Track-RNN: Joint Detection and Tracking Using Recurrent Neural Networks

Kuan Fang
kuanfang@stanford.edu

Abstract

As deep neural networks revolutionize many fundamental computer vision problems, there have not been many works using neural networks to track objects. In this project, we design and implement a tracking pipeline using convolutional neural networks and recurrent neural networks. Our model can handle detection and tracking jointly using appearance and motion features. We use MOT data challenge as a highly occluded single object tracking dataset. We demonstrate good qualitative and quantitative results of our model and discuss how to extend the pipeline to multi-object tracking.

1 Introduction

We are undergoing a revolutionary era in artificial intelligence and computer vision. Starting from 2012, deep neural network based approaches are making tremendous progress in many fundamental computer vision problems, including image classification, object detection, semantic segmentation and so on. Unlike traditional computer vision algorithms, which relies on using sophisticated hand designed features and building complex algorithm pipelines, these deep learning methods favor an end-to-end learning manner and automatically optimize their weights and features for different tasks given huge amount of training data.

While this neural network techniques have been pushing multiple traditional computer vision tasks towards human-level accuracy, there is still a large gap in using deep learning in motion understanding, especially in tracking, which also has been studied for decades. We need to design mathematic models and algorithms based on the current success to handle these challenges to handle the challenges in tracking.

Compared with detection in static images, tracking is similar but different. Detection is about giving thousands of region proposals in an image, we predict a score of each candidate region proposals, which indicates whether or not there is a desired object in it. And we choose those region proposals with high scores. While for tracking, we need to not only detect the object but also tell if this is the object we have been following in the history. There is single-object tracking, in which case a single target is initialized in the beginning and we try to follow this specific target; and multi-object tracking, where the number of targets is changing though time, we try to detect and track each of the target during the time they are visible in the camera. Multi-object tracking is more challenging, since we need to re-identify the object in different time steps and deal with the situation when the object is occluded by the environment or other objects.

For either tracking problems, there few learning based works published. Most of the previous works use hand designed features such as optical flows and bounding box positions. As there are more and more training data for both of single object tracking and multi-object tracking, it is promising to improve the tracking performance using deep neural network based approaches.

29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

2 Comparison with Previous Works

2.1 Previous works

There are two types of tracking problems: single object tracking and multi-object tracking. For both categories there are many related works.

Among the most successful single object tracking algorithms, there are Kalman filters, Meanshift[4], Camshift[2] and Kanade–Lucas–Tomasi (KLT) feature tracker [8] and so on. None of these algorithms is learning based. Recently, Held et al designed a neural network tracker, but it can hardly be extended to multi-object tracking.

For multi-object tracking, most previous works use tracking by detection scheme. These methods first detect the bounding boxes of objects of specific categories (e.g. cars, people), and then connect the bounding boxes of the same trajectories in different time frames using data association algorithms. Recently, Xiang et al proposed a learning based Markov decision process (MDP) tracker [10] that train multi-object tracker using each of the trajectories as single object tracking data.

2.2 Advantages of Our Model

For multi-object tracking, the state-of-the-art methods [10] predict the tracking trajectories using detection results. First, they use an object detector to get bounding box candidates in each time step without using any temporal information. Next, at each time step t, given the previous tracking history of each target, they match the bounding box candidates with previous tracking trajectoies based on image features and motion priors. The state-of-the-art algorithms use optical-flow based image features and naïve motion priors such as constant velocity models. However, these two-steps tracking by detection methods suffers from errors from the detection stage. And hand designed features are not robust to variants of foreground and background appearance.

In this project, we propose a neural network framework for joint detection and tracking. In our model, the detection is used for initializing a tracking trajectory, while for tracking itself no detection is needed. We start from single-object tracking and eventually aim at multi-object tracking. The tracking frameworks is currently designed for highly occluded single object tracking scenarios but can also easily extended to multi-object tracking. Our contribution will be as follow:

- A joint detection and tracking framework using region convolutional neural networks (R-CNN) [5] and recurrent neural networks (RNN).
- A training procedure that can train our model efficiently. It needs to handle data augmentation and data balance well.
- Our model achieves good performance of detection and tracking. And we show qualitative and quantitative results in the report.

3 Methods

3.1 Summary of the Technical Solution

We formulate the tracking problem as a Markov decision problem, which is inspired by [10]. A tracking target in the video can go through 4 life stages: initialized, tracked, lost, inactive. In this milestone, we mainly focus on the single-object tracking aspect with given initialization. And we aim to extend the model to multi-object tracking in the final report.

Our track-rnn model is mainly composed of two parts: the detection part and the tracking part. The two parts share the convolutional layers in the bottom.

The detection part is a Fast-RCNN [5] model used for initializing a trajectory. When an object is detected, a new trajectory is added to the trajectory list with an initial RNN hidden state computed from the detected bounding box.

The tracking part is composed of a motion prior and a appearance comparison network. The motion prior propose region proposals in the current frame of a trajectory given the selective search results and previous tracking history. While the appearance comparison network outputs a tracking score given

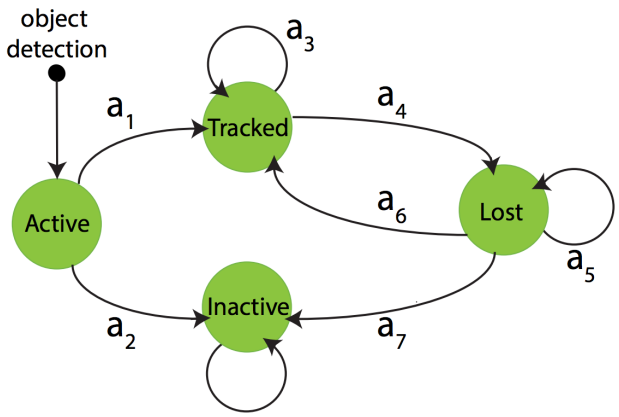

Figure 1: The four life stage of a tracking object.

the region proposal and the tracking history. And we will choose the bounding box corresponding to the largest tracking score in each frame.

3.2 Motion Prior

A motion prior model takes the previous bounding box coordinates (center x, center y, width and height) and predict the most possible bounding box coordinate in the next frame. Currently, we use a constant velocity motion prior of window size 10. This means that the model takes the bounding box coordinates in the most recent 10 frames, compute the average velocity (value changes) and add that in to the box in the most recent frame.

We also consider using RNN motion priors, which we expect to have slightly better prediction accuracy. But due to the uncertainty of human motions, it is unlikely to predict much better than linear assumption.

3.3 Appearance Comparison Network

The appearance comparison network takes a bunch of region proposals and the frame image as input. It predicts the corresponding intersection of union (IOU) between the region proposals and the ground truth bounding box as a confidence score.

On the bottom, we use a RCNN with AlexNet [7] to efficiently extract the image features of each region proposal. The RCNN feeds the image feature into a recurrent neural network on the top. We re-implement the RCNN code, including the ROI pooling layer CUDA code, [5] in Tensorflow [1]. We adopt the caffe-tensorflow toolkit wrote by Saumitro Dasgupta to convert pretrained Caffe models into Tensorflow models.

On the top, we design a novel recurrent neural network (RNN) to utilize the temporal information from previous time steps and image features from the current time step. At each time step, the RNN compute the IOU prediction of each region proposal with the current hidden state. Each region proposals will generate a unique new hidden state vector respectively. We will choose the hidden state vector with the highest predicted IOU score as a candidate to update the hidden state. If the highest predicted score is above 0.5, we will update the hidden state; if none of the predicted score is higher than 0.5, we will assume that the object is occluded and the hidden state will remain unchanged.

In our current implementation, we use GRU [3] with 128 units.

3.4 Track a Single Object

When tracking a single object, we will first predict the most possible bounding box in the current frames and sample 256 region proposals from the pool of selective search results [9]. For each of the region proposals, we predict a IOU score and choose the region proposal with the highest score in the current frame.

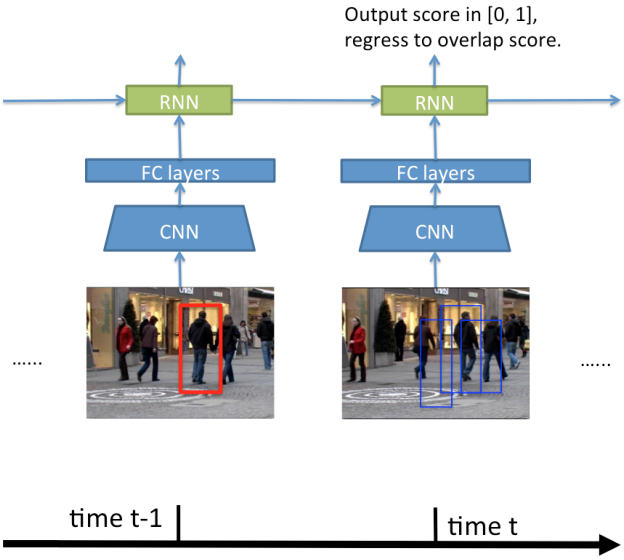

Figure 2: Appearance comparison network.

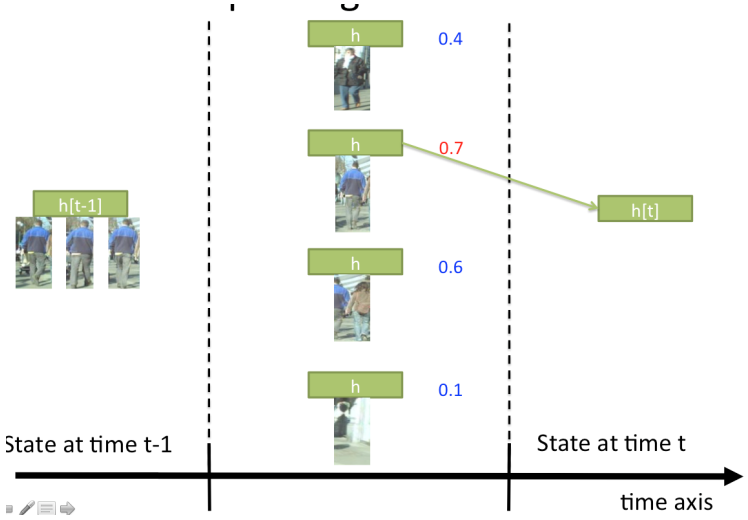

Figure 3: Update RNN hidden state when there is no occlusion.

3.5 The Training Procedure

We train and test our model on the MOT data challenge. We preprocess the training trajectories to remove false annotation and to compute and occlusion labels.

During the training, we first cut off each trajectories into a batch. Then we sample 512 region proposals around the ground truth bounding box in each frame and compute the IOU scores between the region proposals and the ground truth box. The n we feed each of these batches into our neural networks and train for the IOU score regression.

We found that the testing performance can benefit a lot from utilizing longer temporal information in training. We have designed a new training scheme propagating RNN hidden states. The results are on going and we will report the details in the final report.

Comparing with the milestone, we save the conv5 features locally before the training, thus enables bigger batches. We trained in batches of 20 steps, which lead to more robust tracking for long sequence during test time.

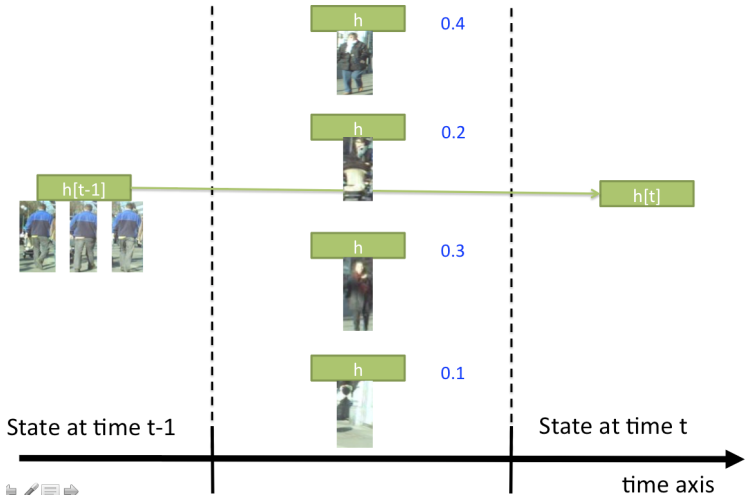

Figure 4: Update RNN hidden state when there is occlusion.

4 Experiments

4.1 Detection Results

Since Fast-RCNN only has Caffe version, we implement the Fast-RCNN [5] Model from scratch using Tensorflow. The link to the code can be found in the paper. We use half of the training videos in MOT dataset as training data and half as the testing data. We train our model using the Adam optimizer [6] with default hyperparameters on a Titan X GPU. Each iteration takes 0.3 second. The training loss went down to 0.0781 after 10 epochs (22130 iterations).

As shown in Fig. 5 is a list of the detection AP score of our model and the default detection results provided by the MOT data challenge website. Our detection performance is 12 to 52 percent better than the default detection. Even in some highly challenging scenes like ETH-Pedcross2 we still have a reasonable AP score of 0.6172.

seq name	split	Default detection in MOT	Ours
TUD-Stadtmitte	train	0.7025	0.8897
TUD-Campus	val	0.5949	0.8121
PETS09-S2L1	train	0.8799	0.9559
ETH-Bahnhof	train	0.5454	0.912
ETH-Sunnyday	val	0.4345	0.9214
ETH-Pedcross2	val	0.1188	0.6172
ADL-Rundle-6	train	0.4883	0.756
ADL-Rundle-8	val	0.3484	0.5754
KITTI-13	train	0.3748	0.8976
KITTI-17	val	0.5295	0.7581
Venice-2	val	0.4276	0.5607

Figure 5: Detection scores.

4.2 Tracking Results

We train the tracking using the same training configurations as that in the detection. We demonstrate qualitative tracking results in Figure 6. The plotted boxes are the region proposals sampled by the motion priors. The color indicates the tracking score: red for high, blue for low and white in the middle. As we can see, our tracker successfully tracks the target object and gives high scores to those bounding boxes of higher overlap when there is no occlusion. When the object is occluded by others,

none of the boxes gives a high enough score. When the object shows up again, the tracker reidentify it by predicting a high tracking score around the correct bounding box.

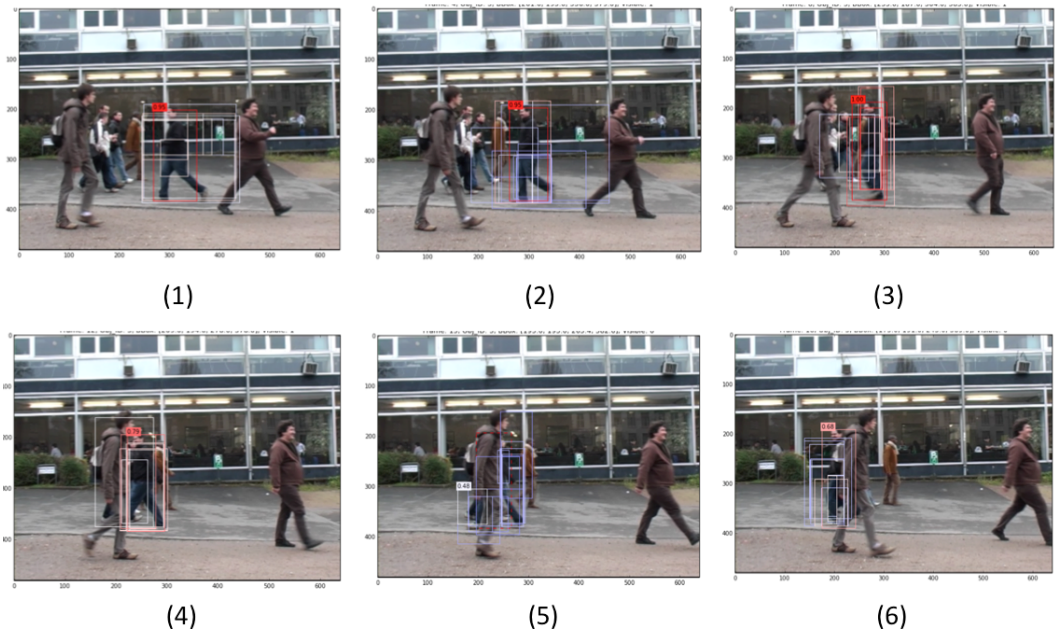

Figure 6: Qualitative tracking results (a).

In Figure 7 there are some earlier results we showed in the milestone. The red box is the ground truth and the green box is the prediction. We can see that the tracker and indicate that the object is invisible when it is occluded and is able to identify it after it shows up again. In this video, a lot of objects have similar appearance, but it is proved that our tracker can still track the object well.

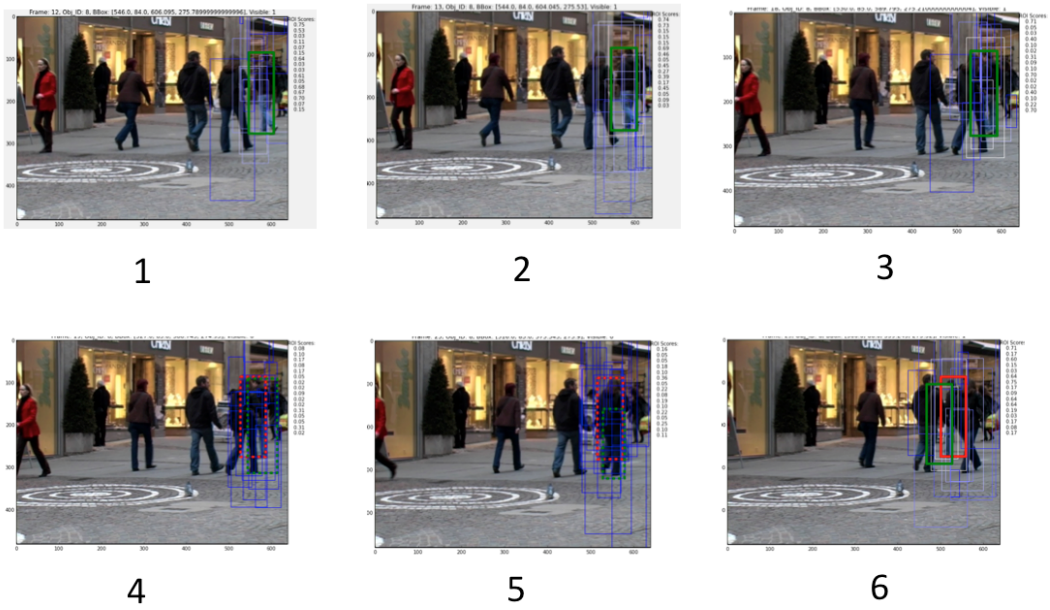

Figure 7: Qualitative tracking results (b).

4.3 Analysis of Failure Cases

However, given the current model, we can still see a lot of failure cases. As shown in Fig. 8, the target object and the occlusion object have similar appearance (brown jacket). After the target object is occluded, the tracker wrongly switch to the occlusion. In the future, we will consider add motion features to prevent this kind of failure. Because in this case, although these two objects have similar appearance, their positions and box size are different, which can be used to distinguish them.

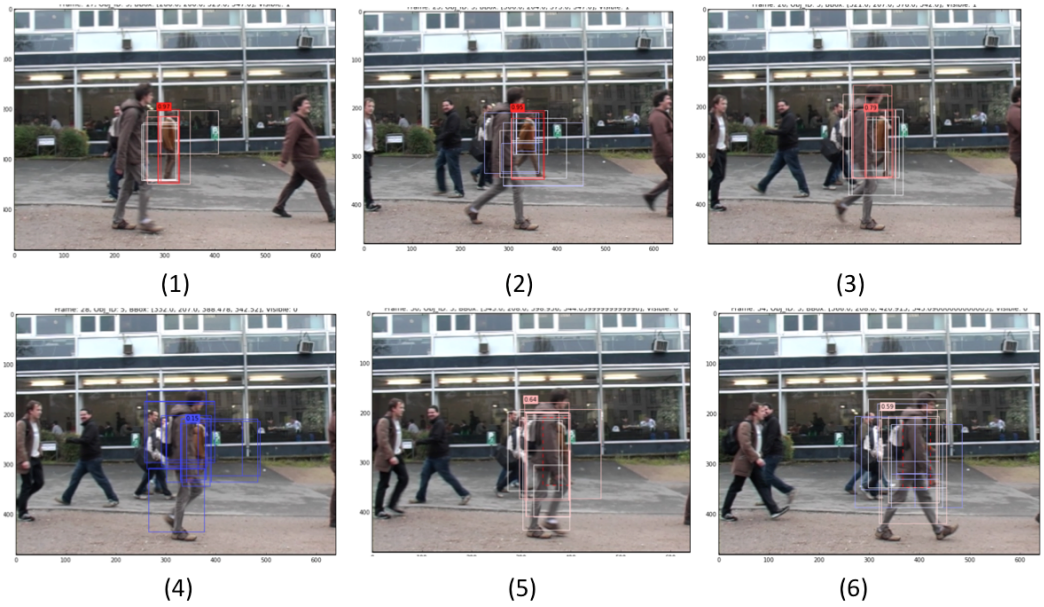

Figure 8: Failure cases.

5 Conclusion

We design and implement a model for joint detection and tracking. Based on the Markov decision process, we combine the Fast-RCNN with a novel recurrent neural network architecture. Our model achieves good performance of detection and tracking. And it can be easily extend to multi-object tracking as elaborated in the paper.

The dropbox link to the code of this project is:

`https://www.dropbox.com/s/xwt674mro2e4et9/cs231a_project_code.zip?dl=0`

If there is any problem to get the code, please feel free to send me an email.

Since this project is related to an ongoing unpublished research, I wish the TAs can keep this report and the code private. Thank you so much.

6 Future Works

In the future, we will combine the detection and tracking parts together and evaluate it on the MOT benchmark. We also expect to include motion features as RNN inputs to boost the performance. Furthermore, it would be interesting to consider formulating the training procedure as a reinforcement learning procedure.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey

Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] John G. Allen, Richard Y. D. Xu, and Jesse S. Jin. Object tracking using camshift algorithm and multiple quantized feature spaces. In *Proceedings of the Pan-Sydney Area Workshop on Visual Information Processing*, VIP '05, pages 3–7, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.

[3] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[4] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, May 2002.

[5] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition*, 2014.

[6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[7] 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.

[8] Carlo Tomasi and Takeo Kanade. *Detection and tracking of point features*. School of Computer Science, Carnegie Mellon Univ. Pittsburgh, 1991.

[9] J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, and A.W.M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 2013.

[10] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to track: Online multi-object tracking by decision making. In *International Conference on Computer Vision (ICCV)*, pages 4705–4713, 2015.