Multiple Object Tracking using RNNs

Khizer Amin

A thesis submitted in partial fulfillment for the
degree of Master in Artificial Intelligence

in the

Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

Supervisor: Prof. Thomas Brox

December 2017

Contents

Chapter 1

Introduction

1.1 Problem Statement

Object Tracking is one of the primal tasks in computer vision. To address this problem, huge amount of work has been published to date, hence we observe significant progress in the domain of single object tracking research. On the other hand, object tracking in the multiple object domain is still an open problem in the real-world settings.

Object tracking means to associate data over a period of time. Typical challenges for object tracking algorithms include, occlusions, varying illumination of the scene due to light source or motion of the object, object viewpoint changes make it difficult to use a static model for the object, and background clutter exacerbates the confusion in the search for the correct target. On top of these typical problems for single-object trackers, multi-object tracking introduces further challenges including, different types of occlusions (object-object, object-background etc.), illumination variations due to shadowing effects and motion of other objects in the scene, and very importantly the tracking latency increases (atleast linearly) with respect to number of objects to track in the scene.

Recently, deep learning have shown a great potential and produced some excellent results towards soving a wide spectrum of computer vision tasks. Single-object tracking is one of those areas of research which has benefited from the power of deep learning. However, these improvements have not yet been translated to the task of multi-object tracking. In this work, we investigate the mutli-object tracking task based on Recurrent Neural Networks approach proposed by Milan et al. [1].

1.2 State of the art and related work

1.2.1 Multi-target tracking

There are multiple papers on Multi-Object or Multi-Target tracking. The JPDA uses a tracker based on the joint probabilistic data associations, it is a very interesting approach, but is more appropriate to signals processing rather then image processing. JPDA-m [2] is a more refined approach by the same researchers where they tried to add more computational power to the system with image data using some additional features, they are solving the non-trivial problem of data association using an integer linear problem. With

the recent advancement in computing power the quality of image data, we have seen multiple new projects being developed in recent years, one example is the work by (A. Sadeghian, A. Alahi, S. Savarese) [3], the interesting thing to note her is that they used RNNs/LSTMs to individual features of the pedestrian, i.e. they call it a cue(their appearance, motion and inter-relations), the results are impressive in MOTChallenge 2015.

1.2.2 Deep leaning in tracking

Recently, deep learning has revolutionized how we detect and track objects. In the object detection research, for instance, the methodologies that provide the best performance are dominated by Deep Neural Networks (DNNs). Similarly, the top performing tracking method for the single-object tracking benchmark VOT Challenge, is based on deep convolutional network. Unlike popularity of DNNs for single-object tracking, the multi-object tracking research is ruled by the approaches based on offline partitioning of the graphical models. Naturally, such trackers work as a post processing step once the meta-information regarding objects in all of the video frames is available. Due to this non-causal behavior (all past, current, and future frames required), such tracking methods cannot be used in real-time for online object tracking.

Deep Learning has influenced almost all the computer vision research, in case of tracking this is a little bit different as here the data is not as abundant as in we have in normal detection and classification cases [4], i.e. for the likes of imageNET which is entirely a labelled data etc. Convolutional neural networks(CNNs) works well in an ANN setting but they are not well suited for a sequential data, lets say text, or images with sequence of frames. There CNNs gets complicated and also the outputs are not as required. On the other hand RNNs works back and fourth in neural network and ofcourse the fact that can retain and preserve memory in the form of a hidden state.

The work that we are re implementing in this thesis is inspired by the successful application RNNs on language and text modeling, i.e. [5], here in this approach the idea is to implement an RNN in such a way that the system should not only predict the next letter or word at a time. When it comes to images the we encounter a multi-dimensional state space, which is not the case with textual data, but applying a multi-dimensions to text data would mean that the whole context of the text should be contained or memorized and given at the input.

The task of data association demands that the context between different frames and their targets or objects should be contained very well in order to create smooth trajectories for individual targets. LSTMs can be really helpful in this regard, because the LSTMs memory model is strong and the network itself has a built-in support for preserving the context, they will discussed further with more detail. In the paper [3] discussed earlier LSTMs are used at every level of matching the individual targets, i.e. they used three LSTM cells for each cue(features).

1.3 Motivation and Scope

The scope of this thesis is to develop an online multiple-object tracking application which uses recurrent neural networks. The RNNs are special purpose neural networks which supports preserving memory and states for an object. In object tracking case this is the state of object in a specific frame. The job of recurrent nets is to preserve the current state(position) of an object and to compare it with its previous state. For the task of data association an LSTM(Long-short-term-memory) cell is used.

The main motivation for the thesis is to re-design and re-implement a object tracking application in Tensorflow, which is deep learning library developed by Google. This gives the author an opportunity to learn and understand the concepts about tracking and more importantly multiple object tracking in depth. Also, the research helped in understanding the differences between normal detection and the tracking process.

As a motivation the following figure gives an insight on the tracking process, here we just show results using the ground truth bounding boxes, as can be seen below:

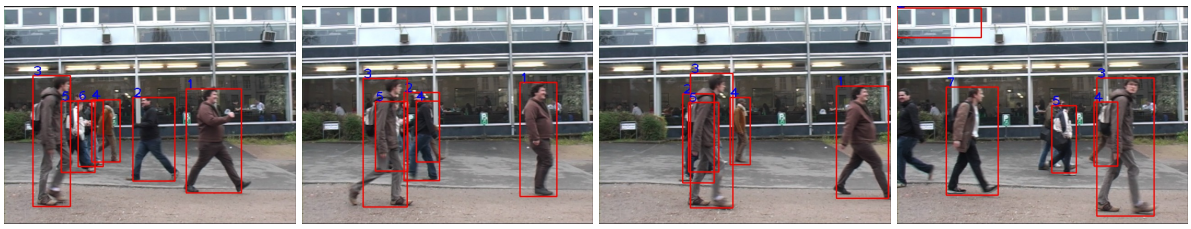

Figure 1: A scene from TUD dataset showing each pedestrians with their IDs assigned to them and with their bounding boxes. As a person disappear from the frame, his trajectory is also removed, i.e. ids are removed. As a new person appears in the scene his/her is given a new id .

1.4 Challenges in tracking

1.4.1 Occlusion

In object tracking occlusions can subvert the functionality of a tracking application. It has emerged as one of the most important topic in object tracking. Occlusion happens when multiple targets [6] overlaps each other and even for good tracker it sometimes become difficult to differentiate and/or to assign proper ids to the targets. There are number of approaches to handle occlusion, here are some of those approaches [7] and [8].

1.4.2 View points

One of the big challenge in object tracking, is to track objects and targets with a moving camera and different view points for individual target. Change in viewpoints cause changes in object's appearance. In this paper [9] the authors demonstrate a method that performs the tracking in 3D with different view points.

1.4.3 Varying number of targets

Lastly, appearance and disappearance of a target in the scene, i.e. We call it birth/death of the targets. This problem can pose serious challenge to the tracker. The original paper that we are re-implementing in this thesis, has proposed an approach to counter this challenge. Using effective data association can solve this issue and as discussed later, LSTMs are used in this regard where they generate an association matrix which when given to the network as an input provides an existence probability matrix which observes the varying numbers of targets.

1.5 Thesis Overview

The thesis is split into different chapters. Each chapter will ad- dress specific aspect/s of the project. The summary of each chapter's content is as follows:

Chapter 1 - Introduction

This chapter gives an overview of the thesis. It also explains the aims and motivation of the project.

Chapter 2 - Technical Background

This chapter covers the theory and the fundamental concepts behind recurrent neural network.

Chapter 3 - Multi-Target Tracking

This chapter describes the implementation details involved behind the development for this MOT framework. It also provides a detail analysis for training of the RNNs and LSTMs using a training data based on video sequences.
This chapter also discuss in detail the methodologies and the baseline architecture used in the creation of a tracking application based on RNNs. The challenging task of data association is also discussed in this chapter.

Chapter 4 - Experiment and Results

This chapter presents the results of the approach and also compares the approach of this thesis results with the MOT benchmark challenges and their obtained results.

Chapter 6 - Conclusion and Future work

This chapter concludes the thesis and presents the future direction for extending the work.

Chapter 2

Technical Background

2.1 Recurrent Neural Networks

RNN is a type of neural network which runs on a sequential data. It basically makes predictions based on the current input and the previous state of the networks. In RNNs, the key element that stores the values or condition of the previous state is called the RNN hidden Layer. This hidden state works as a control unit for the RNN. So in order to make good predictions the RNN hidden state makes sure a smooth transition between previous and current states, by making sure the difference between the current and previous state of the network is not large. The illustration in Figure 2.1 shows the function of a typical RNN. In the following we describe at the very detail of what is actually happening in this kind of neural networks.

In the Figure below, we observe y_t is just a function of hidden state of the RNN i.e. h_t, multiplied by the weight matrix W_y plus the bias term b_y (overall a linear transformation). Here h_t represents the value of the hidden state at time t. Mathematically we write it as,

$$y_t = W_y h_t + b_y \tag{2.1}$$

Whereas, the value of the hidden state h_t is estimated as,

$$h_t = g_h\Big(W_I x_t + W_R . h_{t-1} + b_h\Big) \tag{2.2}$$

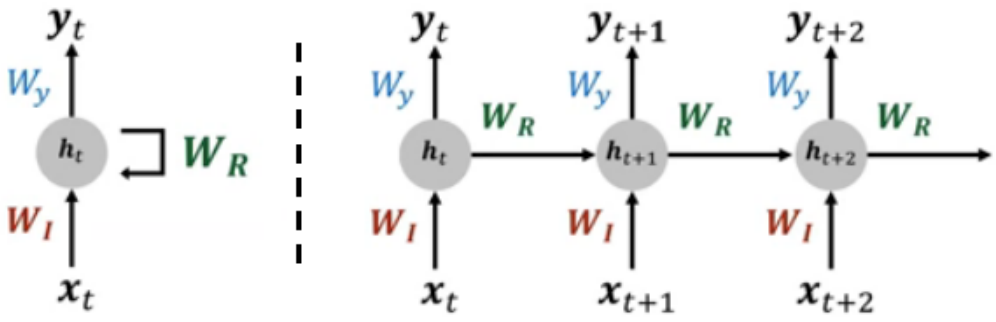

Figure 2.1: *Left*: A typical Recurrent Neural Network (RNN). *Right*: Unrolled RNN in time.

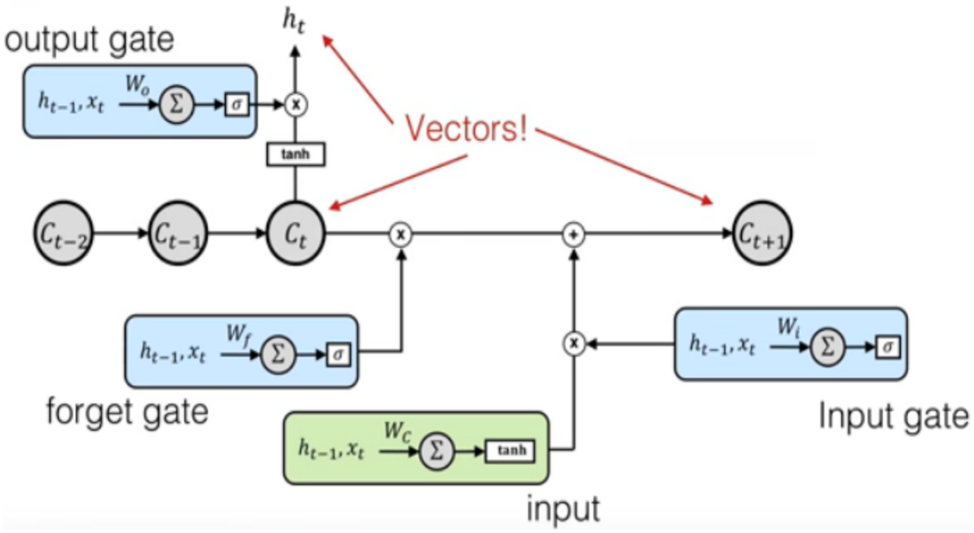

Figure 2.2: A basic LSTM cell.

Here, W_I represents the weight matrix for linear transformation of input variable x_t at time t, and W_R carries out the transformation of hidden state of previous time step $t-1$. Weighted information from both current input x_t and hidden state of previous time steps h_{t-1} are added together with the bias term b_h, and then passed through a non-linear activation function g_h to estimate the value of the new hidden state of the RNN at time t i.e., h_t.

For better understanding, we depict an unrolled version of RNN cell in Figure 2.1, which shows the function of an RNN cell as a feed-forward network. One thing to note here is that the activity of the unit h_t depends not only on the input x_t but also on the activity at the previous timestep h_{t-1}. So the activity at the memory unit at time t is passed on to the unit activity at time $t+1$. Note that, the weights W_R are shared accross the layers in time.

2.2 Long Short-Term Memory RNNs

Long short-term memory (LSTM) is a special purpose RNN. Typically, an LSTM cell is comprised of three gates, that basically controls the flow of the information. These gates are,

- Forget gate: A memory forgetting mechanism

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{2.3}$$

- Input gate: A memory saving mechanism

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2.4}$$

- Output gate: A memory focusing mechanism, which saves the long-term memory into workable memory.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.5}$$

In the above, i_t represents the input gate with learnable parameters W_i and bias b_i, f_t represent the forget gate with learnable parameter W_f and bias b_f, and similarly, o_t represent the forget gate with learnable parameter W_o and bias b_o. x_t represents the input to the current RNN layer (where LSTM cell is located) at time t, and h_{t-1} represents the previous state of this RNN layer.

From the mathematical derivation of the gates, we notice that each gate performs a function of a typical fully-connected layer of a neural network. These gates take the hidden state h_{t-1} and the current input x_t as inputs for estimating their appropriate action. Figure 2.2 shows the basic structure of an LSTM cell. We notice that each gate has a small memory block. Combined, these gates form a strong memory unit which is helpful in solving the non-trivial tasks of sequential processing and data associations in temporal domain.

Let us look at the mathematical derivations of the data flow in a typical LSTM from Figure 2.2. $\tilde{C}_t$ represents the current input to the LSTM cell at time t, given by.

$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_c) \tag{2.6}$$

Here W_C represents the weight matrix which computes the new input for the LSTM memory cell based on x_t and h_{t-1}

The output of the LSTM memory cell is produced by applying non-linearity to the current memory state C_t and then multiplying with the status of the output gate.

$$h_t = o_t \tanh(C_t) \tag{2.7}$$

The new state of the LSTM memory cell for timestep $t + 1$ is a weighted combination of the current state C_t and the current input to the LSTM memory cell $\tilde{C}_t$. Mathematically this means,

$$C_{t+1} = f_t C_t + i_t \tilde{C}_t \tag{2.8}$$

If we look more closely at this equation, we find that activity of the cell at time $t + 1$ (the next timestep) is equal to the memory or the amount memory we want to forget from the previous time step, i.e. C_t. Plus the input at the new timestep which is multiplied by how much we want to accept from this new input that we get at timestep t. So in practice we notice that new memory of the LSTM cell C_{t+1} is basically a weighted sum between the amount of information that the two gates, i.e. input and forget, allow to flow.

In this master thesis project, LSTMs are specifically investigated for the task of data association which will discuss in the next chapter.

2.3 Bayesian Filtering

Bayes filtering is a method used to estimate the posterior probabilities of the systems output given the current evidence system has collected in terms of the current system state and the measurements upto current timestep. In other words, Bayes filters allow to continuously update the most likely state of the system, based on the most recently acquired sensor data. This is a recursive algorithm, consisting of two

parts: prediction and update. If the variables are linear and normally distributed the Bayes filter becomes equal to the Kalman filter.

One of task of this project is to make estimations and pruning of ground truth data, i.e. the true state of the object with the detected measurements which are noisy and unclear. In the similar bayesian fashion, the probabilities in this tracking project are evaluated in two steps, i.e. a prediction stage which predicts the state of the next frame x^*_{t+1} based on current state x_t and the hidden state h_t.

Chapter 3

Multi-Target Tracking using RNNs

Multiple target tracking in unconstrained environments is quite challenging. Significants advancements have been made but the results on state-of-the-art performance benchmarks such as MOTChallenge (Leal-Taixe et al. 2015), are still not able to come close to the performance we achieve on single target tracking benchmarks such as VOT. Multi target tracking essentially means locating all targets of interest in a video sequence without losing their identities over time. Its not obvious, how to represent the model for tracking data which handles arbitrary videos with different viewpoints, motion dynamics, level and type of occlusions, camera motion and varying number of targets.

Typical approaches for multi-target tracking employ so called tracking-by-detection strategy. To that end, first the object detections are obtained in the individual video frames and then they are associated over time using various techniques e.g., graph-cuts. There has been surprisingly little work on multi-target tracking using recent depp learning based approaches. This is mainly due to the fact that the amount of labeled data that it is available for multi-target settings is limited, that essentially inhibits the deep learning models with huge number of parameters to train effectively.

In this thesis work, we follow the approach of Milan et al. [1] and develop our own multi-target tracking framework in Keras deep learning library, with some model simplifications. Using Bayesian formulation we decompose the tracking problem in two stages, i.e., (i) state prediction and (ii) state update. This way we can isolate the tracking algorithm which makes it easier to implement and debug the system. It also enables one to train each individual block separately

Following the notations from [1], in our work we represent the objects or targets as bounding boxes with four features per object (x, y, w, h). Lets dissect each term used in the above the architecture. At input we have x_t^i which represents an ith target at time t. h^t is the hidden state of the RNN network at time t. z_{t+1} is a vector for measurements, i.e. detection's at time $t+1$. The A^{t+1} is an assignment probability matrix and it represents for each target the probabilities for assigning the measurements, we get this matrix from the LSTM stage which will be discussed later. The ϵ is a vector which calculates the existence of a target at time t in a frame.

At the output we have at first x_{t+1}^* the predicted targets at time $t+1$ along with its hidden state h_{t+1}. Once we get the predicted values we can use them to update the state of the system at the update stage,

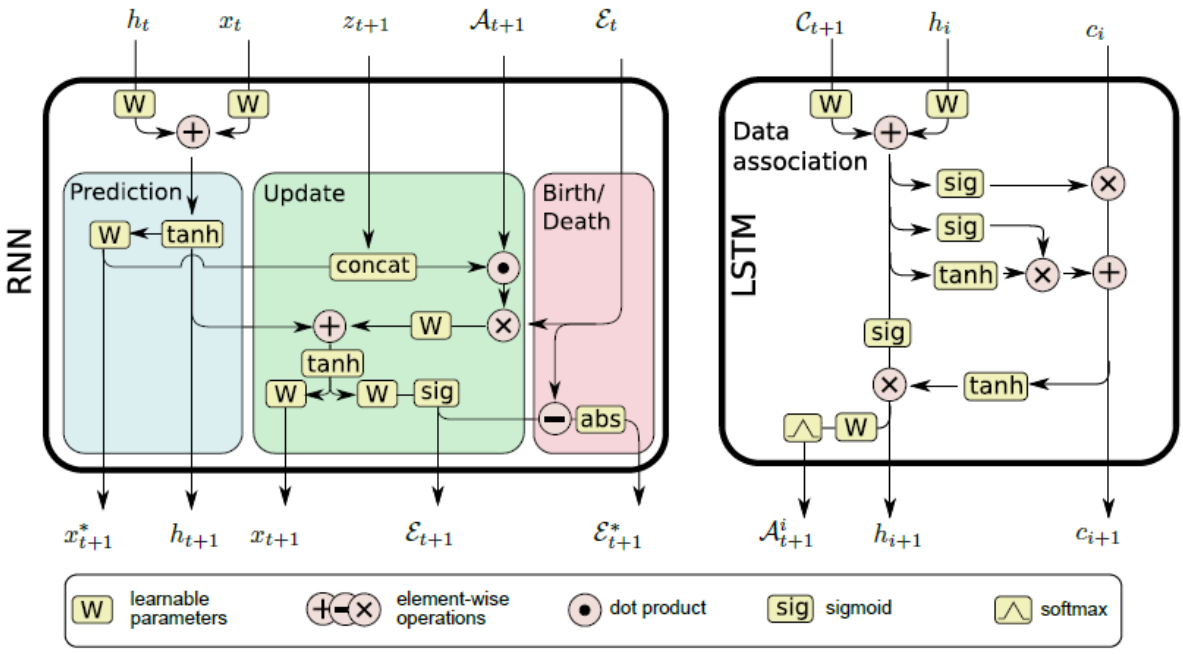

Figure 3.1: *Left*: RNN architecture for 3 operations, state prediction, state update and Birth/Death estimation for targets. *Right*: An architecture for LSTM for the task of data association. The picture is reproduced from the original work of Milan et al. [1]

i.e. x_{t+1}. For this purpose we also require the target measurements and the assignment matrix which is the association matrix provided after the LSTM process. Along with the update target we also get ϵ_{t+1} which is necessary to get an absolute value of the existence of individual target, i.e. their appearance and disappearance in a frame. We call it ϵ^*_{t+1} at time $t+1$.

3.1 RNN stage

Based on bayesian filtering concept, the problem of tracking is decomposed into two major blocks: state prediction and update. Figure 3.1 depicts the baseline architecture for this project on RNN-based multi-target tracking. Later in the following sections we will also discuss our modifications.

Overall RNN model shown in the figure has two main stages each of them contains their states and operations. In the task of predicting the motion of the target for the next timestep, the RNN system takes five inputs including the hidden state at time t and produces five outputs again including the hidden state for the next time step $t+1$ Lets look at the RNN stage first, the primary objective for this stage is based on 3 main operations.

a. Prediction process: This operation just takes the original data, i.e. the ground truth bounding boxes and predicts the next state in the target motion. Let us drive the equations from the above figure for the prediction layer.

$$h_{t+1} = \tanh(W_I.x_t + W_R.h_t) \tag{3.1}$$

$$x^* = W.h_{t+1} \tag{3.2}$$

b. Update process: This operation takes into consideration the detected data that we already have. We call it measurements this data shall be passed to this stage, a comparison is made between the predicted bounding boxes and the measurements to get an updated state for the target, i.e. x_{t+1}. The equations for this layer are as follows:

$$C_(x.a) = [x^*_{t+1}, z_{t+1}] \tag{3.3}$$

$$temp = \tanh(h_{t+1} + W.(\epsilon_t.[C_(x.a), A_{t+1}]) \tag{3.4}$$

$$x_{t+1} = W.temp \tag{3.5}$$

$$\epsilon_{t+1} = \sigma(W.temp) \tag{3.6}$$

The C_xa is result of the concatenation between measurements and detections. Argument 'temp' is a variable that just takes the results after the some bit-wise operations as we can see above. After that we get the updated state of the target by taking dot product between 'temp' and a weight matrix 'W'. We also need to take another dot product between the two values in order to get the ϵ_{t+1}, the existance matrix at 't+1'.

c. Birth/Death process In this operation the appearance and disappearance of the targets is measured. A so called existence probability vector which calculates how likely the object has a real trajectory. At the output a vector of absolute difference is obtained, i.e. :

$$E^*_{t+1} = \|E_{t+1} - E_t\| \tag{3.7}$$

Let's just discuss the flow in more depth. The network starts at the RNNs prediction stage which takes the input x_t and h_t at time 't', this data is the ground truth bounding boxes which are passed to the network in order to get the predicted bounding boxes x_{t+1} at time $t + 1$. In the update stage of the RNN the predictions and all the measurements are concatenated, i.e. $[x^*_{t+1}, z_{t+1}]$ they also require the assignments probabilities A. Along with the new updated state, the existence probabilities ϵ_{t+1} are also generated.

3.1.1 Data Association

To update the predictions of the first step of the recurrent neural network, we require evidence from the detection measurements. To able to use these measurements, we need to associate the detection bounding boxes to the predicted bounding boxes coming from the first prediction step. The association here means finding the similarity of the bounding boxes. In the following, we describe the baseline LSTM-based approach followed by a simpler solution based on Intersection-over-Union overlap between bounding boxes.

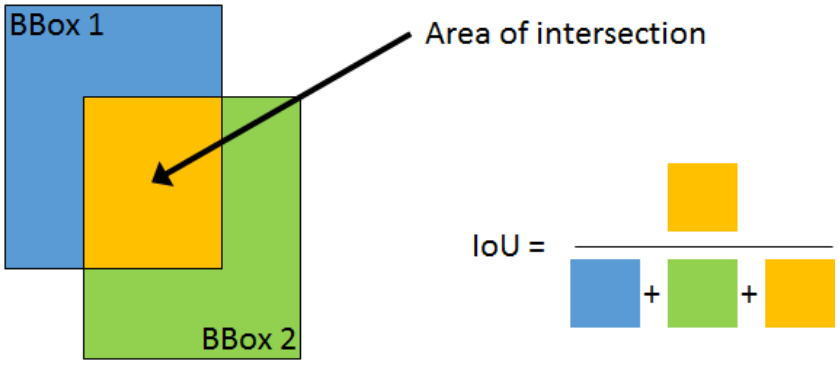

Figure 3.2: Intersection-over-Union (IoU) overlap between two bounding boxes.

3.2 LSTM-based Data Association

The task of data association is addressed in this section. LSTM has a strong memory unit, in our case it predicts the association probabilities for the targets in the next frame and provides at the output a vector of associations between all the predicted targets x^*_{t+1} and the detection measurements z_{t+1} for the video frame $t+1$. Let us look at the derivations for the LSTM stage. Assuming, inputs C_{t+1} as the distance matrix between predictions x^*_{t+1} and measurements z_{t+1}, and h_t as the previous hidden state of the LSTM block, the LSTM stage is designed to predict the data association probablilities between the predictions and the measurements for the frame $t+1$. The output is given in terms of an association matrix A_{t+1}.

The pairwise euclidean distance matrix C_{t+1} is given by the euclidean norm of the difference of coordinates of the predicted bounding boxes x^*_{t+1} and the measurement detection bounding boxes z_{t+1}. Single entry in the matrix is given by,

$$C = \|x^i - z^j\| \tag{3.8}$$

Based on this distance matrix and its internal hidden state, the LSTM stage produces a matrix of probabilities A_{t+1} which associates one target with all the measurements. This association matrix is then passed to the update stage of the RNN, which then is able to estimate the updated state of the target, i.e. x_{t+1}. This completes the tracking pipeline proposed by Milan et al. [1].

3.3 Greedy IoU-based Data Association

The model described in the work of Milan et al. [1], as described above in detail, is quite complex to train and easily overfits if the hyper-parameters are not carefully tuned. In particular the implementation of the LSTM stage for data association is not straight-forward. In this thesis work, we propose to simplify the model by removing the LSTM based data association component with a standard greedy approach based on just intresection-over-union (IoU) overlap.

Intersection-over-Union (IoU) tells the similarity of the two bounding boxes in terms of their overlap with each other. Higher the IoU value means more similar the two bounding boxes are to each other. IoU

between two given bounding boxes is shown in Figure 3.2 and also defined below,

$$IoU = \frac{Intersection\ Area\ of\ Two\ Bounding\ Boxes}{Union\ of\ Areas\ of\ Both\ Bounding\ Boxes} \tag{3.9}$$

In our approach, we compute the intresection-over-union (IoU) overlap for all predicted targets and detection measurement pairs for the current video frame. This directly gives us the NxM+1 matrix A_{t+1} between N targets x^*_{t+1} and M detection measurements z_{t+1} for the video frame $t+1$. For consistency with the baseline approach we normalize the rows of the association matrix to make the sum of scores for each target equal to 1.0. This gives the IoU score matrix a probabilistic meaning. Though, we also experimented with just considering the maximum overlap between the predicted targets with the measurements for data association. We implemented this by just assigning 1.0 to the pair with highest IoU overlap, in the association matrix A_{t+1} of the predicted targets and the detection measurements.

The resulting association matrix is then directly applicable in the RNN approach described above, replacing completely the complex LSTM component in Figure 3.1. We show in our experiments that this simple approach gives quite reasonable results. Such a simple and greedy association of prediction and measurement targets based on IoU overlap is quite common in the literature.

Chapter 4

Experiment and Results

In this chapter, we evaluate our approach and compare it to the state-of-the-art results in the literature for multi-target tracking. We start by defining the training data, and implementation details. Later we discuss the evaluation metrics used and the results we obtained.

4.1 Training Data

As we know that for any deep learning method a huge amount of training data is required, since the idea of deep learning for computer vision is to learn representations of the raw data to simplify the vision tasks. For example, the computer vision community has benefited a lot from the datasets such as *ImageNet*, which is a huge dataset of images with thousands of image-level class labels.

However, in our case we require the training data with sequence of images to learn to track objects over time, which also requires a huge amount of annotated video data which is hard to get and also time consuming. In this work, we focus on the bounding box tracking without using the image features, however the approach can be extended as also argued by [1]. To benchmark our approach in comparison to the state-of-the-art in the field, we use real world video data from the MOTChallenge 2015 benchmark [10]. It is a dataset for pedestrian tracking with a total of 22 video sequences. The dataset also includes the detection measurements for the pedestrian bounding boxes together with their ground-truth annotations in all the 22 video sequences.

4.2 Implementation

The task is to re-design and re-implement the whole prosposed solution from [1] using Keras which is a well known library able to run on top of Google's deep learning library Tensorflow, and simplify the methodology where required. To that end, we used functional API provided by Keras which gives us the support to pass multiple inputs to the network and more importantly to get multiple outputs from each layer. The code and the datasets are publicly available. The orignal paper is inspired by Andrea Karpathy's approach on using RNNs on the textual data. [1] [2].

[1] karpathy.github.io/2015/05/21/rnn-effectiveness/
[2] https://github.com/karpathy/char-rnn

Method	MOTA (%)	MOTP (%)
AMIR15(SadeghianAS17)	37.6	71.7
MDP(xiang_iccv15)	30.3	71.3
JPDMm(Rezatofighi2016)	23.8	68.2
RNN_LSTM(Milan:2017)	**19.0**	71.0
Our implementation		
Prediction Layer	5.1	30.3
Update Layer with IoU	17.6	70.8

Table 4.1: Tracking results on MOTChallenge test set.

4.3 Network

The RNN stage contains one layer for each operation, i.e. state prediction, update and Birth/Death. Each of these operations have their individual layers. I created my own custom layers for every operation because the project required some extra bit-wise operations which are not usually supported by the default API of Keras-Tensorflow. The training process is fast and more accurate if done using GPUs compared to normal CPUs. With GPUs it takes around 3 hours to train the network using training data of MOTChallenge 2015. For optimization, I used Keras built-in Adam optimizer with mean-squared error (MSE) loss function, for the minimization of training loss. The choice of MSE is natural since we want to compare the distance between the coordinates of the ground-truth and the estimated bounding boxes.

4.4 Evaluation Metrics

To be able to compare our results to the state-of-the-art approaches we opt for two standard evaluation metrics used commonly in the research on object tracking. These are, Multiple Object Tracking Precision (MOTP), and the Multiple Object Tracking Accuracy (MOTA), that intuitively express a tracker's characteristics and could be used in general performance evaluations. These metrics are proposed by, Bernardin et al. [11] and given by the following equations,

$$MOTP = \frac{\sum_{i,t} d_{i,t}}{\sum_t c_t} \tag{4.1}$$

Here, c_t is the number of matches found for time t, and $d_{i,t}$ is the distance between the ground-truth object and its corresponding estimated hypothesis. It is the total position error for matched object hypothesis pairs over all frames, averaged by the total number of matches made.

On the other hand, MOTA can be defines as,

$$MOTA = 1 - \frac{\sum_t (m_t + fp_t + mme_t)}{\sum_t g_t} \tag{4.2}$$

where m_t, fp_t and mme_t are the number of misses, of false positives, and of mismatches respectively for time t.

4.5 Results

In Table 4.1, we show the results on MOT Challenge 2015 test dataset from different state-of-the-art approaches. At the bottom of the table we also present the results we obtained using our own implementation of [1] with our modifications.

We first make a baseline comparison with the most related work on tracking. At first we have an RNN based approach by [3] with a MOTA score of 37.6. This is currently the highest score in online Multi-object tracking. [3] proposes online method for tracking multiple target using cues such as, image appearance, motion, and inter-relations as their features. Then we have tracking using decision making by [12] with a MOTA of 30.3. In this case the tracking is using Markov decision process (MDP), where each object's lifetime is modeled by an MDP. So MDP handles the Birth/Death of the target and to solve the data association task they used a separate reinforcement learning approach. Lastly the JPDA method by [2] with MOTA of 23.8, here the idea is to associate the detected measurements in each time frame with existing targets using a joint probabilistic score. It is important to note that the MOTP scores of all the approaches are quite similar, including the RNN/LSTM based approach by [1], which we chose to be our baseline for further development.

However, we notice in the same Table 4.1 that the RNN/LSTM based approach by [1], does not reach up to the expected levels of Multi-target tracking accuracy in terms of MOTA score. But it is the first approach that uses RNNs and LSTMs for a multi-target tracking task. Also, since its an online tracking approach, makes it quite interesting for our research and further investigations in the era of deep neural networks.

As shown in the table, our results for the re-implementation of the whole project in Keras-Tensorflow are slightly worse than the original work of [1]. We believe with slightly more investigation and modifications to the network architecture such a minor gap can be overcome. Let us start with the first step in building the network which is training the model for the prediction layer of the RNNs. We also show the results of individual stages from our implementation. For instance, if we just consider the output of the prediction stage, where we do not consider the state update using the measurement data, we get a MOTA value of 5%. This is due to the fact that, in the absence of actual measurements (e.g., detection boxes), the target track will diverge over time without any possibility of recovering. As soon as we apply the update layer, our results approach the results of the original implementation. It is important to note that we do not use the LSTM stage for data association. Instead we propose to integrate the standard intersection-over-union based approach for associating the targets with the detection boxes. This data association is then used during the update layer of our RNN stage. Finally, we get the MOTA value of 17.6%, with the difference of only 1.2% compared to the original work of [1].

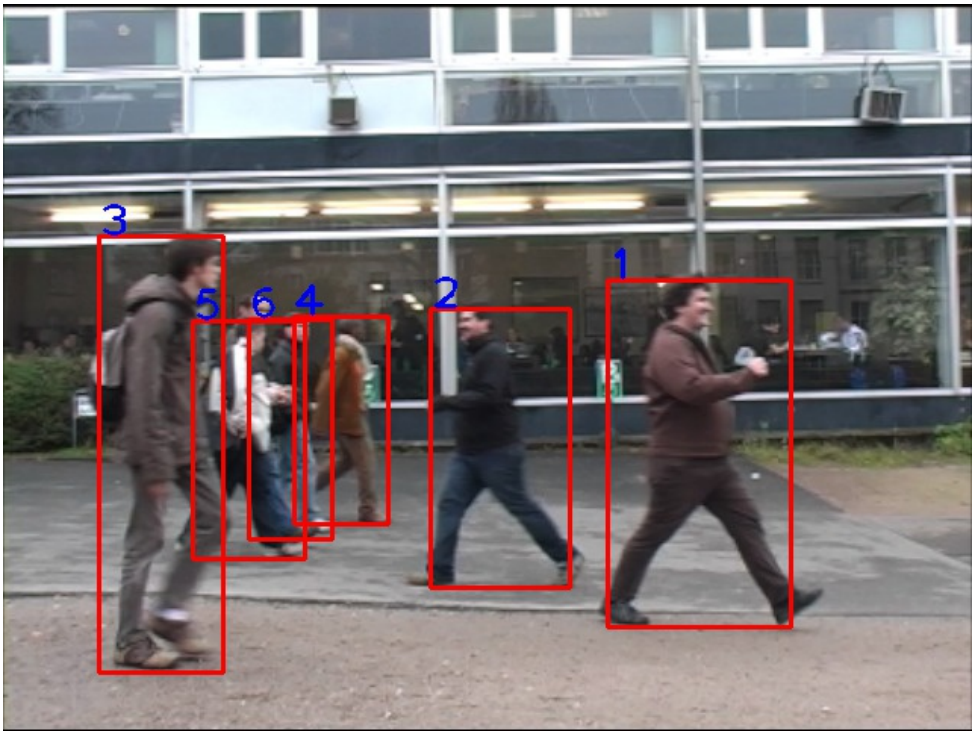

Figure 4.1: Example tracking results on the TU Darmstadt MOT dataset. Frame 1.

Figure 4.2: Example tracking results on the TU Darmstadt MOT dataset. Frame 46.

Chapter 5

Conclusion and Future work

This project represents an approach to solve the non-trivial task of multiple object tracking using recurrent neural nets which are very well used for the problems involving textual data, with this experiment we proposed a way to use a sequential image data. To solve the specific task of data association in such settings, LSTMs were originally proposed. In our work, we simplified the model and proposed the IoU-based data association technique to replace the complex LSTM stage. We show that similar results can be obtained using such a simpler approach. For the next steps, we believe that introducing more features like appearance, object to object interaction, optical flow and their motion can help in improving the results for an online MOT approach.

As a further future work, we would also like to take this framework to calculate 3D position tracking to calculate and track targets using ground coordinates setting.

Bibliography

[1] A. Milan, S. H. Rezatofighi, A. Dick, I. Reid, and K. Schindler. Online multi-target tracking using recurrent neural networks. In *Association for Advancement of Artificial Intelligence (AAAI)*, February 2017.

[2] Seyed Hamid Rezatofighi, Anton Milan, Zhen Zhang, Qinfeng Shi, Anthony Dick, and Ian Reid. Joint probabilistic matching using m-best solutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[3] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. *CoRR*, abs/1701.01909, 2017.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[5] Oriol Vinyals and Quoc V. Le. A neural conversational model. *CoRR*, abs/1506.05869, 2015.

[6] C. Schmaltz, B. Rosenhahn, T. Brox, D. Cremers, J. Weickert, L. Wietzke, and G. Sommer. Occlusion modeling by tracking multiple objects. In *Pattern Recognition (Proc. DAGM)*, Lecture Notes in Computer Science. Springer, Sept. 2007.

[7] Tao Yang, Stan Z. Li, Quan Pan, and Jing Li. Real-time multiple objects tracking with occlusion handling in dynamic scenes. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 970–975, Washington, DC, USA, 2005. IEEE Computer Society.

[8] Christian Schmaltz, Bodo Rosenhahn, Thomas Brox, Joachim Weickert, Daniel Cremers, Lennart Wietzke, and Gerald Sommer. *Occlusion Modeling by Tracking Multiple Objects*, pages 173–183. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[9] Yu Xiang, Changkyu Song, Roozbeh Mottaghi, and Silvio Savarese. Monocular multiview object tracking with 3d aspect parts. In *European Conference on Computer Vision (ECCV)*, 2014.

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

[11] Keni Bernardin, Alexander Elbs, and Rainer Stiefelhagen. Multiple object tracking performance metrics and evaluation in a smart room environment. 2006.

[12] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to track: Online multi-object tracking by decision making. In *International Conference on Computer Vision*, 2015.