Video Object Tracking Using Neural Networks

Mohammed Gasmallah

10042860

11mhg

Project Type: Thesis

Supervisor: Professor Farhana Zulkernine

Date: April 10th, , 2018

Abstract:

In this paper, we have implemented a novel video object detector and tracker using the YOLOv2 architecture as a basis for the network. With the rise of video datasets and self-driving cars, many industries seek a way to perform quick object detection on video, as well as perform predictive tracking on these objects. The proposed video object detector and tracker is in fact a predictive object detector and tracker, allowing for object prediction and information to be known before the event takes place. This network implements the novel convolutional 2D LSTM layer presented in [1] . Two different approaches were taken to implement this network. The first is an intuitively easy to understand post-temporal pattern matching attempt wherein the YOLOv2 detector is used to detect objects in two images and the novel convolutional 2D LSTM layer performs it's temporal feature mapping across the output tensors of the detectors. The second approach, although less intuitive, provides better results by performing the temporal feature mapping first across two images and feeding the output into the YOLOv2 detector that has been partially wrapped using a Time Distributed layer. Finally, the network was tested on the MOT 2017 dataset and although did not perform competitively against state-of-the-art techniques, the network was able to perform predictive object detection and tracking, demonstrating that the convolutional 2D LSTM layer is useful for a variety of video analysis problems. Future work in this field proves promising for video analysis and predictive object detection in general.

Table of Contents

Chapter 1: Introduction

1.1. **Motivation**

Image object recognition and object tracking are fundamentally important problems in computer vision. Image object recognition has many interesting solutions within the field of neural networks. With the advent of newer, more complete image datasets interesting questions regarding the use and applications of object detection and object tracking have arisen. The difficulty in many video object tracking algorithms is due to the fact that a singular algorithm often has to deal with a multitude of different obstacles[1]. More recently, using neural networks to perform object detection and object tracking has become more and more feasible and many industry applications have become apparent. Using object detection and tracking allows for video analysis to determine paths that objects take and how they change throughout that path. Self-driving cars should track pedestrians, surveillance cameras should track suspect objects, drones should track people and determine possible paths along a route.

Unfortunately, state-of-the-art methods are not exceptionally fast, often forcing current applications to perform on incomplete information. These traditional visual tracking methods often must deal with dimensionality and loss of local structural information problems. Due to the speed requirement of online visual object tracking, most methods involve the use of kernels [2], [3][4]. These often require kernel weight updates in real time, unfortunately, this requires heavy computation to perform quickly[4].

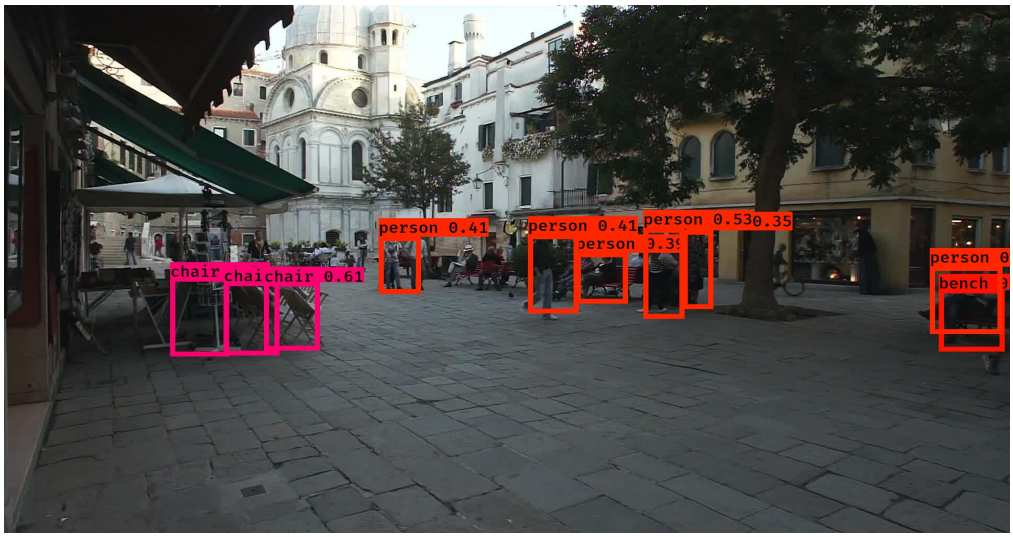

Fig 1.1 Sample YOLO detection on the MOT dataset.[5]

When a human drives a car, they are able to determine where a pedestrian, vehicle or other object is moving and how fast they are moving. This allows them to, in a difficult scenario, make

quick decisions based on this information. For drones and self-driving cars, having the ability to track and predict motion of objects allows them to make the important decisions when required. This predictive capability is something that is often lacking in object detectors and the temporal pattern matching is often missing in the current object trackers.

1.2. **Key contributions**:

In this paper, we propose the use of a convolutional 2D LSTM layer as a spatio-temporal pattern matching layer in order to perform predictive object detection/tracking. Many current object detection and tracking networks lose the temporal information that occurs in the use of purely image-based object detection. The ability to have visual memory of particular video sequences, allows for spatial and temporal based object detection and tracking[6]. By adding a memory module, using a 2 dimensional convolutional network composed of LSTM (long-short term memory) nodes, the network learns to remember particular motion patterns and possible changes in height or width. This allows the network to use both spatial awareness (through the convolutional network) and temporal awareness. Finally, the temporal pattern matching allows for memory of possible classifications of bounding boxes. When the network classifies a bounding box as being a “person”, it remembers that it has done so and requires stronger features to classify it as anything else.

This convolutional 2D LSTM was first introduced in the now-forecasting paper as it allows for interesting and useful spatial and temporal pattern matching[1]. This novel layer was proposed in order to deal with spatio-temporal sequence pattern matching, which relates thoroughly to the network tracker implemented in this paper. This make sense for use in the predictive object tracker and detector proposed as it allows for sequence-to-sequence learning of the dataset. What this network shows is that predictive object detection and tracking benefits from this novel layer and further research using this layer could lead to valuable breakthroughs in predictive image processing.

1.3. **Layout**:

In this paper, we will discuss current implementations of object detectors and trackers, as well as current trends in image processing with regards to neural networks in chapter 2. We will continue by discussing the actual implementation of the network as it currently stands and the results that were received by training the network in chapter 3. A comparison against the YOLOv2 Detector is done in order to demonstrate the effectiveness of the network against state-of-the-art approaches. Finally,

chapter 4 will discuss and summarize the novel aspects of the network as well as discuss the future works required to further the networks accuracy and precision.

Chapter 2: Background

2.1. YOLO9000

One of the most interesting object detectors in the field is known as the You-Only-Look-Once 9000 detector (YOLO9000). This is the network that inspired this paper and revolutionized the field by introducing a state-of-the-art, real time object detector capable of detecting over 9000 classes from ImageNet[7] . This object detector is a series of 22 blocks of convolutional, max pooling, and leakyReLU layers. A skip layer is attached on the 20^{th} block, concatenating a fine and coarse series of feature maps.

The object detector they have implemented uses a more robust training method wherein images are preprocessed and augmented. Furthermore, both images with and without bounding boxes are passed in. For those with bounding boxes, a full backpropagation is performed, but if an image with no bounding box and only classification is seen, simple classification only backpropagation is done. This allows the network to learn a very large number of classes and closes the boundary between object detection and image classification[7].

2.2. Tracking Papers

Single object tracking often requires the previous frame and the current frame, as well bounding box information on the previous frame[8]. They are able to track novel objects in images and estimate their movements and variance. One of these networks is described in "*Learning to Track at 100 FPS with Deep Regression Networks*". This network learns motion smoothness by making certain that the center of bounding boxes are described as relative to the previous frame giving a delta x and delta y value. In a similar situation, they model changes in scale using the previous frame and a gamma width and gamma height to describe the change[8].

Most object trackers focus on the definition of tracking as "*.. the analysis of video sequences for the purpose of establishing the location of the target over a sequence of frames ..." [3]*. This often starts with the bounding box on the initial frame. This unfortunately is not always the case in modern

day computer vision analysis. We are often in a situation where time is sensitive and performing object tracking without the initial bounding box is critical to the situation.

Other trackers, such as the one from [9] focus on the problem from a practical surveillance and activity tracking perspective. These trackers look at distributed consensus networks that perform activity recognition and tracking from a multitude of cameras. The method used is a non-neural network method, which focuses on the Kalman-Consensus algorithm and look at a variety of different parameters to both infer and track the activity[9].

It can be seen from [3], [10]–[12] that there are certain features of a neural network tracker and object detector that are extremely important. The ability for the network to generalize well, handling occlusion and reflections, motion blur and transfer training are all extremely important in a good object tracker/detector.

2.3. Datasets and Transfer Learning

Some of the most popular datasets for object detection are the MS COCO, PASCAL VOC, and for object tracking the MOT dataset. Unfortunately, the number of good video tracking datasets is small in comparison to the amount of object detection datasets. As such, [13] demonstrated that these networks are able to be fully trained on synthesized virtual data and scale well to real world data. In fact, [13] trained a network entirely on virtual data and performed competitively against state-of-the-art networks on modern datasets.

Chapter 3: Implementation Details

3.1. Overview

There are two extremely well known approaches to solving the problem of object detection. The first involves the use of a sliding window that allows the network to deal with particular elements of the image and associate a confidence and a class score to each window. This method is commonly used by nets such as the R-CNN or the faster R-CNN.[12][3] Unfortunately, these methods are still quite slow in comparison to the method used by YOLO and YOLO9000. These networks use an anchor based method that allows the network to predict over the whole image using grids to segment specific parts of the image. An image-net or VGG16 style network is used to extract features out of the image and outputs a specific (GRID_HEIGHT,GRID_WIDTH, _) shape tensor (in this case YOLOv2 feature extractor was used). This is then fed to a convolutional neural network with outputs that relate to specifics of the results ([anchors,x,y,delta x, delta y],[confidence],[classes]).[7], [14] A custom loss function is used to judge the networks ability to predict bounding boxes and class. This loss function is discussed in detail in YOLOv2 and YOLO9000[7].

$$\begin{aligned}
&\lambda_{\mathbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{ij}^{\text{obj}} \left[\left(x_i - \hat{x}_i \right)^2 + \left(y_i - \hat{y}_i \right)^2 \right] \\
&+ \lambda_{\mathbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} L_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} L_i^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2
\end{aligned}$$

Fig. 3.1 Custom Loss Function defined by YOLO9000 and YOLOv2[7]

This initial implementation used an object detection network that is trained separately and used as part of a tracker network that, using the same loss function, trains the convolutional 2D LSTM and final convolutional layers to output bounding boxes and classes. The second implementation of the object tracker, as it changes the object detection network (from YOLO) by wrapping it in a Time Distributed wrapper and adds a Convolutional 2D LSTM, must be trained anew and weights cannot be initialized to the initial YOLO weights.

The YOLO loss function is extremely important to the network as it manages to evaluate a variety of different aspects of the problem all at once. Constants can be changed to mark particular

elements more important than others. The network is evaluated on the loss from the regression of the x,y box coordinates, width and height values, as well as the loss from classifying objects, the loss from classifying wrong objects and a final overall classification loss. The total of this value corresponds to the loss of the network[7]. The network performs non-max suppression in order to stop overlapping predictions of the same class. This means that a maximum number of boxes can be predicted. For the sake of computational efficiency, a maximum of ten boxes are chosen as the final predictions.

3.2. Data:

Some of the most common datasets being used in video tracking and object detection are the COCO dataset and the MOT benchmark dataset. The COCO dataset is a large-scale object detection, segmentation, and captioning dataset with several features.[15] There are 80 object categories and a multitude of different annotation styles. This dataset was collected by gathering images of complex everyday scenes containing common objects. The 2017 Validation and Test images were the images used for classification training of the network. Some preprocessing is done on the dataset to select the appropriate number of objects per image. A maximum of ten objects are chosen per image and annotated appropriately. Anchors are created using k-means clustering to determine the aspect ratio of 5 anchor boxes that fit most of the dataset and the average IOU that it produces.[7] This is important so that when the network predicts a bounding box, it selects the type of anchor box, the location for the anchor box and the difference in width and difference in height for the box.

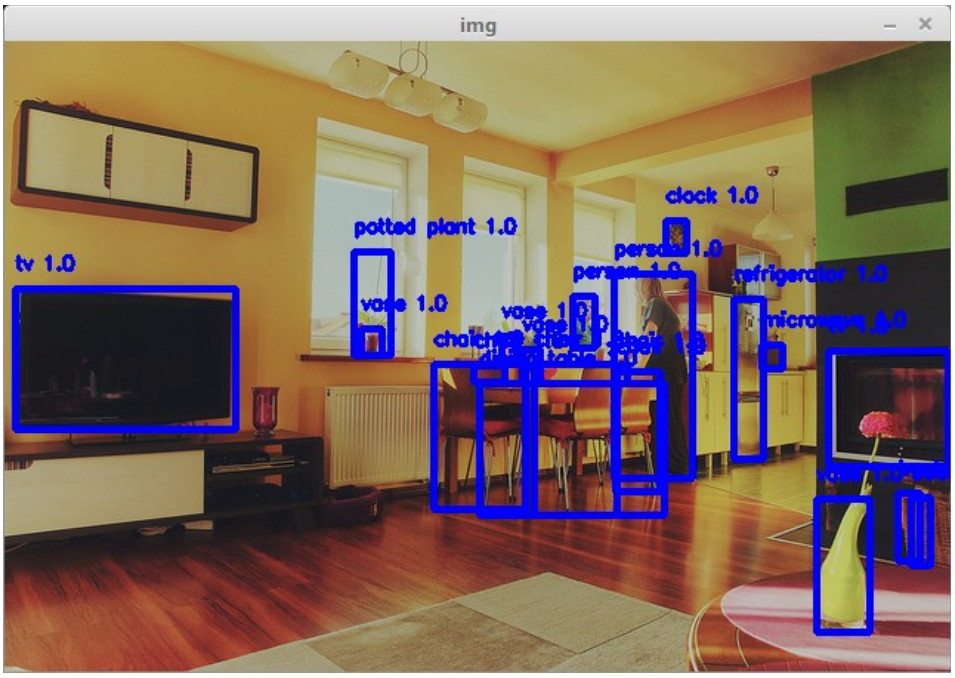

Fig. 3.2. Coco dataset annotations and bounding boxes on sample image

The MOT 2017 dataset is being used for the video object tracking component of the network. This dataset is a large collection of video frames from a variety of different types of scenes. Each frame is laboriously detailed with bounding boxes and label annotations, as well as unique IDs pertaining to the unique path of every object. This dataset is important as it also comes with a crucial standardized benchmark, leaderboards and evaluation techniques.

3.3. Program:

The most commonly used program in neural networks is the Tensorflow backend for python. It is extremely popular and has been used by a multitude of different networks and papers. As a high level interface to Tensorflow, Keras was used to allow for quick and simple prototyping. These both have pythonic implementations. Tensorboard was used for loss visualization and model architecture confirmation. We are also using YAD2K (Yet Another Darknet 2 Keras), to allow for the use of the YOLO v2 network for the detectors in the first implementation. This allows for weight transformation from Darknet to Keras, which simplifies the training of the tracker, as then we are certain that the detectors work.

3.4. Implementation Details:

For the initial implementation of this network, a base feature extractor is important. This feature extractor is common and present in many popular object detection networks. The feature extractor is a series of 2D convolutional layers, sometimes followed by batch normalization, then a LeakyReLU and a max pooling layer. A Skip connector connects one of the earlier layers of the feature extractor with a later layer. This is thoroughly described in the YOLO9000 paper, and an in-depth, layer by layer, explanation of the final predictive tracker implementation can be found in Appendix A.

The YOLOv2 object detector is implemented in Keras and Tensorflow and the official weights from darknet were ported to make sure the network was trained and operation to state-of-the-art performance. Two frames are passed into the network in a sequence generator and must pass through their own detectors. The output tensor from the two detectors are concatenated and reshaped to be able to pass through a convolutional 2D LSTM layer and finally to a convolutional 2D layer. Although this implementation is interesting as it allows for a clearer understanding of the intuition behind the network, it does not perform as well as the second implementation of the network.

The second implementation of this network uses the base feature extractor from YOLOv2 but performs a convolutional 2D LSTM initially before being fed into the a series of Time Distributed wrapped YOLOv2 extractor layers. The output from this network is closer in architecture to the initial YOLOv2 implementation but the temporal pattern matching is front-loaded and distributed along the network. This is the final implementation of the tracker that was used.

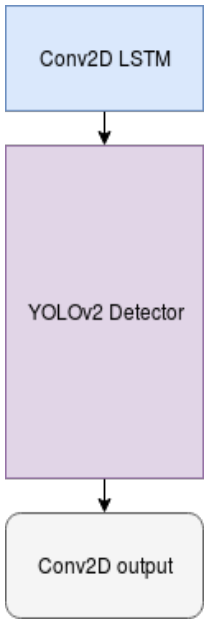

Fig.3.3. Architecture for implementation 2 of predictive object tracker

Certain parameters are chosen for the loss function and for the network in general that allow the network to perform differently. The weight decay and momentum are chosen from [7] as they have been shown to aid in converging quickly. The object scale loss is kept at five times the scale of the other losses in order to punish predicting objects over all other aspects of the network. If this number is made to be lower, the network tends to predict a multitude of false boxes all over the image.

3.5. Validation:

Validation is done post training and uses the COCO dataset as well as the MOT dataset. The validation requires a prediction of bounding boxes for each image and a IOU (Intersection of Union) calculation against the ground truth. The intersection of union is a particular evaluation metric that is used to measure the accuracy of an object detector on a particular dataset. This is often done by dividing the area of overlap by the area of the union. This calculation can be seen as the percentage of the ground truth that is encompassed within the predicted bounding box (*fig3.4)*. An additional validation method is used called the mAP (mean Average Precision). This is a very commonly used method amongst object detectors and there are a variety of different values that can be used to compare the network against other networks.

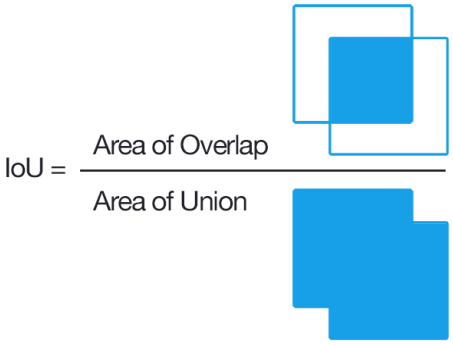

Fig. 3.4. IOU measurement figure

One issue with using the MS COCO dataset for validation is that this dataset is well know for producing lower mAP compared to many other datasets. The COCO dataset achieves a 48.1% mAP in YOLO, but YOLO achieves 63.4% mAP in VOC 2007[14]. These numbers however are still better if not similar to those by conventional state-of-the-art methods[15].

Our trained implementation of the YOLOv2 network trained on the MOT dataset performed at 65.6% mAP on the MOT 2017 dataset. The first implementation of the predictive tracker yielded a 43.4% mAP on the MOT 2017 dataset. The second implementation yielded a much better 54.9% mAP on the MOT 2017 dataset (over a 10% increase in mAP). This number is low compared to many other object trackers, however it should be noted that other trackers require initial bounding boxes to be fed into the network[3], [8]. As they do not perform object detection (because the bounding boxes are initially fed into the network), it is expected that our network would perform worse on complicated object tracking datasets such as MOT 2017.

Fig. 3.5. Frame from Predictive Tracker Demo

Fig. 3.6. Example frame from Tracker

3.6. Discussion:

The initial implementation yielded a rather low mAP on the MOT dataset, while the second implementation yielded over a 10% increase in mAP compared to the initial implementation. Our intuition for this increase in mAP has to do with something discussed in [11] . They discuss the idea of generating in-domain training data by synthesizing plausible future video frames. The predictive detector/tracker does not synthesize plausible future video frames. However the convolutional 2D LSTM layer allows the network to develop interesting feature maps that incorporate temporal patterns amongst the spatial features. This allows for the network to perform it's detection on the temporal and spatial patterns from the beginning, rather than attempting to finely separate them at the end.

Regardless, both networks perform worse than state-of-the-art methods for object tracking. The reason for this seems to be the fact that the predictive tracker is entirely predictive, unlike the other object trackers. Object trackers such as those detailed in [11], [16] require initial bounding boxes in order to perform object tracking. As such, they perform object assignment inference and object velocity inference. The predictive tracker must do object assignment inference, object velocity inference, classification, and object detection. Each of these steps introduces possible loss in accuracy and precision and compounds over time.

On top of all this, the predictive tracker has a maximum number of boxes to predict of ten. Most of the MOT dataset has a much larger number of bounding boxes to track. By increasing the number of bounding boxes however, we introduce a greater possibility of error in object detection. This can be mitigated slightly by performing some confidence score thresholding, IOU thresholding and increasing the maximum boxes to predict. This comes to us at the cost of increasing computational requirements and increasing dataset size requirements.

To handle this problem and make the network perform better, fine-tuning, and separating predictive object detection and object tracking should be done. Fine-tuning has been shown to provide excellent results on neural networks performing object tracking[13] and the use of large virtual datasets for initial training would allow for quick transfer learning. Finally, separating predictive object detection and object tracking will allow the network to focus on a particular problem while achieving parallelism, hence increasing the online speed of the network.

Chapter 4: Conclusion and Future Work

4.1. Future Work:

For future works, the use of object relative velocity and object relative width and height changes might be used in the stead of absolute x and y values. This would allow the network to focus on object velocity prediction. Secondly, having a specialized predictive object detector and a dedicated object tracker would allow each network to focus on different aspects of the problem. Dividing the problem into sections and combining them into a final product should increase performance and accuracy of the networks. Finally, some changes should be made with regards to the amount of images that can be fed into the network (perhaps even a full video). This would allow the network the ability to use more information than is often found in two simple images. An implementation similar to [4] would work quite well for feeding a multitude of frames, if not a whole video into the network. This may even be expanded with video super resolution, allowing the network to either synthesize future video frames or synthesize better video frames (with motion blur, occlusion and lighting invariance).

4.2. Summary:

In this paper, we have proposed a predictive object tracker that performs predictive object detection and classification through the use of the previous two frames. It uses the architecture from YOLOv2 paper and expands upon it using the novel convolutional LSTM 2D layer. This layer, having only been used in predictive now-forecasting, has demonstrated it’s ability to learn sequence-to-sequence object detection and tracking through prediction. Although the network under performs compared to state-of-the-art methods, it provides insight on the development of the field in terms of the predictive capabilities of these networks.

The use of the MOT dataset for video object detection was useful in terms of benchmarking and demonstrating the capabilities of the video object tracker, particularly in comparison to the YOLOv2 detector. Regardless, the future of video object detection and tracking is bright. With advances in neural network libraries, new and novel convolutional implementations and increasing research in the field, video and image processing is increasing in importance in the field of computer vision.

Works Cited

[1] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W. Wong, and W. Woo, “Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting,” pp. 1–12, 2015.

[2] Q. Wang and J. Liu, “Visual Tracking Using the Kernel Based Particle Filter and Color Distribution,” Zhejiang University, 2005.

[3] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, “Visual tracking: An experimental survey,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 7, pp. 1442–1468, 2014.

[4] K. Kang, W. Ouyang, H. Li, and X. Wang, “Object Detection from Video Tubelets with Convolutional Neural Networks,” The Chinese University of Hong Kong, 2016.

[5] I. Reid, S. Roth, K. Schindler, A. Milan, and L. Leal-taix, “MOT16 : A Benchmark for Multi-Object Tracking,” pp. 1–12.

[6] P. Tokmakov, K. Alahari, and C. Schmid, “Learning Video Object Segmentation with Visual Memory,” *Iccv*, pp. 4481–4490, 2017.

[7] J. Redmon and A. Farhadi, “YOLO9000: Better, Faster, Stronger,” University of Washington, 2016.

[8] D. Held, S. Thrun, and S. Savarese, “Learning to track at 100 FPS with deep regression networks,” Stanford University, 2016.

[9] B. Song, A. T. Kamal, C. Soto, C. Ding, J. A. Farrell, and A. K. Roy-Chowdhury, “Tracking and activity recognition through consensus in distributed camera networks,” *IEEE Trans. Image Process.*, vol. 19, no. 10, pp. 2564–2579, 2010.

[10] I. Posner and P. Ondruska, “Deep Tracking: Seeing Beyond Seeing Using Recurrent Neural Networks,” University of Oxford, 2016.

[11] A. Khoreva, R. Benenson, E. Ilg, T. Brox, and B. Schiele, “Lucid Data Dreaming for Multiple Object Tracking,” pp. 1–16, 2017.

[12] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. Van Den Hengel, “A Survey of Appearance Models in Visual Object Tracking,” *ACM Trans. Intell. Syst. Technol.*, vol. 4, no. 4, p. 58:1–58:48, 2013.

[13] E. Bochinski, V. Eiselein, and T. Sikora, “Training a Convolutional Neural Network for Multi-Class Object Detection Using Solely Virtual World Data,” Technical University of Berlin, 2016.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” 2015.

[15] T. Y. Lin *et al.*, “Microsoft COCO: Common objects in context,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8693 LNCS, no. PART 5, pp. 740–755, 2014.

[16] L. Leal-Taixé, A. Milan, K. Schindler, D. Cremers, I. Reid, and S. Roth, “Tracking the Trackers: An Analysis of the State of the Art in Multiple Object Tracking,” Technical University Munich, 2017.

Appendix A

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 2, 608, 608,	0	
conv_lst_m2d_1 (ConvLSTM2D)	(None, 608, 608, 32)	40320	input_1[0][0]
time_distributed_1 (TimeDistrib	(None, 608, 608, 32)	128	conv_lst_m2d_1[0][0]
leaky_re_lu_1 (LeakyReLU)	(None, 608, 608, 32)	0	time_distributed_1[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 304, 304, 32)	0	leaky_re_lu_1[0][0]
conv2d_1 (Conv2D)	(None, 304, 304, 64)	18432	max_pooling2d_1[0][0]
time_distributed_2 (TimeDistrib	(None, 304, 304, 64)	256	conv2d_1[0][0]
leaky_re_lu_2 (LeakyReLU)	(None, 304, 304, 64)	0	time_distributed_2[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 152, 152, 64)	0	leaky_re_lu_2[0][0]
conv2d_2 (Conv2D)	(None, 152, 152, 128	73728	max_pooling2d_2[0][0]
time_distributed_3 (TimeDistrib	(None, 152, 152, 128	512	conv2d_2[0][0]
leaky_re_lu_3 (LeakyReLU)	(None, 152, 152, 128	0	time_distributed_3[0][0]
conv2d_3 (Conv2D)	(None, 152, 152, 64)	8192	leaky_re_lu_3[0][0]
time_distributed_4 (TimeDistrib	(None, 152, 152, 64)	256	conv2d_3[0][0]
leaky_re_lu_4 (LeakyReLU)	(None, 152, 152, 64)	0	time_distributed_4[0][0]
conv2d_4 (Conv2D)	(None, 152, 152, 128	73728	leaky_re_lu_4[0][0]
time_distributed_5 (TimeDistrib	(None, 152, 152, 128	512	conv2d_4[0][0]
leaky_re_lu_5 (LeakyReLU)	(None, 152, 152, 128	0	time_distributed_5[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 76, 76, 128)	0	leaky_re_lu_5[0][0]
conv2d_5 (Conv2D)	(None, 76, 76, 256)	294912	max_pooling2d_3[0][0]
time_distributed_6 (TimeDistrib	(None, 76, 76, 256)	1024	conv2d_5[0][0]
leaky_re_lu_6 (LeakyReLU)	(None, 76, 76, 256)	0	time_distributed_6[0][0]
conv2d_6 (Conv2D)	(None, 76, 76, 128)	32768	leaky_re_lu_6[0][0]
time_distributed_7 (TimeDistrib	(None, 76, 76, 128)	512	conv2d_6[0][0]
leaky_re_lu_7 (LeakyReLU)	(None, 76, 76, 128)	0	time_distributed_7[0][0]
conv2d_7 (Conv2D)	(None, 76, 76, 256)	294912	leaky_re_lu_7[0][0]
time_distributed_8 (TimeDistrib	(None, 76, 76, 256)	1024	conv2d_7[0][0]
leaky_re_lu_8 (LeakyReLU)	(None, 76, 76, 256)	0	time_distributed_8[0][0]
max_pooling2d_4 (MaxPooling2D)	(None, 38, 38, 256)	0	leaky_re_lu_8[0][0]
conv2d_8 (Conv2D)	(None, 38, 38, 512)	1179648	max_pooling2d_4[0][0]
time_distributed_9 (TimeDistrib	(None, 38, 38, 512)	2048	conv2d_8[0][0]
leaky_re_lu_9 (LeakyReLU)	(None, 38, 38, 512)	0	time_distributed_9[0][0]
conv2d_9 (Conv2D)	(None, 38, 38, 256)	131072	leaky_re_lu_9[0][0]
time_distributed_10 (TimeDistri	(None, 38, 38, 256)	1024	conv2d_9[0][0]
leaky_re_lu_10 (LeakyReLU)	(None, 38, 38, 256)	0	time_distributed_10[0][0]
conv2d_10 (Conv2D)	(None, 38, 38, 512)	1179648	leaky_re_lu_10[0][0]
time_distributed_11 (TimeDistri	(None, 38, 38, 512)	2048	conv2d_10[0][0]
leaky_re_lu_11 (LeakyReLU)	(None, 38, 38, 512)	0	time_distributed_11[0][0]
conv2d_11 (Conv2D)	(None, 38, 38, 256)	131072	leaky_re_lu_11[0][0]

```
time_distributed_12 (TimeDistri (None, 38, 38, 256)  1024        conv2d_11[0][0]
leaky_re_lu_12 (LeakyReLU)      (None, 38, 38, 256)  0           time_distributed_12[0][0]
conv2d_12 (Conv2D)              (None, 38, 38, 512)  1179648     leaky_re_lu_12[0][0]
time_distributed_13 (TimeDistri (None, 38, 38, 512)  2048        conv2d_12[0][0]
leaky_re_lu_13 (LeakyReLU)      (None, 38, 38, 512)  0           time_distributed_13[0][0]
max_pooling2d_5 (MaxPooling2D)  (None, 19, 19, 512)  0           leaky_re_lu_13[0][0]
conv2d_13 (Conv2D)              (None, 19, 19, 1024) 4718592     max_pooling2d_5[0][0]
time_distributed_14 (TimeDistri (None, 19, 19, 1024) 4096        conv2d_13[0][0]
leaky_re_lu_14 (LeakyReLU)      (None, 19, 19, 1024) 0           time_distributed_14[0][0]
conv2d_14 (Conv2D)              (None, 19, 19, 512)  524288      leaky_re_lu_14[0][0]
time_distributed_15 (TimeDistri (None, 19, 19, 512)  2048        conv2d_14[0][0]
leaky_re_lu_15 (LeakyReLU)      (None, 19, 19, 512)  0           time_distributed_15[0][0]
conv2d_15 (Conv2D)              (None, 19, 19, 1024) 4718592     leaky_re_lu_15[0][0]
time_distributed_16 (TimeDistri (None, 19, 19, 1024) 4096        conv2d_15[0][0]
leaky_re_lu_16 (LeakyReLU)      (None, 19, 19, 1024) 0           time_distributed_16[0][0]
conv2d_16 (Conv2D)              (None, 19, 19, 512)  524288      leaky_re_lu_16[0][0]
time_distributed_17 (TimeDistri (None, 19, 19, 512)  2048        conv2d_16[0][0]
leaky_re_lu_17 (LeakyReLU)      (None, 19, 19, 512)  0           time_distributed_17[0][0]
conv2d_17 (Conv2D)              (None, 19, 19, 1024) 4718592     leaky_re_lu_17[0][0]
time_distributed_18 (TimeDistri (None, 19, 19, 1024) 4096        conv2d_17[0][0]
leaky_re_lu_18 (LeakyReLU)      (None, 19, 19, 1024) 0           time_distributed_18[0][0]
conv2d_18 (Conv2D)              (None, 19, 19, 1024) 9437184     leaky_re_lu_18[0][0]
time_distributed_19 (TimeDistri (None, 19, 19, 1024) 4096        conv2d_18[0][0]
conv2d_20 (Conv2D)              (None, 38, 38, 64)   32768       leaky_re_lu_13[0][0]
leaky_re_lu_19 (LeakyReLU)      (None, 19, 19, 1024) 0           time_distributed_19[0][0]
time_distributed_21 (TimeDistri (None, 38, 38, 64)   256         conv2d_20[0][0]
conv2d_19 (Conv2D)              (None, 19, 19, 1024) 9437184     leaky_re_lu_19[0][0]
leaky_re_lu_21 (LeakyReLU)      (None, 38, 38, 64)   0           time_distributed_21[0][0]
time_distributed_20 (TimeDistri (None, 19, 19, 1024) 4096        conv2d_19[0][0]
space_to_depth (Lambda)         (None, 19, 19, 256)  0           leaky_re_lu_21[0][0]
leaky_re_lu_20 (LeakyReLU)      (None, 19, 19, 1024) 0           time_distributed_20[0][0]
concatenate_1 (Concatenate)     (None, 19, 19, 1280) 0           space_to_depth[0][0]
                                                                 leaky_re_lu_20[0][0]
conv2d_21 (Conv2D)              (None, 19, 19, 1024) 11796480    concatenate_1[0][0]
time_distributed_22 (TimeDistri (None, 19, 19, 1024) 4096        conv2d_21[0][0]
leaky_re_lu_22 (LeakyReLU)      (None, 19, 19, 1024) 0           time_distributed_22[0][0]
conv2d_22 (Conv2D)              (None, 19, 19, 425)  435625      leaky_re_lu_22[0][0]
==================================================================================================
Total params: 51,023,017
Trainable params: 51,002,345
Non-trainable params: 20,672
```