Online Multi-Target Tracking Using Recurrent Neural Networks

Anton Milan,[1] **S. Hamid Rezatofighi,**[1] **Anthony Dick,**[1] **Ian Reid,**[1] **Konrad Schindler**[2]
[1]School of Computer Science, The University of Adelaide
[2]Photogrammetry and Remote Sensing, ETH Zürich
[1]{firstname.lastname}@adelaide.edu.au, [2]schindler@geod.baug.ethz.ch

Abstract

We present a novel approach to online multi-target tracking based on recurrent neural networks (RNNs). Tracking multiple objects in real-world scenes involves many challenges, including *a)* an a-priori unknown and time-varying number of targets, *b)* a continuous state estimation of all present targets, and *c)* a discrete combinatorial problem of data association. Most previous methods involve complex models that require tedious tuning of parameters. Here, we propose for the first time, an end-to-end learning approach for online multi-target tracking. Existing deep learning methods are not designed for the above challenges and cannot be trivially applied to the task. Our solution addresses all of the above points in a principled way. Experiments on both synthetic and real data show promising results obtained at ≈300 Hz on a standard CPU, and pave the way towards future research in this direction.

Introduction

Tracking multiple targets in unconstrained environments is extremely challenging. Even after several decades of research, it is still far from reaching the accuracy of human labelling. (*cf*. *MOTChallenge* (Leal-Taixé et al. 2015)). The task itself constitutes locating all targets of interest in a video sequence and maintaining their identity over time. One of the obvious questions that arises immediately is how to model the vast variety of data present in arbitrary videos that may include different view points or camera motion, various lighting conditions or levels of occlusion, a varying number of targets, *etc*. Tracking-by-detection has emerged as one of the most successful strategies to tackle this challenge. Here, all "unused" data that is available in a video sequence is discarded and reduced to just a few single measurements per frame, typically by running an object detector. The task is then to associate each measurement to a corresponding target, *i.e*. to address the problem of data association. Moreover, due to clutter and an unknown number of targets, the option to discard a measurement as a false alarm and a strategy to initiate new targets as well as terminate exiting ones must be addressed.

With the recent rise of deep learning, there has been surprisingly little work related to multi-target tracking. We presume that this is due to several reasons. First, when dealing with a large number of parameters, deep models require huge amounts of training data, which is not yet available in the case of multi-target tracking. Second, both the data and the desired solution can be quite variable. One is faced with both discrete and continuous variables, unknown cardinality for input and output, and variable lengths of video sequences. One interesting exception in this direction is the recent work of Ondrúška and Posner (2016) that introduces deep recurrent neural networks to the task of state estimation. Although this work shows promising results, it only demonstrates its efficacy on simulated data with near-perfect sensor measurements, a known number of targets, and smooth, linear motion. Their follow-up work introduces real-world measurements and multi-class scenarios (Ondruska et al. 2016), however, in both cases, tracking is formulated as estimating the world occupancy, without explicit data association.

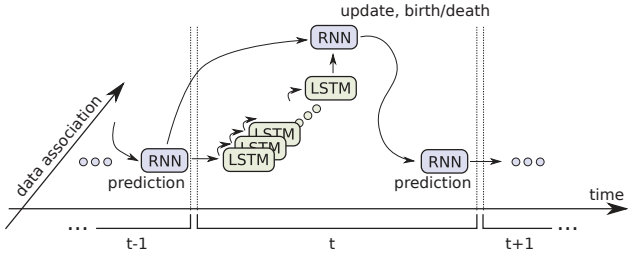

Figure 1: A schematic illustration of our architecture. We use RNNs for temporal prediction and update as well as track management. The combinatorial problem of data association is solved via LSTMs for each frame.

With this paper, we make an important step towards end-to-end model learning for online tracking of multiple targets in realistic scenarios. Our main contributions are as follows:

1. Inspired by the well-studied Bayesian filtering idea, we present a recurrent neural network capable of performing all multi-target tracking tasks including prediction, data association, state update as well as initiation and termination of targets within a unified network structure (Fig. 1). One of the main advantages of this approach is that it is completely model-free, *i.e*. it does not require any prior knowledge about target dynamics, clutter distributions, *etc*. It can therefore capture linear (*cf*. Kalman filter), non-

linear (*cf*. particle filter), and higher-order dependencies.

2. We further show, that a model for the challenging combinatorial problem of data association including birth and death of targets can be learned entirely from data. This time-varying cardinality component demonstrates that it is possible to utilise RNNs not only to predict sequences with fixed-sized input and output vectors, but in fact to infer unordered sets with unknown cardinality.
3. We present a way to generate arbitrary amounts of training data by sampling from a generative model.
4. Qualitative and quantitative results on simulated and real data show encouraging results, confirming the potential of this approach. We firmly believe that it will inspire other researchers to extend the presented ideas and to further advance the performance.

Related Work

Multi-object tracking. A multitude of sophisticated models have been developed in the past to capture the complexity of the problem at hand. Early works include the multiple hypothesis tracker (MHT) (Reid 1979) and joint probabilistic data association (JPDA) (Fortmann, Bar-Shalom, and Scheffe 1980). Both were developed in the realm of radar and sonar tracking but were considered too slow for computer vision applications for a long time. With the advances in computational power, they have found their way back and have recently been re-introduced in conjunction with novel appearance models (Kim et al. 2015), or suitable approximation methods (Rezatofighi et al. 2015). Recently, a large amount of work focused on simplified models that could be solved to (near) global optimality (Jiang, Fels, and Little 2007; Zhang, Li, and Nevatia 2008; Berclaz et al. 2011; Butt and Collins 2013). Here, the problem is cast as a linear program and solved via relaxation, shortest-path, or min-cost algorithms. Conversely, more complex cost functions have been considered in (Leibe, Schindler, and Van Gool 2007; Milan, Roth, and Schindler 2014), but without any theoretical bounds on optimality. The optimization techniques range from quadratic boolean programming, over customised alpha-expansion to greedy constraint propagation. More recently, graph multi-cut formulations (Tang et al. 2016) have also been employed.

Deep learning. Early ideas of biologically inspired learning systems date back many decades (Ivakhnenko and Lapa 1966). Later, convolutional neural networks (also known as CNNs) and the back propagation algorithm were developed and mainly applied to hand-written digit recognition (LeCun et al. 1998). However, despite their effectiveness on certain tasks, they could hardly compete with other well-established approaches. This was mainly due to their major limitation of requiring huge amounts of training data in order not to overfit the high number of parameters. With faster multi-processor hardware and with a sudden increase in labelled data, CNNs have become increasingly popular, initiated by a recent breakthrough on the task of image classification (Krizhevsky, Sutskever, and Hinton 2012). CNNs achieve state-of-the-art results in many applications (Wang et al. 2012; Eigen and Fergus 2015) but are restrictive in their output format. Conversely, *recurrent* neural networks (RNNs) (Goller and Küchler 1996) include a loop between the input and the output. This not only enables to simulate a memory effect, but also allows for mapping input sequences to arbitrary output sequences, as long as the sequence alignment and the input and output dimensions are known in advance.

Our work is inspired by the recent success of recurrent neural nets (RNNs) and their application to language modeling (Vinyals et al. 2015). However, it is not straightforward to apply the same strategies to the problem of multi-target tracking for numerous reasons. First, the state space is multi-dimensional. Instead of predicting one character or one word, at each time step the state of all targets should be considered at once. Second, the state consists of both continuous and discrete variables. The former represents the actual location (and possibly further properties such as velocities) of targets, while a discrete representation is required to resolve data association. Further indicator variables may also be used to infer certain target states like the track state, the occlusion level, *etc*. Third, the desired number of outputs (*e.g*. targets) varies over time. In this paper, we introduce a method for addressing all these issues and demonstrate how RNNs can be used for end-to-end learning of multi-target tracking systems.

Background

Recurrent Neural Networks

Broadly speaking, RNNs work in a sequential manner, where a prediction is made at each time step, given the previous state and possibly an additional input. The core of an RNN is its hidden state $h \in \mathbb{R}^n$ of size n that acts as the main control mechanism for predicting the output, one step at a time. In general, RNNs may have L layers. We will denote h_t^l as the hidden state at time t on layer l. h^0 can be thought of as the input layer, holding the input vector, while h^L holds the final embedded representation used to produce the desired output y_t. The hidden state for a particular layer l and time t is computed as $h_t^l = \tanh W^l \left(h_t^{l-1}, h_{t-1}^l\right)^\top$, where W is a matrix of learnable parameters.

The RNN as described above performs well on the task of motion prediction and state update. However, we found that it cannot properly handle the combinatorial task of data association. To that end, we consider the long short-term memory (LSTM) recurrence (Hochreiter and Schmidhuber 1997). Next to the hidden state, the LSTM unit also keeps an embedded representation of the state c that acts as a memory. A gated mechanism controls how much of the previous state should be "forgotten" or replaced by the new input (see Fig. 2, right, for an illustration). More formally, the hidden representations are computed as $h_t^l = o \odot \tanh\left(c_t^l\right)$ and $c_t^l = f \odot c_{t-1}^l + i \odot g$, where $\odot$ represents element-wise multiplication. The input, output and forget gates are all vectors of size n and model the memory update in a binary fashion

using a sigmoid function:

$$i, o, f = \sigma \left[W^l \left(h_t^{l-1}, h_{t-1}^l \right)^\top \right], \quad (1)$$

with a separate weight matrix W^l for each gate.

Bayesian Filtering

In Bayseian filtering, the goal is to estimate the true state x from noisy measurements z. Under the Markov assumption, the state distribution at time t given all past measurements is estimated recursively as

$$p(x_t|z_{1:t}) \propto p(z_t|x_t) \int p(x_t|x_{t-1}) p(x_{t-1}|z_{1:t-1}) dx_{t-1}, \quad (2)$$

where $p(z_t|x_t)$ is the last observation likelihood and $p(x_t|x_{t-1})$ the state transition probability. Typically, Eq. (2) is evaluated in two steps: a *prediction* step that evaluates the state dynamics, and an *update* step that corrects the belief about the state based on the current measurements. Two of the most widely used techniques for solving the above equation are Kalman filter (Kalman 1960) and particle filter (Doucet, Godsill, and Andrieu 2000). The former performs exact state estimation under linear and Gaussian assumptions for the state and measurements models, while the latter approximates arbitrary distributions using sequential importance sampling.

When dealing with multiple targets, one is faced with two additional challenges. 1) Before the state update can be performed, it is crucial to determine which measurements are associated with which targets. A number of algorithms have been proposed to address this problem of data association including simple greedy techniques, and sophisticated probabilistic approaches like JPDA (see (Bar-Shalom and Fortmann 1988) for an overview). 2) To allow for a time-varying number of targets, it is necessary to provide a mechanism to spawn new targets that enter the scene, and remove existing ones that disappear indefinitely. Like data association, this task is non-trivial, since each unassigned measurement can potentially be either the start of a new trajectory or a false alarm. Conversely, a missing measurement for a certain target could mean that the target has disappeared, or that the detector has failed. To address this challenge, online tracking approaches typically base their decisions about births and deaths of tracks on heuristics that consider the number of consecutive measurement errors.

Our Approach

We will now describe our approach to cast the classical Bayesian state estimation, data association as well as track initiation and termination tasks as a recurrent neural net, allowing for full end-to-end learning of the model.

Preliminaries and Notation

We begin by defining $x_t \in \mathbb{R}^{N \cdot D}$ as the vector containing the states for all targets at one time instance. In our setting, the targets are represented by their bounding box coordinates (x, y, w, h), such that $D = 4$. Note that it is conceptually straightforward to extend the state to an arbitrary dimension, *e.g.* to incorporate velocity, acceleration or appearance model. N is the number of interacting targets that are represented (or tracked) simultaneously in one particular frame and x_t^i refers to the state of the i^{th} target. N is what we call the network's *order* and captures the spatial dependencies between targets. Here, we consider a special case with $N = 1$ where all targets are assumed to move independently. In other words, the same RNN is used for each target. Similar to the state vector above, $z_t \in \mathbb{R}^{M \cdot D}$ is the vector of all measurements in one frame, where M is maximum number of detections per frame.

The assignment probability matrix $\mathcal{A} \in [0, 1]^{N \times (M+1)}$ represents for each target (row) the distribution of assigning individual measurements to that target, *i.e.* $\mathcal{A}_{ij} \equiv p(i \text{ assigned to } j)$ and $\forall i : \sum_j \mathcal{A}_{ij} = 1$. Note that an extra column in $\mathcal{A}$ is needed to incorporate the case that a measurement is missing. Finally, $\mathcal{E} \in [0, 1]^N$ is an indicator vector that represents the existence probability of a target and is necessary to deal with an unknown and time-varying number of targets. We will use $(\sim)$ to explicitly denote the ground truth variables.

Multi-Target Tracking with RNNs

As motivated above, we decompose the problem at hand into two major blocks: state prediction and update as well as track management on one side, and data association on the other. This strategy has several advantages. First, one can isolate and debug individual components effectively. Second, the framework becomes modular, making it easy to replace each module or to add new ones. Third, it enables one to (pre)train every block separately, which not only significantly speeds up the learning process but turns out to be necessary in practice to enable convergence. We will now describe both building blocks in detail.

Target Motion

Let us first turn to state prediction and update. We rely on a temporal RNN depicted in Fig. 2 (left) to learn the temporal dynamic model of targets as well as an indicator to determine births and deaths of targets (see next section). At time t, the RNN outputs four values[1] for the next time step: A vector $x_{t+1}^* \in \mathbb{R}^{N \cdot D}$ of predicted states for all targets, a vector $x_{t+1} \in \mathbb{R}^{N \cdot D}$ of all updated states, a vector $\mathcal{E}_{t+1} \in (0, 1)^N$ of probabilities indicating for each target how likely it is a real trajectory, and $\mathcal{E}_{t+1}^*$, which is the absolute difference to $\mathcal{E}_t$. This decision is computed based on the current state x_t and existence probabilities $\mathcal{E}_t$ as well as the measurements z_{t+1} and data association $\mathcal{A}_{t+1}$ in the following frame. This building block has three primary objectives:

1. **Prediction**: Learn a complex dynamic model for predicting target motion in the absence of measurements.
2. **Update**: Learn to correct the state distribution, given target-to-measurement assignments.

[1] We omit the RNN's hidden state h_t at this point in order to reduce notation clutter.

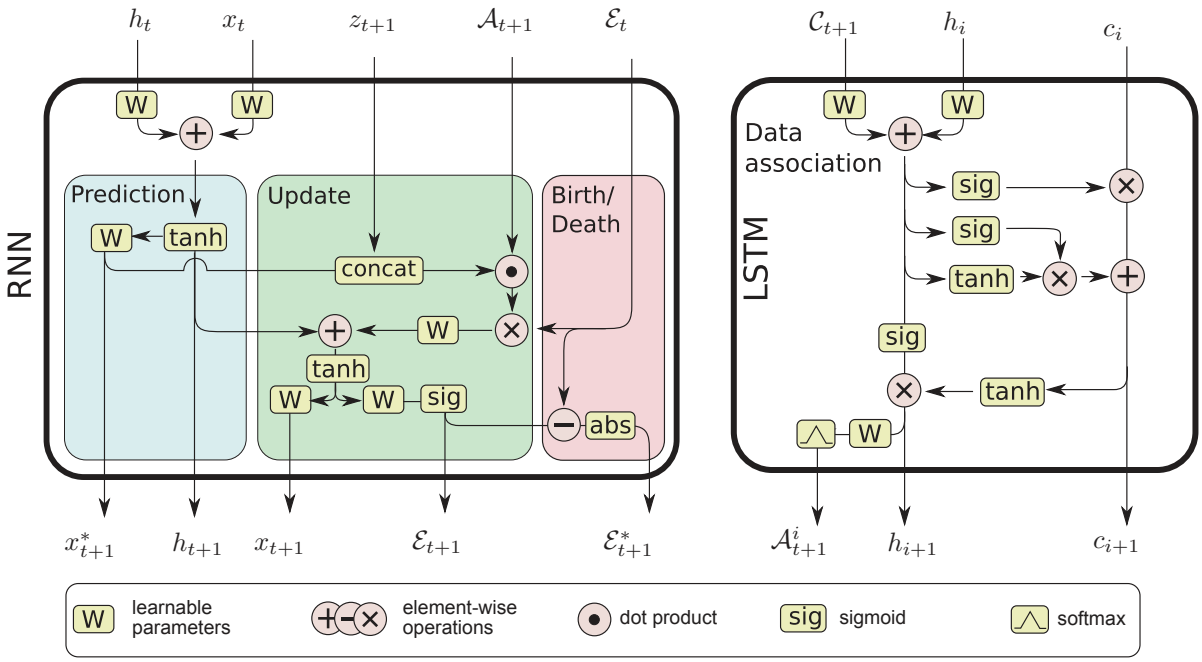

Figure 2: *Left:* An RNN-based architecture for state prediction, state update, and target existence probability estimation. *Right:* An LSTM-based model for data association.

3. **Birth / death**: Learn to identify track initiation and termination based on the state, the measurements and the data association.

The prediction x^*_{t+1} for the next frame depends solely on the current state x_t and the network's hidden state h_t. Once the data association $\mathcal{A}_{t+1}$ for the following frame is available, the state is updated according to assignment probabilities. To that end, all measurements and the predicted state are concatenated to form $\hat{x} = [z_{t+1}; x^*_{t+1}]$ weighted by the assignment probabilities $\mathcal{A}_{t+1}$. This is performed for all state dimensions. At the same time, the track existence probability $\mathcal{E}_{t+1}$ for the following frame is computed.

Loss. A loss or objective is required by any machine learning algorithm to compute the goodness-of-fit of the model, *i.e.* how close the prediction corresponds to the true solution. It is typically a continuous function, chosen such that minimising the loss maximises the performance of the given task. In our case, we are therefore interested in a loss that correlates with the tracking performance. This poses at least two challenges. First, measuring the performance of multi-target tracking is far from trivial (Milan, Schindler, and Roth 2013) and moreover highly dependent on the particular application. For example, in vehicle assistance systems it is absolutely crucial to maintain the highest precision and recall to avoid accidents and to maintain robustness to false positives. On the other hand, in sports analysis it becomes more important to avoid ID switches between different players. One of the most widely accepted metrics is the multi-object tracking accuracy (MOTA) (Bernardin and Stiefelhagen 2008) that combines the three error types mentioned above and gives a reasonable assessment of the overall performance. Ideally, one would train an algorithm directly on the desired performance measure. This, however, poses a second challenge. The MOTA computation involves a complex algorithm with non-differentiable zero-gradient components, that cannot easily be incorporated into an analytical loss function. Hence, we propose the following loss that satisfies our needs:

$$\mathcal{L}(x^*, x, \mathcal{E}, \widetilde{x}, \widetilde{\mathcal{E}}) = \underbrace{\frac{\lambda}{ND}\sum \|x^* - \widetilde{x}\|^2}_{\text{prediction}} + \underbrace{\frac{\kappa}{ND}\|x - \widetilde{x}\|^2}_{\text{update}} + \underbrace{\nu\mathcal{L}_{\mathcal{E}} + \xi\mathcal{E}^*}_{\text{birth/death + reg.}}, \tag{3}$$

where x^*, x, and $\mathcal{E}$ are the predicted values, and $\widetilde{x}$ and $\widetilde{\mathcal{E}}$ are the true values, respectively. Note that we omit the time index here for better readability. In practice the loss for one training sample is averaged over all frames in the sequence.

The loss consists of four components. Let us first concentrate on the first two, assuming for now that the number of targets is fixed. Intuitively, we aim to learn a network that predicts trajectories that are close to the ground truth tracks. This should hold for both, predicting the target's motion in the absence of any measurements, as well as correcting the track in light of new measurements. To that end, we minimise the mean squared error (MSE) between state predictions and state update and the ground truth.

Initiation and Termination

Tracking multiple targets in real-world situations is complicated by the fact that targets can appear and disappear in the

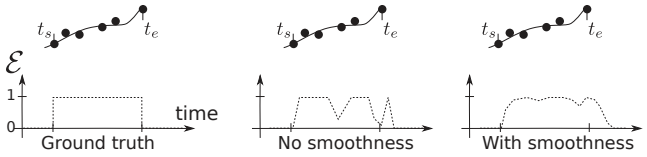

Figure 3: The effect of the pairwise smoothness prior on the existence probability. See text for details.

area of interest. This aspect must not be ignored but is difficult to model within the fixed-sized vector paradigm in traditional neural network architectures. We propose to capture the time-varying number of targets by an additional variable $\mathcal{E} \in (0,1)^N$ that mimics the probability that a target exists ($\mathcal{E} = 1$) or not ($\mathcal{E} = 0$) at one particular time instance. At test time, we then simply discard all targets for which $\mathcal{E}$ is below a threshold (0.6 in our experiments).

Loss. The last two terms of the loss in Eq. (3) guide the learning to predict the existence of each target at any given time. This is necessary to allow for target initiation and termination. Here, we employ the widely used binary cross entropy (BCE) loss

$$\mathcal{L}_{\mathcal{E}}(\mathcal{E}, \widetilde{\mathcal{E}}) = \widetilde{\mathcal{E}} \log \mathcal{E} + (1 - \widetilde{\mathcal{E}}) \log(1 - \mathcal{E}) \qquad (4)$$

that approximates the probability of the existence for each target. Note that the true values $\widetilde{\mathcal{E}}$ here correspond to a box function over time (*cf*. Fig. 3, left). When using the BCE loss alone, the RNN learns to make rather hard decisions, which results in track termination at each frame when a measurement is missing. To remedy this, we propose to add a smoothness prior $\mathcal{E}^*$ that essentially minimises the absolute difference between two consecutive values for $\mathcal{E}$.

Data Association with LSTMs

Arguably, the data association, *i.e*. the task to uniquely classify the corresponding measurement for each target, is the most challenging component of tracking multiple targets. Greedy solutions are efficient, but do not yield good results in general, especially in crowded scenes with clutter and occlusions. Approaches like JPDA are on the other side of the spectrum. They consider *all* possible assignment hypotheses jointly, which results in an NP-hard combinatorial problem. Hence, in practice, efficient approximations must be used.

In this section, we describe an LSTM-based architecture that is able to learn to solve this task entirely from training data. This is somewhat surprising for multiple reasons. First, joint data association is in general a highly complex, discrete combinatorial problem. Second, most solutions in the output space are merely permutations of each other w.r.t. the input features. Finally, any possible assignment should meet the one-to-one constraint to prevent the same measurement to be assigned to multiple targets. We believe that the LSTM's non-linear transformations and its strong memory component are the main driving force that allows for all these challenges to be learned effectively. To support this claim, we demonstrate the capability of LSTM-based data association on the example of replicating the linear assignment problem.

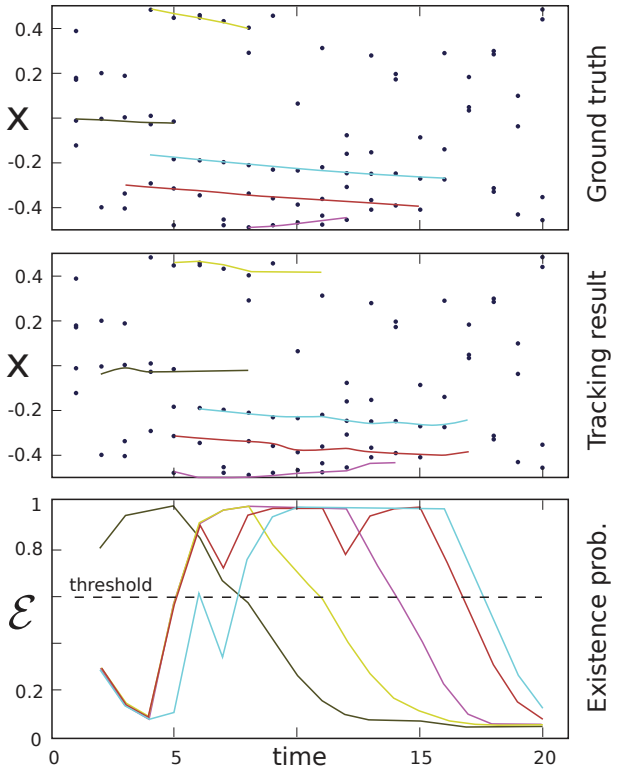

Figure 4: Results of our tracking method on a 20-frame long synthetic sequence with clutter. **Top:** Ground truth (x-coordinate vs. time). **Middle:** Our reconstructed trajectories. **Bottom:** The existence probability $\mathcal{E}$ for each target. Note the delayed initiation and termination, *e.g*. for the top-most track (yellow) in the middle. This an inherent limitation of any purely online approach that cannot be avoided.

Our model is illustrated in Figures 1 and 2 (right). The main idea is to exploit the LSTM's temporal step-by-step functionality to predict the assignment for each target one target at a time. The input at each step i, next to the hidden state h_i and the cell state c_i, is the *entire* feature vector. For our purpose, we use the pairwise-distance matrix $\mathcal{C} \in \mathbb{R}^{N \times M}$, where $\mathcal{C}_{ij} = \|x^i - z^j\|_2$ is the Euclidean distance between the predicted state of target i and measurement j. Note that it is straight-forward to extend the feature vector to incorporate appearance or any other similarity information. The output that we are interested in is then a vector of probabilities $\mathcal{A}^i$ for one target and all available measurements, obtained by applying a softmax layer with normalisation to the predicted values. Here, $\mathcal{A}^i$ denotes the i^{th} row of $\mathcal{A}$.

Loss. To measure the misassignment cost, we employ the common negative log-likelihood loss

$$\mathcal{L}(\mathcal{A}^i, \tilde{a}) = -\log(\mathcal{A}_{i\tilde{a}}), \qquad (5)$$

where $\tilde{a}$ is the correct assignment and $\mathcal{A}_{ij}$ is the target i to measurement j assignment probability, as described earlier.

Method	Rcll↑	Prcn↑	MT↑	ML↓	FP↓	FN↓	IDs↓	FM↓	MOTA↑	MOTP↑
Kalman-HA	28.5	79.0	32	334	3,031	28,520	685	837	19.2	69.9
Kalman-HA2*	28.3	83.4	39	354	2,245	28,626	105	342	22.4	69.4
JPDA$_m$*	30.6	81.7	38	348	2,728	27,707	109	380	23.5	69.0
RNN_HA	37.8	75.2	50	267	4,984	24,832	518	963	24.0	68.7
RNN_LSTM	37.1	73.5	50	260	5,327	25,094	572	983	22.3	69.0

Table 1: Tracking results on the MOTChallenge training dataset. *Denotes offline post-processing.

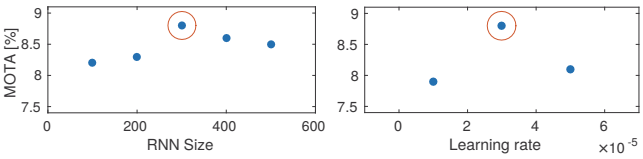

Figure 5: Influence of two exemplar hyper-parameters on the overall performance on the MOTChallenge benchmark, measured by MOTA. The optimal parameter is marked with a red circle. Note that this graph shows the performance of our prediction/update RNN block for only one target ($N = 1$), which explains the relatively low MOTA.

Training Data

It is well known that deep architectures require vast amounts of training data to avoid overfitting the model. Huge labelled datasets like ImageNET (Russakovsky et al. 2014) or Microsoft COCO (Lin et al. 2014) have enabled deep learning methods to unfold their potential on tasks like image classification or pixel labelling. Unfortunately, mainly due to the very tedious and time-consuming task of video annotation, only very limited amount of labelled data for pedestrian tracking is publicly available today. We therefore resort to synthetic generation by sampling from a simple generative trajectory model learned from real data. To that end, we first learn a trajectory model from each training sequence. For simplicity, we only estimate the mean and the variance of two features: the start location x_1 and the average velocity $\bar{v}$ from all annotated trajectories in that sequence. For each training sample we then generate up to N tracks by sampling from a normal distribution with the learned parameters. Note that this simplistic approach enables easy generation of realistic data, but does not accomodate any observations.

Implementation Details

We implemented our framework in Lua and Torch7. Both our entire code base as well as pre-trained models are publicly available.[2] Finding correct hyper-parameters for deep architectures still remains a non-trivial task (Greff et al. 2015). In this section we will point out some of the most important parameters and implementation details. We follow some of the best practices found in the literature (Greff et al. 2015; Karpathy, Johnson, and Li 2015), such as setting the initial weights for the forget gates higher (1 in our case), and also employ a standard grid search to find the best setting for the present task.

[2]https://bitbucket.org/amilan/rnntracking

Network size. The RNN for state estimation and track management is trained with one layer and 300 hidden units. The data association is a more complex task, requiring more representation power. To that end, the LSTM module employed to learn the data association consists of two layers and 500 hidden units.

Optimisation. We use the RMSprop (Tieleman and Hinton 2012) to minimise the loss. The learning rate is set initially to 0.0003 and is decreased by 5% every $20\,000$ iterations. We set the maximum number of iterations to $200\,000$, which is enough to reach convergence. The training of both modules takes approximately 30 hours on a CPU. With a more accurate implementation and the use of GPUs we believe that training can be sped up significantly.

Data. The RNN is trained with approximately 100K 20-frame long sequences. The data is divided into mini-batches of 10 samples per batch and normalised to the range $[-0.5, 0.5]$, w.r.t. the image dimensions. We experimented with the more popular zero-mean and unit-variance data normalisation but found that the fixed one based on the image size yields superior performance.

Experiments

To demonstrate the functionality of our approach, we first perform experiments on simulated data. Fig. 4 shows an example of the tracking results on synthetic data. Here, five targets with random birth and death times are generated in a rather cluttered environment. The initiation / termination indicators are illustrated in the bottom row.

We further test our approach on real-world data, using the MOTChallenge 2015 benchmark (Leal-Taixé et al. 2015). This pedestrian tracking dataset is a collection of 22 video sequences (11/11 for training and testing, respectively), with a relatively high variation in target motion, camera motion, viewing angle and person density. The evaluation is performed on a server using unpublished ground truth. Next to precision and recall, we show the number of mostly tracked ($>80\%$ recovered) and mostly lost ($<20\%$ recovered) trajectories (Li, Huang, and Nevatia 2009), the number of false positive (FP), false negative (FN) targets, identity swaps (IDs) and track fragmentations (FM). MOTA and MOTP are the widely used CLEAR metrics (Bernardin and Stiefelhagen 2008) and summarise the tracking accuracy and precision, respectively. Arrows next to each metric indicate weather higher (↑) or lower (↓) values are better.

Method	MOTA↑	MOTP↑	MT%↑	ML%↓	FP↓	FN↓	IDs↓	FM↓	FPS↑
MDP (Xiang et al. 2015)	30.3%	71.3%	13.0	38.4	9,717	32,422	680	1,500	1.1
SCEA (Hong Yoon et al. 2016)	29.1%	71.7%	8.9	47.3	6,060	36,912	604	1,182	6.8
JPDA$_m$* (Rezatofighi et al. 2015)	23.8%	68.2%	5.0	58.1	6,373	40,084	365	869	32.6
TC_ODAL (Bae and Yoon 2014)	15.1%	70.5%	3.2	55.8	12,970	38,538	637	1,716	1.7
RNN_LSTM (ours)	19.0%	71.0%	5.5	45.6	11,578	36,706	1,490	2,081	165.2

Table 2: Tracking results on the MOTChallenge test dataset. *Denotes an offline (or delayed) method.

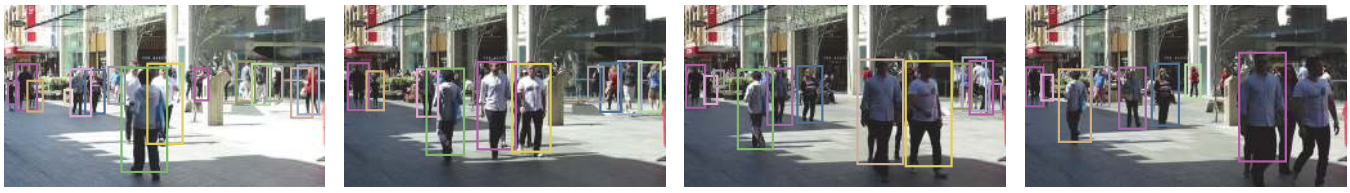

Figure 6: Our RNN tracking results on the MOTChallenge sequence ADL-Rundle-3. Frames 104, 149, 203, and 235 are shown. The colour of each bounding box indicates the person identity.

Baseline comparison. We first compare the proposed approach to three baselines. The results on the training set are reported in Tab. 1. The first baseline (Kalman-HA) employs a combination of a Kalman filter with bipartite matching solved via the Hungarian algorithm. Tracks are initiated at each unassigned measurement and terminated as soon as a measurement is missed. This baseline is the only one that fully fulfils the online state estimation without any heuristics, time delay or post-processing. The second baseline (Kalman-HA2) uses the same tracking and data association approach, but employs a set of heuristics to remove false tracks in an additional post-processing step. Finally, JPDA$_m$ is the full joint probabilistic data association approach, recently proposed in (Rezatofighi et al. 2015), including post-processing. We show the results of two variants of our method. One with learned motion model and Hungarian data association, and one in which both components were learned from data using RNNs and LSTMs. Both networks were trained separately. Our learned model performs favourably compared to the purely online solution (Kalman-HA) and is even able to keep up with similar approaches but without any heuristics or delayed output. We believe that the results can be improved further by learning a more sophisticated data association technique, such as JPDA, as proposed by Milan *et al*. (2017), or by introducing a slight time delay to increase robustness.

Benchmark results. Next, we show our results on the benchmark test set in Tab. 2 next to three *online* methods. The current leaderboard lists over 70 different trackers, with the top ones reaching over 50% MOTA. Even though the evaluation is performed by the benchmark organisers, there are still considerable differences between various submissions, that are worth pointing out. First, all top-ranked trackers use their own set of detections. While a better detector typically improves the tracking result, the direct comparison of the tracking method becomes rather meaningless. Therefore, we prefer to use the provided detections to guarantee a fair setting. Second, most methods perform so-called *offline* tracking, *i.e*. the solution is inferred either using the entire video sequence, or by peeking a few frames into the future, thus returning the tracking solution with a certain time delay. This is in contrast to our method, which aims to strictly compute and fix the solution with each incoming frame, before moving to the next one. Finally, it is important to note that many current methods use target appearance or other image features like optic flow (Choi 2015) to improve the data association. Our method does not utilise any visual features and solely relies on geometric locations provided by the detector. We acknowledge the usefulness of such features for pedestrian tracking, but these are often not available in other application, such as *e.g*. cell or animal tracking. We therefore refrain from including them at this point.

Overall, our approach does not quite reach the top accuracy in pedestrian online tracking (Xiang, Alahi, and Savarese 2015), but is two orders of magnitude faster. Fig. 6 shows some example frames from the test set.

Discussion and Future Work

We presented an approach to address the challenging problem of data association and trajectory estimation within a neural network setting. To the best of our knowledge, this is the first approach that employs recurrent neural networks to address online multi-target tracking. We showed that an RNN-based approach can be utilised to learn complex motion models in realistic environments. The second, somewhat surprising finding is that an LSTM network is able to learn one-to-one assignment, which is a non-trivial task for such an architecture. We firmly believe that, by incorporating appearance and by learning a more robust association strategy, the results can be improved significantly.

Acknowledgments. This work was supported by ARC Linkage Project LP130100154, ARC Laureate Fellowship FL130100102 and the ARC Centre of Excellence for Robotic Vision CE140100016.

References

Bae, S.-H., and Yoon, K.-J. 2014. Robust online multi-object tracking based on tracklet confidence and online discriminative appearance learning. In *CVPR*.

Bar-Shalom, Y., and Fortmann, T. E. 1988. *Tracking and Data Association*. Academic Press.

Berclaz, J.; Fleuret, F.; Türetken, E.; and Fua, P. 2011. Multiple object tracking using k-shortest paths optimization. *IEEE T. Pattern Anal. Mach. Intell.* 33(9):1806–1819.

Bernardin, K., and Stiefelhagen, R. 2008. Evaluating multiple object tracking performance: The CLEAR MOT metrics. *Image and Video Processing* 2008(1):1–10.

Butt, A. A., and Collins, R. T. 2013. Multi-target tracking by Lagrangian relaxation to min-cost network flow. In *CVPR*.

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

Doucet, A.; Godsill, S.; and Andrieu, C. 2000. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing* 10(3):197–208.

Eigen, D., and Fergus, R. 2015. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*.

Fortmann, T. E.; Bar-Shalom, Y.; and Scheffe, M. 1980. Multi-target tracking using joint probabilistic data association. In *IEEE Conference on Decision and Control*.

Goller, C., and Küchler, A. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *ICNN*, 347352. IEEE.

Greff, K.; Srivastava, R. K.; Koutnk, J.; Steunebrink, B. R.; and Schmidhuber, J. 2015. LSTM: A search space odyssey. *arXiv:1503.04069 [cs]*. arXiv: 1503.04069.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Comput.* 9(8):17351780.

Hong Yoon, J.; Lee, C.-R.; Yang, M.-H.; and Yoon, K.-J. 2016. Online multi-object tracking via structural constraint event aggregation. In *CVPR*.

Ivakhnenko, A. G., and Lapa, Valentin, G. 1966. Cybernetic predicting devices.

Jiang, H.; Fels, S.; and Little, J. J. 2007. A linear programming approach for multiple object tracking. In *CVPR*.

Kalman, R. E. 1960. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering* 82(Series D):35–45.

Karpathy, A.; Johnson, J.; and Li, F.-F. 2015. Visualizing and understanding recurrent networks. *arXiv:1506.02078*.

Kim, C.; Li, F.; Ciptadi, A.; and Rehg, J. M. 2015. Multiple hypothesis tracking revisited: Blending in modern appearance model. In *ICCV*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*, 1097–1105.

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

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Leibe, B.; Schindler, K.; and Van Gool, L. 2007. Coupled detection and trajectory estimation for multi-object tracking. In *ICCV*.

Li, Y.; Huang, C.; and Nevatia, R. 2009. Learning to associate: Hybridboosted multi-target tracker for crowded scene. In *CVPR*.

Lin, T.-Y.; Maire, M.; Belongie, S.; Bourdev, L.; Girshick, R.; Hays, J.; Perona, P.; Ramanan, D.; Zitnick, C. L.; and Dollár, P. 2014. Microsoft COCO: Common objects in context. *arXiv:1405.0312 [cs]*.

Milan, A.; Rezatofighi, S. H.; Garg, R.; Dick, A.; and Reid, I. 2017. Data-driven approximations to NP-hard problems. In *AAAI*.

Milan, A.; Roth, S.; and Schindler, K. 2014. Continuous energy minimization for multitarget tracking. *IEEE T. Pattern Anal. Mach. Intell.* 36(1):58–72.

Milan, A.; Schindler, K.; and Roth, S. 2013. Detection- and trajectory-level exclusion in multiple object tracking. In *CVPR*.

Ondruska, P., and Posner, I. 2016. Deep tracking: Seeing beyond seeing using recurrent neural networks. In *AAAI*.

Ondruska, P.; Dequaire, J.; Zeng Wang, D.; and Posner, I. 2016. End-to-end tracking and semantic segmentation using recurrent neural networks. In *RSS Workshop on Limits and Potentials of Deep Learning in Robotics*.

Reid, D. B. 1979. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control* 24(6):843–854.

Rezatofighi, H. S.; Milan, A.; Zhang, Z.; Shi, Q.; Dick, A.; and Reid, I. 2015. Joint probabilistic data association revisited. In *ICCV*.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2014. *ImageNet Large Scale Visual Recognition Challenge*.

Tang, S.; Andres, B.; Andriluka, M.; and Schiele, B. 2016. Multi-person tracking by multicuts and deep matching. In *ECCV Workshop on Benchmarking Multi-Target Tracking*.

Tieleman, T., and Hinton, G. 2012. Rmsprop: Divide the gradient by a running average of its recent magnitude. Coursera: Neural networks for machine learning.

Vinyals, O.; Toshev, A.; Bengio, S.; and Erhan, D. 2015. Show and tell: A neural image caption generator. In *CVPR*.

Wang, T.; Wu, D.; Coates, A.; and Ng, A. 2012. End-to-end text recognition with convolutional neural networks. In *2012 21st International Conference on Pattern Recognition (ICPR)*, 3304–3308.

Xiang, Y.; Alahi, A.; and Savarese, S. 2015. Learning to track: Online multi-object tracking by decision making. In *ICCV*, 4705–4713.

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