	LSTM (app.) + motion affinity	Batch Multi-Hypothesis	B	

36

	Detection	Feature extr. / mot. pred.	Affinity / cost computation	Association / Tracking	Mode	Source and data
[51]	Public / From [38]	DeepCut CNN [105], StackNetPose CNN	StackNetPose CNN	Lifted multicut problem, solved as in [144]	B	Source
[106]	Public	Siamese CNN	Euclidean distance (app. feat.) + IoU + box area ratio	Custom greedy algorithm	O	
[108]	DPM	Siamese CNN with temporal constraints	Mahalanobis distance (app. feat.) + motion affinity	Generalized Linear Assignment solved with Softassign [172], Dual-threshold strategy [173]	B	
[109]	HeadHunter [174]	CNN	Euclidean distance (app. feat.), temporal and kinematic affinities	Hungarian algorithm, Agglomerative clustering	B	Source
[110]	Public	Siamese CNN, contextual features	Gradient Boosting	Linear programming	B	
[112]	Public	CNN, sequence-specific statistics, optical flow, FC layers	FC layer combining app. and mot. distances	Minimax label propagation	B	
[113]	Public	CNN + various app. and non-app. feat.	embedding layer + bidirectional LSTM	Variation of Multiple Hypothesis Tracking	B	
[115]	Public	Linear motion model, Spatial Attention Network CNN	Temporal Attention Network (bidirectional LSTM)	Custom greedy algorithm, ECO (SOT tracker)	O	Source
[116]	Public	Siamese CNN, LSTM, WRN CNN, Siamese Bi-GRU + CNN	Euclidean dist. (app. feat.), spatial distance, GRU feature matching	Hungarian algorithm, bi-GRU RNN (track split), custom algorithm	B	
[117]	Public	DCCRF, visual-displacement CNN	Visual-similarity CNN, IoU	Hungarian algorithm	O	
[118]	Public	R-FCN + Kalman Filter, GoogleNet	Eucl. dist. (app. feat.), IoU	Hierarchical Data Association	O	Source
[119]	Public	Feature Pyramid Siamese Network, motion features	Feature Pyramid Siamese Network	Custom greedy algorithm	O	
[121]	Public	Kalman Filter, GoogLeNet	Distance between sparse coding of features using a learned dictionary	Hungarian algorithm	B	
[123]	Public	3 LSTMs (app., mot., interaction features) using CNN, bb velocity, occupancy map	LSTM	Hungarian algorithm, SOT tracker [134]	O	

	Detection	Feature extr. / mot. pred.	Affinity / cost computation	Association / Tracking	Mode	Source and data
[124]	Public	Linear motion model, CNN	CNN	Association to highest classification score	O	
[126]	Manually generated	Hidden Markov Models, CNN	Mutual information (app. feat.)	Dynamic programming algorithm from [171]	B	
[128]	Public	LK optical flow, Convolutional Correlation Filter CNN, Kalman filter	Optical flow aff., app. feat. aff., IoU, scale affinity, distance between detections	Custom algorithm (with Hungarian alg.)	O	Source
[131]	Public	Kalman filter + Deep RL agent	IoU	Hungarian algorithm + Deep RL agent	O	
[132]	N/A	LSTM (mot.)	Stitching score using IoU	Custom iterative tracklet-stitching algorithm	B	
[133]	Public	RNN (mot.)	LSTM	RNN	O	Source
[136]	Public	2 LSTMs, VGG16 CNN	SVM, Siamese LSTM	Greedy association	B	
[43]	From [38]	Kalman filter or LK optical flow, CNN + motion features	IoU, Siamese LSTM	Hungarian algorithm	B	
[138]	Public		FC layers + Bi-directional LSTM	Hungarian algorithm	O	
[145]	Public / from [147] (combines DPM, SDP and ACF)	Modified Faster R-CNN	Modified Faster R-CNN	Particle filter	O	
[148]	Public	DeepMatching, Siamese CNN	Edge potential as in [149], Siamese CNN	Lifted multicut	B	
[150]	Public	CNN (motion pred.), part of MDNet (CNN)	N/A	Deep RL agents	O	

Table 7: Information summary about the methods commented in section 3. In each column, the approach for each paper in that step is shown. *app.* means appearance, *mot.* means motion, *feat.* means features, *pred.* means prediction; *O* and *B* in the Mode column indicate Online and Batch methods respectively. Text in the last column is clickable and contains links to the specified data.