# Multi-object Tracking with Neural Gating Using Bilinear LSTM

Chanho Kim[1], Fuxin Li[2], and James M. Rehg[1]

[1] Center for Behavioral Imaging
Georgia Institute of Technology, Atlanta GA, USA
{chkim, rehg}@gatech.edu
[2] Oregon State University, Corvallis OR, USA
lif@oregonstate.edu

**Abstract.** In recent deep online and near-online multi-object tracking approaches, a difficulty has been to incorporate long-term appearance models to efficiently score object tracks under severe occlusion and multiple missing detections. In this paper, we propose a novel recurrent network model, the Bilinear LSTM, in order to improve the learning of long-term appearance models via a recurrent network. Based on intuitions drawn from recursive least squares, Bilinear LSTM stores building blocks of a linear predictor in its memory, which is then coupled with the input in a multiplicative manner, instead of the additive coupling in conventional LSTM approaches. Such coupling resembles an online learned classifier/regressor at each time step, which we have found to improve performances in using LSTM for appearance modeling. We also propose novel data augmentation approaches to efficiently train recurrent models that score object tracks on both appearance and motion. We train an LSTM that can score object tracks based on both appearance and motion and utilize it in a multiple hypothesis tracking framework. In experiments, we show that with our novel LSTM model, we achieved state-of-the-art performance on near-online multiple object tracking on the MOT 2016 and MOT 2017 benchmarks.

## 1 Introduction

With the improvement in deep learning based detectors [16, 35] and the stimulation of the MOT challenges [32], tracking-by-detection approaches for multi-object tracking have improved significantly in the past few years. Multi-object tracking approaches can be classified into three types depending on the number of lookahead frames: online methods that generate tracking results immediately after processing an input frame [33, 1, 22], near-online methods that look ahead a fixed number of frames before consolidating the decisions [24, 7], and batch methods that consider the entire sequence before generating the decisions [39, 38]. For tracking multiple people, a recent state-of-the-art batch approach [38] relies upon person re-identification techniques which leverage a deep CNN network that can recognize a person that has left the scene and re-entered. Such

an approach is able to thread together long tracks in which a person is not visible for dozens of frames, whereas the margin for missing frames in online and near-online approaches is usually much shorter.

A key challenge in online and near-online tracking is the development of deep appearance models that can automatically adapt to the diverse appearance changes of targets over multiple video frames. A few approaches based on Recurrent Neural Networks (RNNs) [33, 1] have been proposed in the context of multi-object tracking. [33] focuses on building a non-linear motion model and a data association solver using RNNs. [1] successfully adopted Long Short-Term Memory (LSTM) [21] to integrate appearance, motion, and interaction cues, but Figure 7. (b) in [1] reports results for sequences (tracks) of maximum length 10. In practice, object tracks are much longer than 10 frames, and it is unclear whether the method is equally effective for longer tracks.

Our own experience, coupled with the reported literature, suggests that it is difficult to use LSTMs to model object appearance over long sequences. It is therefore worthwhile to investigate the fundamental issues in utilizing LSTM for tracking, such as what is being stored in their internal memory and what factors result in them being either able or unable to learn good appearance models. Leveraging intuition from classical recursive least squares regression, we propose a new type of LSTM that is suitable for learning sequential appearance models. Whereas in a conventional LSTM, the memory and the input have a linear relationship, in our *Bilinear LSTM*, the LSTM memory serves as the building blocks of a *predictor* (classifier/regressor), which leads to the output being based on a multiplicative relationship between the memory and the input appearance. Based on this novel LSTM formulation, we are able to build a recurrent network for scoring object tracks that combines long-term appearance and motion information. This new track scorer is then utilized in conjunction with an established near-online multi-object tracking approach, multiple hypothesis tracking, which reasons over multiple track proposals (hypotheses). Our approach combines the benefits of deep feature learning with the practical utility of a near-online tracker.

Our second contribution is a training methodology for generating sequential training examples from multi-object tracking datasets that accounts for the cases where detections could be noisy or missing for many frames. We have developed systematic data augmentation methods that allow our near-online approach to take advantage of long training sequences and survive scenarios with detection noise and dozens of frames of consecutive missing detections.

With these two improvements, we are able to generate state-of-the-art multi-target tracking results for near-online approaches in the MOT challenge. In the future, our proposed Bilinear LSTM could be used in other scenarios where a long-term online predictor is needed.

## 2 Related Work

There is a vast literature on multi-target tracking. Top-performing tracking algorithms that do not train a deep network include [7, 29, 24]. These methods

usually utilize long-term appearance models as well as structural cues and motion cues. A review of earlier tracking papers can be found in [27].

The prior work that is closest to ours uses RNNs as a track proposal classifier in the Markov Decision Process (MDP) framework [1]. Three different RNNs that handle appearance, motion, and social information are trained separately for track proposal classification and then combined for joint reasoning over multiple cues to achieve the best performance. Our method is different from this approach both in terms of the network architecture and training sequence generation from ground truth tracks. Also, we present the first incorporation of deep learned track model into an MHT framework.

Other recent approaches [37, 28] adopt siamese networks that learn the matching function for a pair of images. The network is trained for the binary classification problem where the binary output represents whether or not the image pair comes from the same object. The matching function can be utilized in a tracking framework to replace any previous matching function. Approaches in this category are limited to only modeling the information between a pair of the detections, whereas our approach can model the interaction between a track and a detection, thereby exploiting long-term appearance and motion information.

Milan et al. [33] presented a deep learning framework that solves the multi-object tracking problem in an end-to-end trainable network. Unlike our approach, they attempted to solve state estimation and data association jointly in one framework. While this was highly innovative, an advantage of MHT is the ability to use highly optimized combinatoric solvers.

RNN has been applied in single-object tracking [42, 18], however multi-target tracking is a more challenging problem due to the amount of occlusion and problem of ID switches, which is much more likely to happen in a multi-object setting.

## 3 Overview of MHT

In tracking-by-detection, multi-object tracking is solved through data association, which generates a set of tracks by assigning a track label to each detection. MHT solves the data association problem by explicitly generating multiple track proposals and then selecting the most promising ones. Let $T_l(t) = \{d_1^l,\ d_2^l,\ \dots,\ d_{t-1}^l,\ d_t^l\}$ denote the $l^{\text{th}}$ track proposal at frame $t$ and let $d_t^l$ be a detection selected by the $l^{\text{th}}$ track proposal at frame $t$. The selected detection $d_t^l$ can be either an actual detection generated by an object detector or a dummy detection that represents a missing detection.

The track proposals for each object are stored in a track tree in which each tree node corresponds to one detection. For example, the root node represents the first detection of the object and the child nodes represent the detections in subsequent frames (i.e. tree nodes at the same depth represent detections in the same frame). Thus, multiple paths from the root to the leaf nodes correspond to multiple track proposals for a single object. The proposals are scored, and the task of finding the best set of proposals can be formulated as a Maximum

Weighted Independent Set (MWIS) problem [34], with the score for each proposal being the weight of it. Once the best set of proposals is found, proposal pruning is performed. Only the surviving proposals are kept and updated in the next frame. More details about MHT can be found in [24, 34].

### 3.1 Gating in MHT

In MHT, track proposals are updated by extending existing track proposals with new detections. In order to keep the number of proposals manageable, exisiting track proposals are not updated with all of the new detections, but rather with a few selected detections. The selection process is called *gating*. Previous gating approaches rely on hand-designed track score functions [24, 34, 5, 9]. Typically, the proposal score $S(T_l(t))$ is defined recursively as:

$$S(T_l(t)) = S(T_l(t-1)) + \Delta S(T_l(t)) \tag{1}$$

Gating is done by thresholding the score increment $\Delta S(T_l(t))$. New track proposals with score increments below a certain threshold are pruned instantly. Usually the proposal score includes an appearance term, which could be learned by recursive least squares, as well as a motion term which could be learned with Kalman filtering.

### 3.2 Recursive Least Squares as an Appearance Model

An important advantage of our previous MHT-DAM approach [24] is the use of long-term appearance models that leverage all prior appearance samples from a given track and train a discriminative model to predict whether each bounding box belongs to each given track. Because we would like to be able to perform a similar task in our LSTM framework, we briefly review the recursive least squares appearance model used in [24]. Given all the $n_t$ detections at frame $t$, one can extract appearance features (e.g. CNN fully-connected layer) for them and store them in an $n_t \times d$ matrix $\mathbf{X}_t$, where $d$ is the feature dimensionality. Then, suppose that we are tracking $k$ object tracks, an output vector can be created for each track as, e.g. the spatial overlap between the bounding box of each detection and each track (represented by one detection in the frame), with the set of output vectors denoted as an $n_t \times k$ matrix $\mathbf{Y}_t$. Then a regressor for each target can be found by least squares regression:

$$\min_{\mathbf{W}} \sum_{t=1}^{T} \|\mathbf{X}_t\mathbf{W} - \mathbf{Y}_t\|_F^2 + \lambda\|\mathbf{W}\|_F^2 \tag{2}$$

where $\|\cdot\|_F^2$ is a squared Frobenius norm and $\lambda$ is the regularization parameter. As is well-known, the solution can be written as:

$$\mathbf{W} = \left(\sum_{t=1}^{T} \mathbf{X}_t^\top \mathbf{X}_t + \lambda\mathbf{I}\right)^{-1} \left(\sum_{t=1}^{T} \mathbf{X}_t^\top \mathbf{Y}_t\right) \tag{3}$$

where $\mathbf{I}$ is the identity matrix. Notably, one can store $\mathbf{Q}_t = \sum_{i=1}^{t} \left(\mathbf{X}_i^\top \mathbf{X}_i\right)$ and $\mathbf{C}_t = \sum_{i=1}^{t} \left(\mathbf{X}_i^\top \mathbf{Y}_i\right)$ and update them online at frame $t+1$, by adding $\mathbf{X}_{t+1}^\top \mathbf{X}_{t+1}$ and $\mathbf{X}_{t+1}^\top \mathbf{Y}_{t+1}$ to $\mathbf{Q}_t$ and $\mathbf{C}_t$ respectively, while maintaining the optimality of the solution for $\mathbf{W}$. Moreover, the computation of $\mathbf{W}$ is only linear in the number of tracks $k$. The resulting model can train on all the positive examples (past detections in each track) and negative examples (past detections in other tracks not overlapping with a given track) and generate a regressor with good discriminative power. The computational efficiency of this approach and its optimality are the two keys to the success of the MHT-DAM framework.

## 4 RNN as a Gating Network

We use the term *gating network* to denote a neural network that performs gating. We utilize recurrent neural networks (RNNs) for training gating networks since track proposals constitute sequential data whose data size is not fixed. In this work, we adopt Long Short-Term Memory (LSTM) as a recurrent layer due to its success in modeling long sequences on various tasks [17].

We formulate the problem of gating as a sequence labeling problem. The gating network takes track proposals as inputs and performs gating by generating a binary output for every detection in the track proposal. In this section, we describe network inputs and outputs and its utilization within the MHT framework. More details about the network architecture can be found in Sec. 4.2.

**Input.** Track proposals contain both motion and appearance information. We use the bounding box coordinates $(x, y, w, h)$ over time as motion inputs to the network. The coordinates are normalized with respect to the frame resolution $(\frac{x}{\text{image width}}, \frac{y}{\text{image height}}, \frac{w}{\text{image width}}, \frac{h}{\text{image height}})$ to make the range of the input values fixed regardless of the frame resolution. We also calculate sample mean and standard deviation from track proposals (see Sec. 5 for more details on how to generate track proposals from multi-object tracking datasets) and perform another normalization in order to make the input data zero-centered and normalized across different dimensions.

We use object images cropped to detection bounding boxes as appearance inputs to the network. RGB cropped images are first converted to convolutional features by Convolutional Neural Networks (CNN) before the gating networks process them. We use the ImageNet pretrained ResNet-50 [19] as our CNN.

**Output.** Given a current detection, the network makes a binary decision about whether or not it belongs to the proposal based on its compatibility with the appearance and motion of the other detections assigned to the proposal. Thus, the gating networks solve a binary classification task using cross-entropy loss. Note that we have multiple binary labels for each track sequence since gating is done on every frame.

**Track Scorer in MHT.** We use the softmax probability $p$ of the positive output (i.e. current detection belongs to the same object in the proposal) for calculating the score increment $\Delta S(T_l(t))$ as shown in Eq.(4). A higher score

increment implies a higher matching quality between the track proposal $T_l(t-1)$ and the detection $d_t^l$.

$$\Delta S(T_l(t)) = p(d_t^l \in T_l(t-1)|T_l(t-1)) \tag{4}$$

This is a simple aggregation scheme that combines the per-frame predictions from the gating network in order to score tracks. Our assumption is that proposals that generate higher per-frame matching scores are more likely to be correct than proposals with lower per-frame matching scores. New track proposals with score increment below a threshold are pruned instantly by gating. In MHT, every track proposal in the track trees has a unique detection sequence, which is represented as a unique LSTM memory state in the gating network. The memory state for surviving proposals is stored for further gating and scoring in the next frame. The weights of the gating network are shared across all track proposals.

### 4.1 Bilinear LSTM

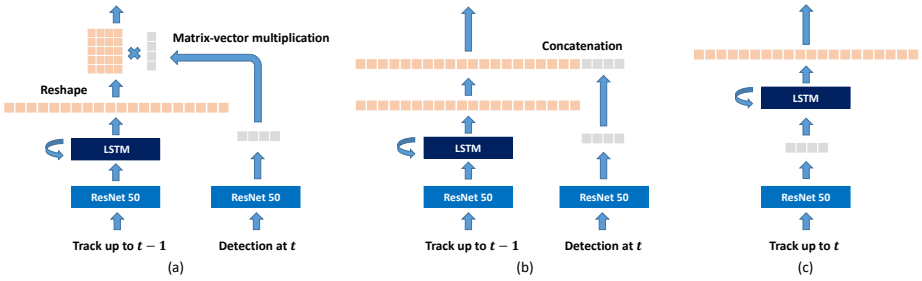


Fig. 1: Motion gating network and appearance gating network are trained separately before training the full model. We evaluate multiple network architectures for each module. (a) The Bilinear LSTM network with a multiplicative relationship between the memory and the input CNN features. The LSTM memory is reshaped into a matrix and multiplied with the input appearance feature vector; (b) Input appearance is concatenated with the LSTM memory output before a fully-connected layer; (c) A conventional LSTM architecture.

Our experience suggests that conventional LSTMs are far more effective at modeling motion than appearance. This led us to ask, "what information about object appearance is being stored in the internal memory of a standard LSTM, and what would be an ideal memory representation for this task?".

Conventional LSTMs utilizes the following update rule:

$$\begin{aligned} \mathbf{c}_t &= \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \mathbf{g}_t, \quad & \mathbf{h}_t &= \mathbf{o}_t \circ \tanh(\mathbf{c}_t) \\ \mathbf{f}_t &= \sigma(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{x}_t]), \quad & \mathbf{i}_t &= \sigma(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{x}_t]), \\ \mathbf{g}_t &= \sigma(\mathbf{W}_g[\mathbf{h}_{t-1}, \mathbf{x}_t]), \quad & \mathbf{o}_t &= \tanh(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t]) \end{aligned} \tag{5}$$

where $\circ$ represents the Hadamard product. $\mathbf{x}_t$ is the current input. $\mathbf{f}_t$, $\mathbf{i}_t$, and $\mathbf{o}_t$ are the forget gate, the input gate, and the output gate. $\mathbf{c}_t$ and $\mathbf{h}_t$ are the cell

state and the hidden state that are repeatedly updated throughout the sequence. $\mathbf{g}_t$ is the new update values for the cell state.

When building an appearance model for multi-object tracking (i.e. data association), $\mathbf{x}_t$ represents the current appearance of an object candidate. For LSTM to solve the tracking task, one intuition is that $\mathbf{h}_t$ may represent some information about the acceptance/rejection of an object candidate. $\mathbf{c}_t$ can roughly be thought of as representing a stored template of the object appearance, and then the output gate $\mathbf{o}_t$ compares the previous stored appearance $\mathbf{c}_{t-1}$ and the new appearance $\mathbf{x}_t$ in order to decide the current output $\mathbf{h}_t$. Experiments in [1] where LSTM performance seems to saturate with a sequence length of $2-4$ frames, seems to suggest that the aforementioned intuition might be partially correct.

However, the main appeal of a long-term appearance model in previous work is the capability of using a classifier/regressor that learns from *all* the previous appearances of the object [24]. Such a model trained from multiple different appearances could generalize better than one or a few stored templates and could potentially interpolate and extrapolate among different previous appearances of the model. An example would be the recursive least squares model in Eq. (2). But if we imagine the $\mathbf{W}$ in Eq. (2) as the memory output $\mathbf{h}_t$, it seems that a multiplicative form $\mathbf{x}^\top\mathbf{W}$ as in Eq. (2) is difficult to obtain from the additive forms in Eq. (5), no matter from $\mathbf{o}_t, \mathbf{c}_t$ or $\mathbf{h}_t$.

Thus, we would like to propose a new LSTM that can realize the multiplicative between the memory $\mathbf{h}_t$ and the input $\mathbf{x}$. We note that the solution of recursive least squares is dependent on the matrix $\mathbf{Q}_t = \sum_{i=1}^{t} \mathbf{X}_i^\top \mathbf{X}_i$ that is updated at each time linearly. It is difficult for LSTM to store a positive-definite matrix as memory, but a common approach to simplify such a positive-definite matrix is to use a low-rank approximation, e.g. assuming $\mathbf{Q}_t^{-1} = \sum_{i=1}^{r} \mathbf{q}_{ti}\mathbf{q}_{ti}^\top$. With this assumption and considering Eq. (3), the regressor output becomes:

$$\mathbf{w}^\top\mathbf{x} = \mathbf{C}_t^\top\mathbf{Q}_t^{-1}\mathbf{x} = \sum_{i=1}^{r} \mathbf{C}_t^\top\mathbf{q}_{ti}\mathbf{q}_{ti}^\top\mathbf{x} \tag{6}$$

Note that when there is only 1 track, $\mathbf{C}_t$ is of the dimensionality $d \times 1$, and hence $\mu_i = \mathbf{C}_t^\top\mathbf{q}_i$ is a scalar. We have:

$$\mathbf{w}^\top\mathbf{x} = \sum_{i=1}^{r} \mu_i\mathbf{q}_{ti}^\top\mathbf{x} \tag{7}$$

Here $\mu_i$ is dependent on both $\mathbf{y}$ and $\mathbf{q}$, hence without loss of generality it could be a standalone variable that is separately estimated. With this derivation, it seems that the approach to emulate a linear regressor is to have several vectors $\mathbf{h}_{ti}$ to be learnable and gradually changing with time (in other words, serve as the memory in the LSTM), and a layer of learnable $\mu_i$ on top of a multiplicative relationship between $\mathbf{h}_{ti}$ and $\mathbf{x}$.

In this spirit, we propose the Bilinear LSTM (bLSTM) which utilizes the following forward pass that enables the multiplicative interaction between the

input and the memory:

$$\begin{aligned} \mathbf{h}_{t-1} &= [\mathbf{h}_{t-1,1}^\top | \mathbf{h}_{t-1,2}^\top | ... | \mathbf{h}_{t-1,r}^\top]^\top = \mathbf{o}_{t-1} \circ \tanh(\mathbf{c}_{t-1}) \\ \mathbf{H}_{t-1}^{\text{reshaped}} &= [\mathbf{h}_{t-1,1} | \mathbf{h}_{t-1,2} | ... | \mathbf{h}_{t-1,r}]^\top, \quad \mathbf{m}_t = f(\mathbf{H}_{t-1}^{\text{reshaped}} \mathbf{x}_t) \end{aligned} \tag{8}$$

where $f(\cdot)$ is a non-linear activation function and $\mathbf{m}_t$ is the new hidden state for bLSTM. $\mathbf{x}_t$ denotes the features from a box at frame $t$. Basically, we utilize a long vector as the LSTM memory which contains the concatenation of all the $\mathbf{h}_{t-1,i}$s. When it comes to the time to multiply $\mathbf{h}_{t-1,i}$ with $\mathbf{x}_t$, the $rd$ dimensional vector $\mathbf{h}_{t-1}$ is reshaped into the $r \times d$ matrix $\mathbf{H}_{t-1}^{\text{reshaped}}$ so that we could utilize matrix-vector multiplication between $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$.

The new hidden state $\mathbf{m}_t$ can then be used as input to other fully-connected layers (resembling the $\mu_i$ in eq. (7)) to generate the final prediction. Note that in online recursive least squares $\mu_i$ should be trained for each tracked object separately, however in our network the fully-connected layers after bLSTM are trained globally and fixed during testing time. Implementing a dynamic $\mu_i$ which is dependent on each object track did not result in significant improvement in performance. We believe that since the system is trained end-to-end, the LSTM updates of $\mathbf{h}$ should be able to encompass the potential changes in $\mu_i$, hence we can keep the fully-connected layers fixed without additional issues.

Intuitively, by saving a matrix-valued memory that resembles a low-rank decomposition of the matrix, at least r templates (as well as combinations of the r templates) can be used for the prediction. Hence bLSTM can store longer-term appearance models than traditional LSTMs and improve on maintaining track identity over many frames.

### 4.2 Network Architecture

We have three types of gating networks based on the network input: Motion, Appearance, and Motion + Appearance. We test three different architectures in Fig. 1 for the motion gating and appearance gating networks. We select the best architecture for each type of input among the three and combine them for motion+appearance gating networks. Experimental results that we used for selecting the architecture are included in Sec. 6.3.

**Motion Gating.** For motion gating, the vanila version of LSTM eq. (5) works the best. Thus, we adopt LSTM as a sequence labeler where LSTM reads motion input recursively and store the sequence information in its hidden state. The FC layers are built on top of the hidden state to produce the final output. Architectures that we used for the comparison are shown in Table 1.

**Appearance Gating.** We propose to use Bilinear LSTM as appearance gating network where LSTM's hidden state becomes a weight vector for the appearance model of the current object. Details about the network architecture and other two baseline architectures are shown in Table 2.

**Motion + Appearance Gating.** In order to enable a joint reasoning over both motion and appearance for object tracking, we construct a motion+gating

| Soft-max | | | |
|---|---|---|---|
| Matrix-vector Multiplication-tanh 4 | | | |
| Reshape | $4 \times 64$ | Reshape | $64 \times 1$ |
| LSTM | 256 | | |
| FC-relu | 64 | FC-relu | 64 |
| Input at $t-1$ | 4 | Input at $t$ | 4 |

(a)

| Soft-max | | | |
|---|---|---|---|
| FC-tanh 64 | | | |
| Concatenation $64 + 64$ | | | |
| LSTM | 64 | | |
| FC-relu | 64 | FC-relu | 64 |
| Input at $t-1$ | 4 | Input at $t$ | 4 |

(b)

| Soft-max | |
|---|---|
| FC-tanh | 8 |
| LSTM | 64 |
| FC-relu | 64 |
| Input at $t$ | 4 |

(c)

Table 1: Different experimented architectures for motion gating. (a) Bilinear LSTM (b) LSTM as a feature extractor for the previous track (c) Vanila LSTM (LSTM as a sequence labeler)

| Soft-max | | | |
|---|---|---|---|
| Matrix-vector Multiplication-relu 8 | | | |
| Reshape | $8 \times 256$ | Reshape | $256 \times 1$ |
| LSTM | 2048 | | |
| FC-relu | 256 | FC-relu | 256 |
| ResNet-50 | 2048 | ResNet50 | 2048 |
| Input at $t-1$ | $128 \times 64 \times 3$ | Input at $t$ | $128 \times 64 \times 3$ |

(a)

| Soft-max | | | |
|---|---|---|---|
| FC-relu 512 | | | |
| Concatenation $2048 + 256$ | | | |
| LSTM | 2048 | | |
| FC-relu | 256 | FC-relu | 256 |
| ResNet-50 | 2048 | ResNet50 | 2048 |
| Input at $t-1$ | $128 \times 64 \times 3$ | Input at $t$ | $128 \times 64 \times 3$ |

(b)

| Soft-max | |
|---|---|
| FC-relu | 512 |
| LSTM | 2048 |
| FC-relu | 256 |
| ResNet-50 | 2048 |
| Input at $t$ | $128 \times 64 \times 3$ |

(c)

Table 2: Different experimented architectures for appearance gating: (a) Bilinear LSTM (b) LSTM as a feature extractor for the previous track (c) Vanila LSTM

network based on our analysis of different baseline architectures. We use Bilinear LSTM to process appearance data and vanila LSTM to process motion data. Then motion and appearance representations (i.e. outputs before soft-max) from both gating networks are concatenated after L2 normalization is applied to each representation separately. Prediction layers are built upon the concatenated features. We first train motion gating and appearance gating networks separately. Then we load all the pretrained layers before the concatenation layer from both gating networks and fine-tune them jointly.

### 4.3 Handling Missing Detections

In tracking-by-detection, it is important to handle missing detections while keeping the correct track identity over time. In traditional Kalman filter-based motion tracking, the diagonal of the noise covariance matrix keeps increasing over time in the case of missing detections, resulting in accepting more detections from gating with a gradually larger gating area.

In the case of recurrent gating networks, the occurrences of missing detections should also modulate the gating network. For instance, one can imagine a gating network applying a stricter gating policy when all the detections are available for the current object than the case where detections are missing in recent frames. In order to encode such information inside the LSTM hidden states, we propose to input to the recurrent network all-zero input vectors in the case of missing detections. By doing so, the LSTM internal (both cell and hidden) states will be updated solely based on its previous states, which is different from normal LSTM

updates where both the input data and the previous state are utilized. The gating network does not need to make any prediction for the missing detection but only need to update the LSTM internal memory. In Sec. 6.3, we show the effectivenss of such explicit missing detection handling for the motion gating networks.

## 5 Generating Training Sequences

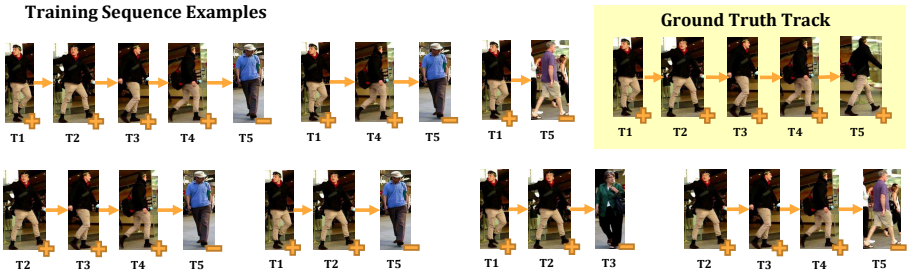


Fig. 2: Training sequences are generated from the public MOT dataset. Each training sequence has detections from the same object throughout the track and one detection from different object at the end. Training sequences are generated in the manner that they reflect actual track proposals that MHT generates during tracking.

Artificial track proposals are generated from ground truth track annotations as training data for training our LSTM network. First, we randomly pick one ground truth track annotation from which we sample track proposals. The starting frame and the ending frame are randomly selected. Due to the memory limit of GPUs, we select them in a way that the length of the track proposal does not exceed $N_{\max}$. Let $N$ be the selected length ($2 \leq N \leq N_{\max}$) of the proposal. Then we collect first $N-1$ bounding boxes of the selected object and pick the last $N^{\text{th}}$ detection from a different object. Positive labels are assigned for the first $N-1$ detections representing the correct object and a negative label is assigned for the $N^{\text{th}}$ detection representing a different object. Thus, each proposal is associated with a binary label vector where only the last element is a negative label as presented in Fig. 2. The maximum length $N_{\max}$ needs to be large enough so that the network learns the gating mechanism regardless of its input length. We show experimental results with different $N_{\max}$ values in Sec. 6.3.

**Data Augmentation.** If ground truth tracks are used without any augmentation to generate track proposals, each track proposal will consist of bounding boxes perfectly aligned with the object in consecutive frames, which may poorly represent actual track proposals consisting of noisy detections. Thus, it is important to perform proper data augmentation so that the track proposals reflect actual detection noise. There are two types of detection errors which need to be considered: localization error and missing detections.

In order to reflect the localization noise, we jitter the bounding boxes in the training track proposals using a noise model estimated from the training data.

For estimating this noise model, given a set of detections and ground truth annotations, we first assign each detection to its closest ground truth bounding box, and then calculate localization error from each detection to its assigned ground truth bounding box. We then fit a normal distribution to these localization errors for all true positive detections. Since the MOT Challenge Benchmark [32] provides three public detectors (DPM [13], FRCNN [41], SDP [35]) which have different accuracy and noise levels, we estimate a different normal distribution for each public detector. Thus, before the training data generator samples random localization errors for the proposal, it first chooses a normal distribution based on the detector. Then for each bounding box in each track proposal, a different localization error is sampled from the estimated normal distribution.

In order to simulate missing detections, for 50% of the tracks, we randomly pick a missing detection rate $p_{\text{miss}}(0.0 \sim 0.5)$ and drop the bounding boxes in the track proposal according to the selected missing rate except for the first bounding box (current object) and the last bounding box (differerent object). Example track proposals with this missing detection augmentation are shown in Fig. 1. The other 50% of the tracks are retained without missing detections.

## 6 Experiments

We report all our experimental results on the validation set except for the final benchmark result which was evaluated on MOT 16/17 test sequences.

### 6.1 Training Data

In order to generate track proposals, we use MOT17 (MOT16) and MOT15 sequences [32, 26] and a few other tracking sequences [15, 10, 2] where pedestrian annoatations are available. All the training, validation, and testing sequences are shown in Table 3. In addition to the MOT sequences, we also use two public person re-identification datasets, Market1501 [43] and CUHK03 [30], in order to pre-train the appearance gating networks.

| Training Set | Validation Set | Test Set |
|---|---|---|
| MOT17 - {02, 04, 05, 11, 13},<br>MOT15 - {01, 02, 03, 04, 05, 06, 07},<br>ETH - {Jelmoli, Seq01}, KITTI - {16, 19},<br>PETS09-S2L2, TUD-Crossing, AVG-TownCentre | MOT17 - {09, 10} | MOT17 - {01, 03, 06, 07, 08, 12, 14} |

Table 3: Training/Val/Test Splits

### 6.2 Pre-training on Person Re-identification

In the person re-identification task, a pair of images is given to the learner and the learner decides whether two images come from the same person or not. One can treat the pair of two images as a track proposal with a temporal length 2. Such training examples can be also generated from multi-object tracking dataset. Thus, we utilize a person re-identification dataset in addition to the training set

| LSTM | MOTA | IDF1 | IDS |
|---|---|---|---|
| Bilinear | **52.33** | **59.07** | **233** |
| Baseline1 | 50.43 | 51.28 | 412 |
| Baseline2 | 50.97 | 51.49 | 462 |

| State dim. | MOTA | IDF1 | IDS |
|---|---|---|---|
| 512 | 52.14 | 56.66 | 283 |
| 1024 | 52.36 | 55.85 | **222** |
| 2048 | 52.33 | **59.07** | 233 |

| $N_{\max}$ | MOTA | IDF1 | IDS |
|---|---|---|---|
| 10 | 51.96 | 54.36 | 271 |
| 20 | 52.27 | 58.38 | 228 |
| 40 | 52.33 | **59.07** | 233 |
| 80 | 52.32 | 57.21 | 239 |
| 160 | 52.41 | 55.19 | **222** |

Table 4: Ablation Study for Appearance Gating Networks. Baseline1 and Baseline2 are the networks shown in Table 2 (b) and (c) resepectively. **(Left)** State dim. = 2048, $N_{\max} = 40$ **(Middle)** LSTM: Bilinear, $N_{\max} = 40$, **(Right)** LSTM: Bilinear, State dim. = 2048

shown in Table 3 to pre-train our appearance gating network for person re-identification. Similar pre-training was also done in [1, 38]. Table 5 shows the effect of pre-training for person re-identification on the performance of gating networks.

### 6.3 Ablation Study

We conduct an ablation study for different network architectures and training settings on our validation sequences (MOT17-09 and MOT17-10). The MOT17 Benchmark provides three different public detectors. We used the Faster R-CNN detector for the experimental results in this section.

**Metrics.** Among many different tracking metrics, we choose the Multiple Object Tracking Accuracy (MOTA) [4], identity switches (IDS), and IDF1 [36] for this study. MOTA is calculated by object detection mistakes (false positive and false negative) and tracking mistakes (identity switches). MOTA is often dominated by object detection mistakes since the number of false positive/negative is typically much higher than IDS. IDS counts the number of track ID changes for all the objects. IDF1 is a tracking metric which measures how often objects are correctly identified by the same track ID.

**Network Architectures.** We test the three deep architectures shown in Fig. 1 for MHT gating. The results in Table 4 are generated by MHT with different appearance gating networks. Motion gating results for different architectures can be found in the supplementary material. The left table in Table 4 shows the tracking performance of different deep architectures as gating networks. Bilinear LSTM works best as the appearance gating network. In terms of the network sizes (LSTM state dimensions), 2048 state dimension is a good choice for Bilinear LSTM as an appearance gating network.

**Training Settings.** We also try differerent training settings such as different maximum sequence lengths, missing detection augmentations, and network pre-training. The results are included in Tab 4, 5, and 6. We used the (M)+(A) model (in Table 6 (Middle)) which balances well between IDF1 and IDS as our final model for the comparison with MHT and the MOT benchmark in Sec. 6.4.

We used the Adam optimizer [25] for training motion gating networks and set the initial learning rate to 0.01 with the batch size of 64. We used the stochastic gradient optimizer for training appearance and motion+appearance gating

| Input type | MOTA | IDF1 | IDS |
|---|---|---|---|
| (A) Random | 52.00 | 57.46 | 268 |
| (A) Pre-training | 52.33 | **59.07** | **233** |

| Pre-training | MOTA | IDF1 | IDS |
|---|---|---|---|
| (M)+(A) Random | 50.31 | 50.39 | 499 |
| (M)+(A) Pre-training | **52.63** | **58.08** | **197** |

Table 5: Pre-training vs Random initialization **(Left)** LSTM: Bilinear, State dim. = 2048, $N_{\max} = 40$ **(Right)** LSTM: Baseline2 (Motion) + Bilinear (Appearance), State dim. = 64 (Motion), 2048 (Appearance), $N_{\max} = 40$

| Missing Det. | MOTA | IDF1 | IDS |
|---|---|---|---|
| (M) Yes | 52.47 | **50.22** | 229 |
| (M) No | 52.58 | 47.71 | **203** |
| (A) Yes | 52.29 | 41.37 | 244 |
| (A) No | 52.33 | **59.07** | **233** |

| Input type | MOTA | IDF1 | IDS |
|---|---|---|---|
| Motion (M) | 52.47 | 50.22 | 229 |
| Appearance (A) | 52.33 | **59.07** | 233 |
| (M) + (A) | 52.63 | 58.08 | **197** |

| Input type | MOTA | IDF1 | IDS |
|---|---|---|---|
| Motion (M) | 52.30 | 51.14 | 255 |
| Appearance (A) | 52.32 | **57.21** | 239 |
| (M) + (A) | 52.69 | 54.63 | **208** |

Table 6: **(Left)** Missing Detection Augmentation On/Off. $N_{\max} = 40$. We update LSTM states with zero input vectors (as described in Sec. 4.3) for the models which are trained with the missing detection augmentation. The results in this table show that such LSTM state update is beneficial for the motion gating network, but not for the appearance gating network. Thus, we utilize the missing detection handling only for the motion gating network and the motion part in the motion+appearance gating network. **(Middle and Right)** Different input types with different maximum lengths of training sequences (Middle) $N_{\max} = 40$ (Right) $N_{\max} = 80$.

networks and set the initial learning rate to 0.005 with the batch size of 16. In all cases, we let the learning rate decrease every 5000 iterations by exponential decay with the decay rate 0.9 until we observe a decrease in performance on the validation set.

### 6.4 MOT Challenge Benchmark

In this section, we report the performance comparison with MHT-DAM and our tracking results on the MOT Challenge 17/16 Benchmark.

**Comparison with MHT-DAM.** In order to see whether our trained models work well with MHT, we first compare the tracking performance with MHT-DAM [24] on the validation split. Unlike bLSTM, [24] does not benefit from any

| Method | MOTA | IDF1 | IDS |
|---|---|---|---|
| MHT-DAM | **47.6** | 48.2 | **72** |
| Ours | 43.8 | **52.9** | 91 |

| Method | MOTA | IDF1 | IDS |
|---|---|---|---|
| MHT-DAM | 53.7 | 54.8 | **136** |
| Ours | **54.8** | **60.5** | 140 |

| Method | MOTA | IDF1 | IDS |
|---|---|---|---|
| MHT-DAM | **69.4** | 62.7 | **128** |
| Ours | **69.7** | **68.6** | 137 |

Table 7: Comparison with MHT-DAM on our val split (MOT17-02 and MOT17-11). $N_{\max} = 80$. Tracks are interpolated through smoothing. **(Left)** DPM **(Middle)** Faster R-CNN **(Right)** SDP.

off-line training using multi-object tracking datasets. It rather builds appearance models for multiple objects in an online manner. Table 7 shows the comparison in results. Our new MHT with bLSTM works well when Faster RCNN and SDP provide input detections. However, for the case of DPM, MOTA score is lower compared to MHT-DAM, although our new method still shows stronger performance on IDF1. We believe that this is because DPM produces quite noisy detections while we use ground truth tracks to generate training sequences for our model. Thus, there could be still some gap between our training data (even after the data augmentation) and track proposals constructed from DPM.

**MOT16/17 Challenge Benchmarks.** We used the same model and setting as shown in Table 7 as our final method for evaluating on the MOT test sequences. The results are included in Table 8. we grouped previous methods that are closely related to our method separately in order to see the performance difference among these methods.

Table 8: Results from MOT 2017/2016 Challenge (accessed on 7/26/2018)

| Method | MOTA | IDF1 | IDS | Hz |
|---|---|---|---|---|
| JCC [23] | **51.2** | **54.5** | **1,802** | 1.8 |
| MOTDT17* [31] | 50.9 | 52.7 | 2,474 | **18.3** |
| PHD-GSDL17 [14] | 48.0 | 49.6 | 3,998 | 6.7 |
| FWT* [20] | **51.3** | 47.6 | 2,648 | 0.2 |
| MHT Methods | | | | |
| EDMT17* [6] | 50.0 | 51.3 | 2,264 | 0.6 |
| MHT-DAM [24] | **50.7** | 47.2 | 2,314 | 0.9 |
| MHT-bLSTM* | 47.5 | **51.9** | **2,069** | **1.9** |

* indicates the use of additional training data

| Method | MOTA | IDF1 | IDS | Hz |
|---|---|---|---|---|
| NOMT [7] | 46.4 | **53.3** | **359** | 2.6 |
| MCjoint [23] | 47.1 | 52.3 | 370 | 0.6 |
| LMP* [38] | **48.8** | 51.3 | 481 | 0.5 |
| STAM16 [8] | 46.0 | 50.0 | 473 | 0.2 |
| RAR16pub [12] | 45.9 | 48.8 | 648 | 0.9 |
| NLLMPa [29] | 47.6 | 47.3 | 629 | **8.3** |
| JMC [40] | 46.3 | 46.3 | 657 | 0.8 |
| LINF1 [11] | 41.0 | 45.7 | 430 | 4.2 |
| CDA-DDALv2* [3] | 43.9 | 45.1 | 676 | 0.5 |
| MHT and LSTM-based Methods | | | | |
| EDMT* [6] | 45.3 | **47.9** | 639 | **1.8** |
| AMIR* [1] | **47.2** | 46.3 | 774 | 1.0 |
| MHT-DAM [24] | 45.8 | 46.1 | **590** | 0.8 |
| MHT-bLSTM* | 42.1 | **47.8** | 735 | **1.8** |

## 7 Conclusion

In this paper, we proposed using an LSTM network to score track proposals in a near-online multiple hypothesis tracking framework. In order to properly take into account multiple past appearances, we proposed a Bilinear LSTM algorithm that slices the LSTM memory into several vectors and uses a matrix vector multiplication between the memory output and the appearance input to simulate a discriminatively trained predictor model. Such an algorithm is shown to be significantly better than traditional LSTM in modeling the appearance of each track, especially in terms of maintaining track identities. Jointly using appearance and motion LSTM gating networks in an MHT framework, we have achieved state-of-the-art performances in the MOT challenges for near-online methods. We believe the proposed Bilinear LSTM is general and could be applicable in many other problems that require learning an online sequential discriminative model using an end-to-end approach and will explore those as future work.

**Acknowledgments**: This work was supported in part by NIH award 1R24O D020174-01A1. Fuxin Li was partially supported by NSF award 1751402, and DARPA under contract N66001-17-2-4030.

## References


1. A. Sadeghian, A. Alahi, S.S.: Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In: ICCV (2017)
2. Andriluka, M., Roth, S., Schiele, B.: People-tracking-by-detection and people-detection-by-tracking. In: CVPR (2008)
3. Bae, S.H., Yoon, K.J.: Confidence-based data association and discriminative deep appearance learning for robust online multi-object tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence **40**, 595–610 (2018)
4. Bernardin, K., Stiefelhagen, R.: Evaluating multiple object tracking performance: the CLEAR MOT metrics. Image and Video Processing (2008)
5. Blackman, S.: Multiple hypothesis tracking for multiple target tracking. Aerospace and Electronic Systems Magazine **19**(1), 5–18 (2004)
6. Chen, J., Sheng, H., Zhang, Y., Xiong, Z.: Enhancing detection model for multiple hypothesis tracking. In: CVPR Workshops (2017)
7. Choi, W.: Near-online multi-target tracking with aggregated local flow descriptor. In: ICCV (2015)
8. Chu, Q., Ouyang, W., Li, H., Wang, X., Liu, B., Yu, N.: Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism. In: ICCV (2017)
9. Cox, I.J., Hingorani, S.L.: An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence (1996)
10. Ess, A., Leibe, B., Schindler, K., , van Gool, L.: A mobile vision system for robust multi-person tracking. In: CVPR (2008)
11. Fagot-Bouquet, L., Audigier, R., Dhome, Y., Lerasle, F.: Improving multi-frame data association with sparse representations for robust near-online multi-object tracking. In: ECCV (2016)
12. Fang, K., Xiang, Y., Li, X., Savarese, S.: Recurrent autoregressive networks for online multi-object tracking. In: WACV (2018)
13. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part based models. IEEE Transactions on Pattern Analysis and Machine Intelligence **32**(9), 1627–1645 (2010)
14. Fu, Z., Feng, P., Angelini, F., Chambers, J.A., Naqvi, S.M.: Particle phd filter based multiple human tracking using online group-structured dictionary learning. IEEE Access **6**, 14764–14778 (2018)
15. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: CVPR (2012)
16. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR (2014)
17. Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y.: Deep learning, vol. 1. MIT press Cambridge (2016)
18. Gordon, D., Farhadi, A., Fox, D.: Re3: Re al-time recurrent regression networks for visual tracking of generic objects. IEEE Robotics and Automation Letters **3**(2), 788–795 (2018)
19. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
20. Henschel, R., Leal-Taix, L., Cremers, D., Rosenhahn, B.: Fusion of head and full-body detectors for multi-object tracking. In: CVPR Workshops (2018)

21. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**, 1735–80 (12 1997)
22. Hong Yoon, J., Lee, C.R., Yang, M.H., Yoon, K.J.: Online multi-object tracking via structural constraint event aggregation. In: CVPR. pp. 1392–1400 (2016)
23. Keuper, M., Tang, S., Yu, Z., Andres, B., Brox, T., Schiele, B.: A multi-cut formulation for joint segmentation and tracking of multiple objects. arXiv:1607.06317 (2016)
24. Kim, C., Li, F., Ciptadi, A., Rehg, J.: Multiple hypothesis tracking revisited. In: ICCV (2015)
25. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015)
26. Leal-Taixé, L., Milan, A., Reid, I., Roth, S., Schindler, K.: MOTChallenge 2015: Towards a benchmark for multi-target tracking. arXiv:1504.01942 (2015)
27. Leal-Taixe, L., Milan, A., Schindler, K., Cremers, D., Reid, I., Roth, S.: Tracking the trackers: an analysis of the state of the art in multiple object tracking. arXiv:1704.02781 (2017)
28. Leal-Taix, L., Canton-Ferrer, C., Schindler, K.: Learning by tracking: siamese cnn for robust target association. In: CVPR Workshops (2016)
29. Levinkov, E., Uhrig, J., Tang, S., Omran, M., Insafutdinov, E., Kirillov, A., Rother, C., Brox, T., Schiele, B., Andres, B.: Joint graph decomposition & node labeling: Problem, algorithms, applications. In: CVPR (2017)
30. Li, W., Zhao, R., Xiao, T., Wang, X.: Deepreid: Deep filter pairing neural network for person re-identification. In: CVPR (2014)
31. Long, C., Haizhou, A., Zijie, Z., Chong, S.: Real-time multiple people tracking with deeply learned candidate selection and person re-identification. In: ICME (2018)
32. Milan, A., Leal-Taixé, L., Reid, I., Roth, S., Schindler, K.: MOT16: A benchmark for multi-object tracking. arXiv:1603.00831 (2016)
33. Milan, A., Rezatofighi, S.H., Dick, A., Reid, I., Schindler, K.: Online multi-target tracking using recurrent neural networks. In: AAAI (2017)
34. Papageorgiou, D.J., Salpukas, M.R.: The maximum weight independent set problem for data association in multiple hypothesis tracking. Optimization and Cooperative Control Strategies (2009)
35. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: NIPS (2015)
36. Ristani, E., Solera, F., Zou, R., Cucchiara, R., Tomasi, C.: Performance measures and a data set for multi-target, multi-camera tracking. In: ECCV Workshop (2016)
37. Son, J., Baek, M., Cho, M., Han, B.: Multi-object tracking with quadruplet convolutional neural networks. In: CVPR (2017)
38. Tang, S., Andriluka, M., Andres, B., Schiele, B.: Multiple people tracking with lifted multicut and person re-identification. In: CVPR (2017)
39. Tang, S., Andres, B., Andriluka, M., Schiele, B.: Multi-person tracking by multicut and deep matching. In: ECCV (2016)
40. Tang, S., Andres, B., Andriluka, M., Schiele, B.: Multi-person tracking by multicut and deep matching. In: ECCV Workshops (2016)
41. Yang, F., Choi, W., Lin, Y.: Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In: CVPR (2016)
42. Yang, T., Chan, A.B.: Recurrent filter learning for visual tracking. arXiv:1708.03874 (2017)
43. Zheng, L., Shen, L., Tian, L., Wang, S., Wang, J., Tian, Q.: Scalable person re-identification: A benchmark. In: ICCV (2015)