Self-Supervised Learning for Multi-Object Tracking

Favyen Bastani, Songtao He, Sam Madden

Abstract

In this paper, we develop a self-supervised learning approach for multi-object tracking (MOT) under the detect-to-track framework: we assume that an object detector trained on image-level bounding box annotations is available, but train a tracking model using only unlabeled video. A key challenge in applying self-supervision to MOT is training a tracker that incorporates an RNN to robustly associate detections across occlusion. Supervisory signals proposed in prior work on single-object tracking, such as color propagation and cycle-consistency, are not effective for training RNN trackers: they yield degenerate models that, for instance, always match a new detection to the track with the closest initial detection. Broadly, these signals depend on hiding information (e.g, color) from a model, and training the model to recover that information. We propose a novel self-supervised training approach that we call dual-tracker consistency: we construct two distinct inputs for one video sequence, where each input is a variation of the video sequence where different information has been hidden; we then apply two instances of a tracker independently on each input, and train the model to produce consistent outputs. We evaluate our method on the MOT17 benchmark, and find that, when training only on a corpus of unlabeled video, our method is competitive with two recent fully supervised state-of-the-art trackers that train on expensive video-level bounding box and track annotations.

1 Introduction

Multi-object trackers identify all instances of a particular object type in video, and track each instance through the segment of video in which it is visible in the camera frame. Annotating training data for multi-object tracking is tedious and costly; for example, annotation of pedestrian tracks in just six minutes of video in the training set of the MOT15 Challenge (Leal-Taixé et al. 2015) requires an estimated 22 hours (Manen et al. 2017) of human labeling time using LabelMe (Yuen et al. 2009). While unsupervised detect-to-track methods (Bewley et al. 2016; Bochinski, Senst, and Sikora 2018) have been proposed that group detections into tracks by estimating motion using a combination of spatial and visual cues, these methods suffer low-accuracy in scenarios with frequent occlusion.

Recent work has proposed applying self-supervised learning for single-object tracking (Vondrick et al. 2018; Wang, Jabri, and Efros 2019). These approaches train a model to propagate instance labels from a reference frame through the rest of a video sequence. In contrast to work on self-supervised representation learning from video, these approaches require no fine-tuning to apply the model for single-object tracking.

A significant limitation in prior work is that the model independently compares pairs of frames at a time. In multi-object tracking, a key challenge is robustly re-localizing tracks across potentially long occlusions, especially when an instance is occluded by other instances of the same object type. Pairwise frame comparisons are thus insufficient for high-accuracy multi-object tracking; instead, learning recurrent features that encode the history of a track is crucial for enabling robust re-localization. However, extending prior work to learn RNN parameters is challenging. For example, Wang et al. (Wang, Jabri, and Efros 2019) propose training using forward-backward consistency: from a patch in an initial frame, after tracking forwards through video and then backwards to return to the initial frame, the final patch should align with the original patch. Training an RNN in this way would be ineffective as the RNN could simply memorize the features of the original patch.

To address this challenge, we propose a novel self-supervised learning approach, *dual-tracker consistency*. At a high-level, our dual-tracker consistency approach creates a self-supervisory signal by applying two instances of a tracker model (where the instances share the same parameters) through two distinct input variations extracted from one video sequence. The tracker is then trained to produce similar tracking outputs through the video sequence under both inputs. To attain high similarity, the tracker must accurately associate detections that correspond to the same object: if the tracker were to instead arbitrarily associate detections, then variations in the inputs would lead the tracker instances to produce dissimilar outputs.

We use a detect-to-track framework, and assume that a robust detector is available—image-level bounding boxes are much cheaper to annotate than tracks, and are required even in unsupervised detect-to-track methods. Additionally, our approach assumes that a corpus of unlabeled video is available, along with detections in this video automatically in-

ferred by the detector. Our method does not train on expensive video-level bounding box or track annotations. Under this framework, the challenge is to automatically learn to determine which detections correspond to the same object across different video frames, and to identify when an object enters or leaves the camera frame.

To implement dual-tracker consistency, we adapt a now standard RNN model and tracker architecture from prior work (Kim, Li, and Rehg 2018): the tracker maintains a set of active tracks that have not yet left the camera frame, and processes each frame in sequence by matching track features computed through the RNN to detections in the current frame. In prior work, this model is trained under a fully supervised procedure: they randomly sample a video segment $\langle I_0, \ldots, I_n \rangle$ and a track t in that segment, and apply the tracker on t over the video segment. On each frame I_j, the RNN outputs a probability distribution indicating the likelihood that the prefix of t up to I_j matches with each detection in I_j. The label (i.e., the correct detection in the frame) is then back-propagated under cross entropy loss.

Since we assume no track labels, on each training iteration, we propose to instead sample a segment $\langle I_0, \ldots, I_n \rangle$, and apply the RNN model to compute a transition matrix that specifies the probability that each detection in I_0 (rows) matches with each detection in I_n (columns). Then, when applying two instances of the tracker on two input variations extracted from the segment, we obtain two transition matrices (one from each tracker instance). We compute a dot-product similarity score that compares these matrices, and back-propagate this as a loss function.

We propose two input-hiding schemes that each select two distinct input variations from a video segment: occlusion-based hiding and visual-spatial hiding. Occlusion-based hiding eliminates detections in random intermediate subsequences of frames to simulate occlusion incidents. It constructs two distinct inputs by eliminating different subsequences. Visual-spatial hiding applies the tracker once when only observing spatial inputs (bounding box coordinates), and once when only observing visual inputs (pixel values inside detection bounding boxes).

We evaluate our approach on the MOT17 benchmark against six state-of-the-art multi-object tracking methods, including both unsupervised and supervised methods. We train our tracker model using dual-tracker consistency over a corpus of unlabeled video, which can be cheaply obtained. Like other detect-to-track methods (including unsupervised approaches), we also use an object detector trained on image-level bounding box annotations in MOT17Det. We find that our approach improves both IDF1 and MOTA accuracy over the unsupervised baselines by 3% to 7%. Moreover, remarkably, our approach is competitive with two of the three supervised methods we compared, which require expensive video-level bounding box and track annotations.

2 Related Work

Self-supervised learning over video has been studied extensively in many contexts. Most work focuses on learning representations of video that can be applied through fine-tuning for tasks such as activity recognition, image classification, and object detection (Doersch, Gupta, and Efros 2015; Doersch and Zisserman 2017; Srivastava, Mansimov, and Salakhudinov 2015). More closely related to our work, several recent approaches have proposed leveraging widely available unlabeled video through self-supervised learning to directly train single-object tracking models, without needing fine-tuning. Vondrick et al. (Vondrick et al. 2018) train a model to colorize gray-scale video by propagating colors from a colored reference frame. The model is then applied to track objects at inference time by propagating instance IDs instead of colors. Wang et al. (Wang, Jabri, and Efros 2019) train a model to capture correspondence by applying a cycle-consistent loss: from a patch in an initial frame, after tracking forwards through video and then backwards to return to the initial frame, the final patch should align with the original patch.

Our work is also related to unsupervised detect-to-track multi-object tracking methods. SORT (Bewley et al. 2016) proposes modeling the motion of a track based only on spatial cues using a Kalman filter. To assign detections in the next frame to tracks, SORT formulates a bipartite matching problem where the cost is computed from the intersection-over-union score between the detection bounding box and the estimated track position. V-IOU (Bochinski, Senst, and Sikora 2018) incorporates visual cues into this framework, leveraging optical flows computed by comparing pixel values between pairs of consecutive frames to update the position of a track through segments where the detector fails to localize the track.

Several semi-supervised and fully supervised multi-object tracking methods have recently been proposed. Although these methods yield high-accuracy, they require video-level bounding box and track annotations that are expensive to hand-label, and thus are costly to extend to new types of video. Tracktor++ (Bergmann, Meinhardt, and Leal-Taixe 2019) applies the regression network in an object detector to localize a track in the next frame. MHT-BLSTM (Kim, Li, and Rehg 2018) applies a deep multiple hypothesis tracking approach, incorporating a bilinear LSTM to store long-term visual and spatial features and gate the track hypotheses. FAMNet (Chu and Ling 2019) jointly learns to extract features, estimate affinity, and assign track IDs in one network. LSST (Feng et al. 2019) combines a single-object tracking sub-network to capture short term cues with a re-identification sub-network to capture long-term cues.

3 Dual-Tracker Consistency

In our noval dual-tracker consistency method, we derive a self-supervisory signal for training an RNN tracker model through a three-step process. We assume that a corpus of unlabeled video is available, along with detections of the object category of interest computed automatically by an object detector in each video frame. First, during training, we repeatedly randomly sample a video segment $\langle I_0, \ldots, I_n \rangle$, where each I_k is a frame. Let D_k by the detections automatically computed in I_k, and let $d_i^k = (im, x, y, w, h)$ be a detection in D_k, where (x, y, w, h) are the 4D spatial coordinates (center point and lengths) of the detection bounding box, and im is the window of I_k corresponding to that box.

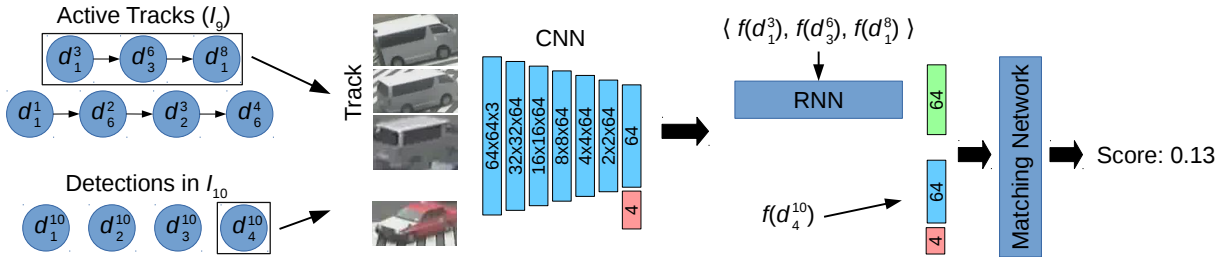

Figure 1: Tracker model architecture.

We apply an input-hiding scheme to select two input variations for the video segment, where each variation is a modified sequence of detections in the frames, $\langle D'_0, \ldots, D'_n \rangle$. For example, some detections may be removed entirely, while others may be partially hidden (e.g. im set to 0 so that one tracker instance only observes bounding box coordinates for those detections). Second, we apply two instances of the tracker model through each input variation to derive two probabilistic tracking outputs, represented as transition matrices. Third, we compare the transition matrices with dot-product similarity to update the RNN parameters.

Below, we first summarize the model architecture that we adapt from prior work in Section 3.1. We then detail our novel self-supervised training procedure, including the computation of the transition matrices and dot-product similarity loss, in Section 3.2. Finally, we propose two input-hiding schemes for selecting the input variations required by our approach in Section 4.

3.1 Tracker Model

We adopt a tracker model that is similar to prior work (Kim, Li, and Rehg 2018). We summarize the architecture in Figure 1. The tracker maintains a set of active tracks that have not yet left the camera frame. Given a video segment $\langle I_0, \ldots, I_n \rangle$, and sets of detections $D_k = \{d_1^k, \ldots, d_{m_k}^k\}$ detected in each frame I_k, to initialize the tracking process, we create a track $t_i = \langle d_i^0 \rangle$ for each detection d_i^0 in the first video frame I_0. On each subsequent frame I_k, the model outputs a probability that each active track t_i corresponds to each detection $d_j^k \in D_k$. At inference time, we apply the Hungarian algorithm to match active tracks with detections based on these probabilities. For each detection in I_k that no track matches to, we create a new active track for that detection. Similarly, if a track does not match to any detection for T_{age} consecutive frames, we remove it from the active set; thus, T_{age} is a threshold on the maximum age of a track since it last matched to some detection.

The model consists of a CNN, RNN, and matcher network. Together, these components score the likelihood that the ith active track, $t_i = \langle d_1, \ldots, d_m \rangle$, matches with the jth detection in I_k, d_j^k. We first apply the CNN to derive detection-level features. Given a detection $d = (im, x, y, w, h)$, the CNN inputs im resized to 64×64, and consists of 6 strided convolutional layers, with ReLU activation in the first 5 layers and linear activation in the last layer. It outputs a 64-vector, which we concatenate with the 4D spatial coordinates to derive a 68-vector detection representation $f(d)$.

Then, we compute track-level features $f(t_i)$ by applying an RNN model (specifically, an LSTM with 64 hidden states) over the sequence of detection-level features corresponding to the track, $\langle f(d_1), \ldots, f(d_m) \rangle$. The RNN outputs track-level features on each timestep, and its last output corresponds to features for the entire track computed so far, $f(t_i)$. Finally, we apply a matching network to score the likelihood that t_i matches d_j^k. The matching network inputs the concatenation of $f(t_i)$ and $f(d_j^k)$, applies four fully-connected layers, and outputs a match score.

3.2 Self-Supervised Training

We develop a novel self-supervised training procedure for learning the model parameters without using video-level bounding box and track annotations. During training, we repeatedly sample segments of up to 16 consecutive video frames $\langle I_0, \ldots, I_n \rangle$. We apply one of two input-hiding schemes, which we will detail in the following section, to extract two distinct input variations from a sampled video segment. We then apply instances of the tracker model on each variation, where the instances share the same model parameters. Dual-tracker consistency trains the model by enforcing it to produce similar outputs on both inputs. To represent tracker outputs, we compute a transition matrix $M^{(0,n)}$, where $M_{i,j}^{(0,n)}$ is the probability that the active track t_i matches d_j^n. When applying the model over video segments during training, we update tracks with new detections based on the scores output by the model on intermediate frames, but do not create additional active tracks on frames after I_0; thus, each active track t_i corresponds directly to a detection d_i^0 in I_0 (i.e., $t_i = \langle d_i^0, \ldots \rangle$). Then, applying two instances of the tracker yields two transition matrices $A^{(0,n)}$ and $B^{(0,n)}$, where both matrices have $|D_0|$ rows and $|D_n| + 1$ columns (the extra column represents tracks that are no longer visible). We train the model (CNN, RNN, and matching network) end-to-end to maximize the dot-product similarity between these matrices.

Below, we detail our method to compute transition matrices, and discuss dot-product similarity loss.

Transition Matrix. We propose computing a transition matrix $M^{(0,k)}$ on each frame I_k to represent the tracker outputs, so that we can compare the matrices produced under different input variations and back-propagate a similarity score.

$M_{i,j}^{(0,k)}$ is the probability that the ith track (which begins with d_i^0) matches d_j^k.

We first construct a score matrix $S^{(0,k)}$, where $S_{i,j}^{(0,k)}$ is the score computed through the matching network for the ith track and jth detection. We could simply compute $M^{(0,k)}$ by taking softmax along rows in $S^{(0,k)}$. However, computing the transition matrix in this way would allow the tracker to maximize dot-product similarity between $A^{(0,n)}$ and $B^{(0,n)}$ by matching all detections in I_0 to a single detection $d_j^n \in D_n$. Indeed, we find that in practice this leads to degenerate models.

Thus, instead, we compute $M^{(0,k),\text{row}}$ and $M^{(0,k),\text{col}}$ by applying softmax along rows and columns, respectively, and compute $M^{(0,k)} = \min(M^{(0,k),\text{row}}, M^{(0,k),\text{col}})$:

$$M_{i,j}^{\text{row}} = \frac{\exp(S_{i,j})}{\sum_k \exp(S_{i,k})} \qquad M_{i,j}^{\text{col}} = \frac{\exp(S_{i,j})}{\sum_k \exp(S_{k,j})} \quad (1)$$

$$M_{i,j} = \min(M_{i,j}^{\text{row}}, M_{i,j}^{\text{col}}) \quad (2)$$

This produces a transition matrix $M^{(0,k)}$ that is almost doubly stochastic: rows and columns sum to at most 1, but not necessarily exactly 1. The operation ensures that the model must match each detection in I_0 to unique detections in I_n to maximize the similarity score: if two detections in I_0 are matched to the same detection in I_n, then the columnar softmax would reduce those probabilities in the transition matrix to at most 0.5, thereby reducing any dot-products involving those rows.

So far, $M^{(0,k)}$ does not indicate the likelihood that a track fails to match to any detection. Determining when a track is not visible in a frame (it may be transiently occluded, or may have left the camera frame) is a key challenge in detect-to-track approaches. To represent this case in our transition matrix, we append an additional column to the score matrix $S^{(0,k)}$ for the "absent" case, where a track does not appear as a detection in I_k. The absent column is populated with a constant score $S_{i,\text{absent}} = s_{\text{absent}}$; the parameter s_{absent} is learned during training. Thus, if the ith track does not appear in a frame, the matching model should output low scores in $S^{(0,k)}$ when matching that track to detections in I_k, which will yield a large probability $M_{i,\text{absent}}^{(0,k)}$.

Dot-Product Similarity. We train the RNN tracker by computing two transition matrices $A^{(0,n)}$ and $B^{(0,n)}$ over different input variations, and then back-propagating a loss that measures the inconsistency between the matrices. In particular, we use the dot-product to measure the similarity of corresponding rows in the matrices. We define the loss as:

$$L = -\sum_i \log \sum_j A_{i,j}^{(0,n)} B_{i,j}^{(0,n)}$$

Here, L is computed by taking the logarithm of the dot product of corresponding rows in $A^{(0,n)}$ and $B^{(0,n)}$, averaged across rows. Note that this is equivalent to the cross-entropy loss between the diagonal matrix and the matrix product of $A^{(0,n)}$ and the transpose of $B^{(0,n)}$.

This loss function has several desirable properties. First, the dot-product is maximized when the matrices computed based on different inputs are most similar. This pushes the matching model to learn reasonable visual and spatial tracking constraints, because arbitrarily matching tracks to detections will lead to dissimilarity. Second, the dot-product pushes each matrix to be almost doubly stochastic rather than leaving some rows and columns summing to much less than 1. The model can only produce doubly stochastic $A^{(0,n)}$ and $B^{(0,n)}$ matrices by finding unique detections in I_n for each detection in I_0. Because the matching model is constrained to observing one track and one detection at a time, it must learn to match similar detections. Third, the loss formulation encourages shifting the probability mass along each row to a unique column – whereas assigning uniform probabilities would yield a dot-product of $\sum_{i=1}^{m}(1/m)^2 = 1/m$ along each row, finding a unique matching detection yields a dot-product of 1.

Spatial Mask. In some cases, training as described above may converge at a local minimum where the model outputs uniform probabilities for all entries in M. To mitigate this issue, we add a spatial constraint to the loss that penalizes the tracker when it matches detections that are highly improbable to correspond to the same object based on bounding box positions. We first compute a mask matrix $C^{(0,n)}$ so that $C_{i,j}^{(0,n)} = 0$ if it is "improbable" that d_i^0 matches d_j^n, and $C_{i,j}^{(0,n)} = 1$ otherwise. Then, we compute L as:

$$L = -\sum_i \log \sum_j A^{(0,n)} B^{(0,n)} C^{(0,n)}$$

We determine whether matches in C are improbable by applying a simple floodfill-like algorithm that propagates sets of labels from the first frame I_0 to the last frame I_n. If the label from a detection d_i^0 in I_0 does not propagate to a detection d_j^k, then it implies there is no sequence of intermediate detections that could form a track between d_i^0 and d_j^k. In I_0, we label each detection d_i^0 with a set containing only that detection, i.e., $\{d_i^0\}$. In I_k, we label each detection d_j^k with the union of sets of labels of detections d_i^{k-l} in preceding frames I_{k-l} ($1 \leq l \leq 10$) such that the bounding boxes of d_j^k and d_i^{k-l} intersect. Note that we consider several preceding frames since the detector may occasionally fail to localize an object in an intermediate frame. Then, $C_{i,j}^{(0,n)} = 1$ only if the label set for d_j^n includes d_i^0.

Artificial Detections. To improve the model's robustness in learning visual features, we artificially construct additional detections in I_n by pairing the spatial coordinates of detections in I_n with object images selected randomly from frames in the underlying video data that are temporally far from $\langle I_0, \ldots, I_n \rangle$. Thus, these artificial detections added to D_n have correct spatial coordinates, but include visual cues that do not correspond to any object in I_0, and so the tracker model must learn to leverage visual cues so that it does not assign high probabilities in $M(0, n)$ to artificial detections.

We exclude artificial detections in the mask C. Then, to perform well under dot-product similarity, the model must learn to leverage visual features to distinguish the correct detections in I_n from artificially constructed ones — a tracker

that only considers spatial features would assign half of its probability mass along each row to artificial detections, and thus would be penalized by the loss.

4 Input-Hiding Schemes

We develop two input-hiding schemes to select the two input variations, which we denote $A(D) = \langle D_0^A, \ldots, D_n^A \rangle$ and $B(D) = \langle D_0^B, \ldots, D_n^B \rangle$: occlusion-based hiding and visual-spatial hiding.

4.1 Occlusion-based Hiding

At a high level, occlusion-based hiding produces the variations $A(D)$ and $B(D)$ by simulating random occlusion incidents where all detections in occluded frames are eliminated from the input, i.e., if I_k is occluded for $A(D)$, then D_k^A is empty. We only occlude intermediate frames I_k, $0 < k < n$, so that the transition matrices still compare detections in I_0 with those in I_n. When processing an occluded I_k, the tracker is forced to match all tracks to the absent column in that frame, and re-localize the tracks after the occlusion.

We first introduce two schemes that do not work in isolation, and then show that we can combine these schemes to produce input variations that result in effective training.

Only-Occlusion. For each training sequence $\langle I_0, \ldots, I_n \rangle$, Only-Occlusion randomly selects four indexes $0 < k_1 \leq k_2 < k_3 \leq k_4 < n$ to construct two disjoint frame subsequences $\langle I_{k_1}, \ldots, I_{k_2} \rangle$ and $\langle I_{k_3}, \ldots, I_{k_4} \rangle$. In $A(D)$, we occlude each frame I_k such that $k_1 \leq k \leq k_2$, and in $B(D)$, we occlude I_k if $k_3 \leq k \leq k_4$.

When training under Only-Occlusion, the tracker is forced to leverage track features computed through the RNN to re-localize tracks after each simulated occlusion ends. Learning to merely compare detection features across consecutive frames would yield low accuracy since features in occluded frames are not observed. Furthermore, because one tracker observes the detections and the other tracker does not, the model must make similar tracking decisions when re-localizing across occluded frames as it does when observing detections in each frame.

However, in practice, Only-Occlusion produces a model that simply memorizes the detections in I_0, and computes $A^{(0,n)}$ and $B^{(0,n)}$ by comparing the detections in I_n against memorized detections. This strategy yields high similarity because it is unaffected by simulated occlusions in intermediate frames. Thus, we must prevent the propagation of features directly from I_0 to I_n to make this scheme effective.

RNN Hand-off. RNN Hand-off prevents simple memorization by cutting off the propagation of RNN features through the application of two separate RNN executions. We select two indexes $0 < k_5, k_6 < n$. Instead of computing $A^{(0,n)}$ directly, we first apply the tracker on the frame sequence $\langle I_0, \ldots, I_{k_5} \rangle$ to derive a transition matrix $A^{(0,k_5)}$ that matches detections in I_0 with detections in I_{k_5}. We then independently apply the tracker on $\langle I_{k_5}, \ldots, I_n \rangle$ to derive another matrix $A^{(k_5,n)}$ that matches detections in I_{k_5} with detections in I_n. We combine these matrices through the matrix product to compute $A^{(0,n)}$: we compute

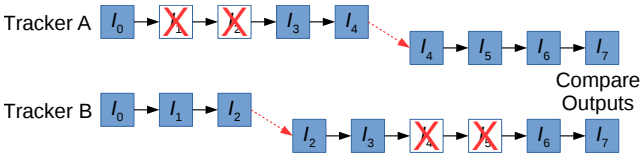

Figure 2: Occlusion-based hiding produces input variations with different subsequences of occluded frames where all detections are hidden from the tracker. It also independently applies the tracker before and after a hand-off frame (I_4 and I_2), and merges the outputs through the matrix product.

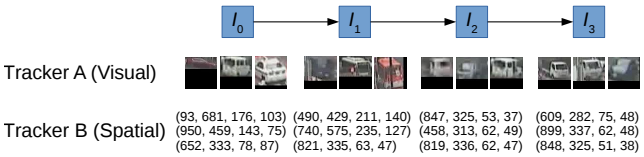

Figure 3: Visual-spatial hiding. One variation includes only visual inputs, and the other includes only spatial inputs.

$A^{(0,n)} = A^{(0,k_5)} A^{(k_5,n)}$. Similarly, we compute $B^{(0,n)} = B^{(0,k_6)} B^{(k_6,n)}$.

This scheme forces the tracker to find the same unique detection in I_{k_5} (and I_{k_6}) for two detections of the same object in I_0 and I_n in order to maximize similarity between the matrix products. On the other hand, though, a tracker that learns to match tracks to detections by comparing only the detection features in consecutive frames will exhibit high similarity between $A^{(0,n)}$ and $B^{(0,n)}$ under this scheme.

Combined Scheme. Only-Occlusion and RNN Hand-off have opposite advantages and opposite drawbacks. Thus, we combine these in our occlusion-based hiding scheme. We first select the two sequences for simulated occlusion, $\langle I_{k_1}, \ldots, I_{k_2} \rangle$ and $\langle I_{k_3}, \ldots, I_{k_4} \rangle$. Then, we randomly pick k_5 and k_6 such that $k_3 \leq k_5 \leq k_4$ and $k_1 \leq k_6 \leq k_2$, i.e., the hand-off for one tracker occurs when the other tracker observes a simulated occlusion. We summarize the scheme in Figure 2.

Under this scheme, neither memorizing features in I_0 nor comparing detections solely in a pairwise frame-by-frame manner is an effective tracking strategy. Instead, the tracker must learn to leverage RNN features for re-localizing across simulated occlusion, while still ensuring the tracking decisions reflect intermediate outputs.

4.2 Visual-Spatial Hiding

Under visual-spatial hiding, we apply one tracker instance that observes only visual features and one tracker instance that observes only spatial features: in $A(D)$, we set $x = 0, y = 0, w = 0, h = 0$ for all detections, and in $B(D)$, we set $im = 0$. By encouraging similarity in the tracking outputs, each tracker learns to leverage its input features and track objects as robustly as possible.

To prevent the visual tracker from examining the background to extract an approximation of the spatial features,

we do not use a recurrent unit for the visual tracker; instead, its matching network inputs visual features for detections in I_0 and for detections in I_n, and scores each pair of detections in $A^{(0,n)}$ without observing intermediate frames. We compute $B^{(0,n)}$ through the spatial tracker, which inputs 4D spatial coordinates for each detection, and processes frames with the matching network and recurrent unit. Figure 3 illustrates the training procedure.

Since the training process eliminates recurrent features for the visual tracker instance, we make corresponding adjustments to the inference process. First, during inference, we compute $M^{(0,k)}$ for an input video sequence $\langle I_0, \ldots, I_k \rangle$ as the minimum of the visual tracker output $A^{(0,k)}$ and the spatial tracker output $B^{(0,k)}$. Additionally, to compute scores in $A^{(0,k)}$, for each active track, we compute match scores between 5 randomly selected images associated with the track and each detection in I_k, and average these scores.

5 Evaluation

We compare our method and six baselines on the MOT17 challenge (Milan et al. 2016).

Baselines. We compare with the two unsupervised methods (SORT (Bewley et al. 2016) and V-IOU (Bochinski, Senst, and Sikora 2018)), one semi-supervised method (Tracktor++ (Bergmann, Meinhardt, and Leal-Taixe 2019)), and three fully supervised methods (MHT-BLSTM (Kim, Li, and Rehg 2018), FAMNet (Chu and Ling 2019), and LSST (Feng et al. 2019)) introduced in Section 2. Like our approach, SORT and V-IOU require image-level bounding box annotations (for training an object detector), but do not train on video-level bounding box and track annotations. The fully supervised methods train on video-level annotations; Tracktor++ also requires video-level annotations for training a re-identification network.

Dataset. The MOT17 dataset (Milan et al. 2016) consists of 14 video sequences of pedestrians in a wide range of contexts, including a moving camera inside a shopping mall and a fixed, elevated view of an outdoor plaza. The dataset is split into 7 training sequences and 7 test sequences; each split includes approximately 11 minutes of video.

Training. The semi-supervised and fully supervised baselines train on video-level bounding box and track annotations provided by MOT17. In contrast, our method trains only on a corpus of unlabeled video, and detections automatically inferred in the video by an object detector. Because video-level annotations are expensive to label, our method requires substantially less annotation time, and thus can greatly reduce the effort needed to apply accurate multi-object tracking on other datasets.

We train a Mask R-CNN object detector (He et al. 2017) on image-level bounding box annotations in the MOT17Det dataset; we use this detector only during training — for inference, we use detector outputs provided in the MOT17 dataset. We collect unlabeled video from two sources: we use five hours of video from seven YouTube walking tours, and all train and test sequences from the PathTrack dataset (Manen et al. 2017) (we do not use the PathTrack ground truth annotations).

We train our tracker model on an NVIDIA Tesla V100 GPU; training time varies between 24 and 48 hours depending on the input-hiding scheme. During training, we select sequence lengths n between 4 and 16, and apply stochastic gradient descent one sequence at a time. We apply the Adam optimizer with learning rate 0.0001.

Metrics. We evaluate the methods in terms of the Multi-Object Tracking Accuracy (MOTA) (Milan et al. 2016) and ID F1 Score (IDF1) (Ristani et al. 2016) metrics. Ground truth and inferred tracks are both sequences of detections (bounding boxes); broadly, MOTA and IDF1 measure the accuracy of the inferred tracks, and penalize both when an inferred track contains a detection that doesn't match to some ground truth detection (or vice versa), and when a ground truth track is split into two or more inferred tracks (or vice versa). We include other MOT metrics reported by MOT17, but focus on IDF1 and MOTA as they are the most widely used and the most comprehensive.

The MOT17 challenge provides 3 detectors: deformable part-based model (DPM) (Felzenszwalb et al. 2009), Faster-RCNN (FRCNN) (Ren et al. 2015), and Scale-Dependent Pooling (SDP) (Yang, Choi, and Lin 2016). We find that DPM is substantially less accurate than FRCNN and SDP, and that both our approach and all six baselines yield the highest average performance using SDP. Since MOT17 reports scores aggregated over all detectors, methods that improve accuracy when using DPM score higher on the aggregated metrics despite not improving accuracy in realistic tracking scenarios where higher-quality detections are available; for example, Tracktor++ artificially improves performance given poor detections by incorporating a high-accuracy object detector in their approach, and using this detector during inference to prune incorrect input detections. Thus, we instead report all metrics averaged over the 7 sequences using the detector on which each approach obtains the highest performance (which is SDP in all cases).

Ablation Study and Unsupervised Baselines. We first compare occlusion-based hiding, visual-spatial hiding, and the unsupervised baselines on the MOT17 training set in Table 1. None of these methods train on MOT17 training annotations. Visual-spatial hiding yields high performance on IDF1 and MOTA, improving over SORT and V-IOU by 4% on both metrics. On the other hand, occlusion-based hiding performs poorly on MOT17; objects are often visible in the video for only a short duration, making it challenging to learn to re-localize objects over simulated occlusion since the simulated occlusion must then also be short.

Under Spatial-Only, we show results for visual-spatial hiding when inputting only the spatial bounding box coordinates of detections during inference (no image features). Even when inputting only spatial features, our method improves accuracy over SORT and V-IOU, suggesting that the model learns spatial patterns that heuristics used in SORT and V-IOU do not effectively capture.

Quantitative Results. Table 2 shows results on the MOT17

Table 1: Ablation study and comparison with unsupervised baselines on the MOT17 training set. For each method, we compute average metrics over the 7 test sequences on each detection set, and report the metrics for the detections on which the method obtains highest MOTA.

Method	IDF1	MOTA	MT	ML	FP	FN	IDSW	Frag
Occlusion (ours)	52.4	56.7	28.7	16.9	1K	5K	136	172
Visual-Spatial (ours)	**57.3**	**60.2**	27.0	20.1	734	5K	83	118
Spatial-Only (ours)	56.5	57.8	27.1	18.3	1K	5K	144	169
SORT (Bewley et al. 2016)	53.4	56.0	20.4	24	457	6K	89	116
V-IOU (Bochinski, Senst, and Sikora 2018)	47.4	56.3	22.6	19.9	577	6K	233	217

Table 2: Performance on the MOT17 test set. For each method, we compute average metrics over the 7 test sequences on each detection set, and report the metrics for the detections on which the method obtains highest MOTA.

Method	IDF1	MOTA	MT	ML	FP	FN	IDSW	Frag
Visual-Spatial (ours)	**50.3**	**46.5**	24.6	34.6	877	10K	100	175
SORT (Bewley et al. 2016)	43.9	43.6	18.4	37.0	438	11K	203	345
IOU (Bochinski, Eiselein, and Sikora 2017)	40.8	44.3	22.7	31.6	857	10K	458	544
Tracktor++ (Bergmann, Meinhardt, and Leal-Taixe 2019)	46.6	43.8	20.3	36.9	768	11K	113	262
MHT-BLSTM (Kim, Li, and Rehg 2018)	48.8	43.7	22.9	39.1	812	11K	102	148
FAMNet (Chu and Ling 2019)	46.5	48.8	23.2	30.2	689	10K	157	281
LSST (Feng et al. 2019)	**59.1**	**49.4**	30.1	33.9	1K	9K	66	193

Table 3: Performance on the MOT17 test set, where metrics are averaged over the 3 detection sets.

Method	IDF1	MOTA	MT	ML	FP	FN	IDSW	Frag
Visual-Spatial (ours)	51.4	51.3	432	939	13,599	259,389	1,872	3,303
SORT (Bewley et al. 2016)	39.8	43.1	295	997	28,398	287,582	4,852	7,127
IOU (Bochinski, Eiselein, and Sikora 2017)	39.4	45.5	369	953	19,993	281,643	5,988	7,404
Tracktor++ (Bergmann, Meinhardt, and Leal-Taixe 2019)	52.3	53.5	459	861	12,201	248,047	2,072	4,611
MHT-BLSTM (Kim, Li, and Rehg 2018)	51.9	47.5	429	981	25,981	268,042	2,069	3,124
FAMNet (Chu and Ling 2019)	48.7	52.0	450	787	14,138	253,616	3,072	5,318
LSST (Feng et al. 2019)	62.3	54.7	480	944	26,091	228,434	1,243	3,726

test set[1]; metrics are automatically computed by the challenge website. Per the challenge policy, we only submit the best method, and thus show Visual-Spatial performance.

Our approach again outperforms the unsupervised and semi-supervised baselines. Moreover, its performance is competitive with two recently proposed supervised tracking methods, MHT-BLSTM (Kim, Li, and Rehg 2018) and FAMNet (Chu and Ling 2019): comparing against FAMNet, our approach improves in IDF1 but yields lower MOTA. Thus, we have shown that by leveraging unlabeled video, our self-supervised training procedure performs competitively with two recent MOT methods that require expensive video-level annotations. Nevertheless, LSST (Feng et al. 2019) yields higher accuracy on both metrics.

In Table 3, we show results when averaging metrics over the 3 MOT17 detection sets rather than showing metrics over the detection set on which each method performs the best. This is the default aggregation method computed used in the challenge; in some cases, metrics are higher since the challenge uses an unknown weighting when averaging metrics across the 7 test sequences. The relative performance of our method against baselines is similar except for Tracktor++ (Bergmann, Meinhardt, and Leal-Taixe 2019), which achieves higher IDF1 and MOTA. As we argued above, this is because Tracktor++ artificially improves performance on the low-quality detection sets by incorporating a high-accuracy object detector in their approach, and using this detector during inference to prune incorrect input detections.

Qualitative Results. We show qualitative results in the supplementary video.

6 Conclusion

In this paper, we have shown that a robust multi-object tracker can be trained through a novel self-supervisory signal, dual-tracker consistency, that enforces consistency in the tracking outputs across different input variations of one video sequence. Despite using only unlabeled video and an object detector during training, our approach is competitive on MOT17 with two recent state-of-the-art supervised trackers, which train on expensive video-level bounding box and track annotations.

[1]These results are taken from https://motchallenge.net/results/MOT17/, where our method is denoted UNS20. There, performance on specific detections is shown under Detailed Performance after selecting a tracking method. Baselines are denoted SORT17, IOU17, Tracktor++, MHT_bLSTM, FAMNet, and LSST17.

References

Bergmann, P.; Meinhardt, T.; and Leal-Taixe, L. 2019. Tracking without bells and whistles. In *IEEE International Conference on Computer Vision (ICCV)*.

Bewley, A.; Ge, Z.; Ott, L.; Ramos, F.; and Upcroft, B. 2016. Simple Online and Realtime Tracking. In *IEEE International Conference on Image Processing (ICIP)*.

Bochinski, E.; Eiselein, V.; and Sikora, T. 2017. High-Speed Tracking-by-Detection Without Using Image Information. In *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*.

Bochinski, E.; Senst, T.; and Sikora, T. 2018. Extending IOU Based Multi-Object Tracking by Visual Information. In *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*.

Chu, P.; and Ling, H. 2019. FAMNet: Joint Learning of Feature, Affinity and Multi-dimensional Assignment for Online Multiple Object Tracking. In *IEEE International Conference on Computer Vision (ICCV)*.

Doersch, C.; Gupta, A.; and Efros, A. A. 2015. Unsupervised Visual Representation Learning by Context Prediction. In *IEEE International Conference on Computer Vision (ICCV)*.

Doersch, C.; and Zisserman, A. 2017. Multi-task Self-Supervised Visual Learning. In *IEEE International Conference on Computer Vision (ICCV)*.

Felzenszwalb, P. F.; Girshick, R. B.; McAllester, D.; and Ramanan, D. 2009. Object Detection with Discriminatively Trained Part-Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32(9): 1627–1645.

Feng, W.; Hu, Z.; Wu, W.; Yan, J.; and Ouyang, W. 2019. Multi-Object Tracking with Multiple Cues and Switcher-Aware Classification. *arXiv preprint arXiv:1901.06129* .

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. 2017. Mask R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*.

Kim, C.; Li, F.; and Rehg, J. M. 2018. Multi-Object Tracking with Neural Gating using Bilinear LSTM. In *European Conference on Computer Vision (ECCV)*.

Leal-Taixé, L.; Milan, A.; Reid, I.; Roth, S.; and Schindler, K. 2015. MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking. *arXiv preprint arXiv:1504.01942* .

Manen, S.; Gygli, M.; Dai, D.; and Van Gool, L. 2017. PathTrack: Fast Trajectory Annotation with Path Supervision. In *IEEE International Conference on Computer Vision (ICCV)*.

Milan, A.; Leal-Taixé, L.; Reid, I.; Roth, S.; and Schindler, K. 2016. MOT16: A Benchmark for Multi-Object Tracking. *arXiv preprint arXiv:1603.00831* .

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Ristani, E.; Solera, F.; Zou, R.; Cucchiara, R.; and Tomasi, C. 2016. Performance Measures and a Data Set for Multi-Target, Multi-Camera Tracking. In *European Conference on Computer Vision (ECCV)*.

Srivastava, N.; Mansimov, E.; and Salakhudinov, R. 2015. Unsupervised Learning of Video Representations using LSTMs. In *International Conference on Machine Learning (ICML)*.

Vondrick, C.; Shrivastava, A.; Fathi, A.; Guadarrama, S.; and Murphy, K. 2018. Tracking Emerges by Colorizing Videos. In *European Conference on Computer Vision (ECCV)*.

Wang, X.; Jabri, A.; and Efros, A. A. 2019. Learning Correspondence from the Cycle-Consistency of Time. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yang, F.; Choi, W.; and Lin, Y. 2016. Exploit All the Layers: Fast and Accurate CNN Object Detector with Scale Dependent Pooling and Cascaded Rejection Classifiers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yuen, J.; Russell, B.; Liu, C.; and Torralba, A. 2009. LabelMe video: Building a Video Database with Human Annotations. In *IEEE International Conference on Computer Vision (ICCV)*.