Video Predictive Object Detector

Mohammed Hamada Gasmallah
School of Computing
Queen's University
Kingston, Canada
11mhg@queensu.ca

Farhana Zulkernine
School of Computing
Queen's University
Kingston, Canada
farhana@cs.queensu.ca

***Abstract*—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. We propose a predictive video object detector (POD net) integrating the You Only Look Once v2 (YOLOv2) framework with the convolutional 2-dimensional (2D) Long Short Term Memory (LSTM) model proposed by Shi et al.. Our POD net performs object detection using YOLOv2 and object prediction using the LSTM model in an iterative manner with a view to improve object detection in video streams via object prediction. In this study we present two different approaches that we implemented to predict objects in subsequent video clips. The first approach, PODv1, applies a post-temporal pattern matching mechanism wherein the YOLOv2 detector is used to detect objects in multiple images and the LSTM layer is used to perform temporal feature mapping across the output tensors of the detectors. The second approach, PODv2, provides better results by applying the temporal feature mapping first across the images and then feeding the output into the YOLOv2 detector which is wrapped using a Time Distributed layer. We tested POD net on the Multi-Object Tracking (MOT) 2017 dataset and the network was able to perform predictive object detection and tracking, demonstrating that the LSTM layer is useful for a variety of video analysis problems.**

***Index Terms*—Computer Vision, Object Detection, Predictive Detector**

I. Introduction

Image object recognition and object tracking are fundamentally important problems in computer vision [2, 3]. Image object recognition has many interesting solutions within the field of neural networks [4, 5, 6]. With the advent of newer more complete image datasets, interesting questions regarding the use and applications of object detection and object tracking have arisen [7, 8]. The difficulty in many video object tracking algorithms is due to the fact that a single 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 [3, 9]. Using object detection and tracking allows for video analysis to determine and track changes in paths that objects take [10]. Self-driving cars should track pedestrians, surveillance cameras should track suspicious objects, drones should track people and determine possible paths along a route [11, 12].

Unfortunately, the 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 [13, 10, 14, 15] where kernel weights need to be updated in real time. This requires both fast and heavy computations [10]. Neural networks in such cases have shown to provide better performance and generalizability.

Many current object detection networks such as the You Only Look Once network (YOLO) [16, 4], lose the temporal information that occurs in the use of purely singular image based object detection. The ability to have visual memory of particular video sequences, allows for spatial and temporal based object detection and tracking [14]. We hypothesize that the addition of a memory module such as a convolutional 2-dimensional (2D) neural network (CNN) composed of Long-Short Term Memory (LSTM) nodes, can enable the network to learn to remember particular motion patterns and possible changes in height or width of predicted boxes [17]. This would allow the network to use both spatial awareness (through the CNN) and temporal awareness. Finally, the temporal pattern matching would allow for the memory of possible classifications of the bounding boxes and improve the performance and accuracy of video object detection.

The 2D CNN with LSTM component nodes was first introduced by Shi et al. [1] to enable interesting and useful spatial and temporal pattern matching. Others, such as Chen et al. [18], have used the 2D CNN LSTM component nodes with the Single-Shot Detector network in order to perform object detection on current image frames in a video. This style of network however, can not be said to be performing tracking as it is not predicting possible future object locations. In the tracking domain, networks must be able to predict possible locations of an object in the future. Trackers do not generate new boxes, they simply track older boxes. To generate new boxes a detector must be used. YOLO, although a novel object detector, is not capable of performing these predictive tracking on its own. We decided to implement a 2D CNN with LSTM components and integrated it with YOLO network to validate our above mentioned hypothesis about predicting objects and location in video data.

Since typical detectors are unable to perform this type of object tracking and future object detection, the use of the 2D CNN with LSTM nodes allows for the spatio-temporal

978-1-5386-5541-2/18/$31.00 ©2018 IEEE

memory [18]. We decided that YOLO would be a great candidate for this as YOLO is already an exceptionally fast network that, if slowed by performing sequence calculations (as with an LSTM), would still be faster than most object detectors. We decided to use the 2D CNN LSTM and the YOLO network to test our hypothesis as it would allow us to create a network that performs object detection with spatio-temporal pattern matching at a reasonable speed.

A. Paper Contributions

We propose a Predictive Object Detector (POD) network and combine a spatiotemporal object prediction model with an object detector model with a view to predict objects in video streams. The contributions of this paper are threefold. First we extend the YOLO [4] by adding recurrency to the network and training sequence-to-sequence for predictive object detection. The goal is to produce a network that can work on a large amount of video data and produce spatiotemporal patterns which can be extrapolated to object detections and classifications. The extended network is able to work with noisy and blurry images and thus is more robust than YOLO. The second and more significant contribution of our paper is the inclusion of a convolutional 2D LSTM model [1] as a spatiotemporal pattern matching layer in the YOLO network to enable predictive object detection. The third contribution is the prototype implementation of two different versions of our framework with a view to create a seamless and more efficient integration of the component object prediction and detection models. We present experimental results to validate the performance and accuracy of the two approaches by implementing and testing prototype artificial neural networks.

The two versions of our network apply the LSTM layer before the YOLO for pre-recurrent object detection, and after YOLO for post-recurrent object detection. The pre-recurrent version uses the 2D CNN-LSTM to generate feature maps that correspond to important patterns required by the detector to perform predictions. See Fig. 1 for some examples of the feature maps generated by this implementation. The results show that our Predictive Object Detector (POD) network offers extremely important results that are not commonly achieved through standard image object detection such as the ability to maintain detection of an object even after it has been fully occluded. We believe further research using this layer can lead to valuable breakthroughs in predictive image processing.

The remainder of this paper is organized as follows. Section II begins by illustrating related works in the field of object detection, and as the model illustrated in this paper is predictive, we will also discuss trackers. Implementation details regarding the model architecture and the loss function are detailed in Section III. Section IV discusses the dataset used for training and validation of the network. Finally, Section V discusses the evaluation methods, results as well as validation of the network.

Fig. 1: Example activations from the convolutional 2D LSTM layer in our POD network.

II. Related Work

A. Object Detection

One of the most interesting object detectors in the field is known as the You-Only-Look-Once 9000 detector (YOLO9000) [19] which revolutionized the field of computer vision by introducing a state-of-the-art, real time object detector capable of detecting over 9000 classes from ImageNet [4]. This object detector is a series of 22 blocks of convolutional, max pooling, and leakyReLU layers. A skip layer is attached on the 20th block, concatenating a fine and a coarse series of feature maps.

The object detector in YOLO9000 uses two robust training methods,one of which we have implemented and one which we have not. The training method we have implemented is the one wherein images are preprocessed and augmented prior to being fed into the network. The second robust training method in the YOLO 9000 network is done such that both images that have been labelled with and those labelled without bounding boxes are fed to the network for training. For the images with bounding boxes, the network performs training as usual, but for those without bounding boxes, the regression aspect of the network is locked so that only classification can be trained.

This allows their network to learn a very large number of classes and reduces the boundary between object detection and image classification [4]. For the sake of computational efficiency, and because we are only training for the 12 Multi-Object Tracking (MOT) classes, we did not implement the selective classification versus regression training.

Commonly used object detection methods applying recursive convolutional neural networks (R-CNN) involve 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 [2, 16]. Unfortunately, these methods are still quite slow in comparison to the method used by YOLO and YOLO9000, which use an anchor based method that allows the network to predict over the whole image using grids to segment specific parts of the image [16, 4]. An image-net or VGG16 style network can be used to extract features out of the image and reshape into a $[GridHeight \quad GridWidth \quad numAnchors * (numClasses + 5)]$ shape tensor (in this case the YOLOv2 feature extractor was used) [4] [16]. This is then fed into a convolutional block with an output shape that relates to the desired results, $[[x \quad y \quad delta\_x \quad delta\_y] \quad [confidence] \quad [classes]]$ [16, 4]. A custom loss function as in Fig. 2, is used to judge the networks ability to predict bounding boxes and classes, which is discussed in detail in YOLOv2 and YOLO9000 [4].

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

Fig. 2: Loss function from the YOLOv2 Network [4].

B. Object Tracking

Single object tracking requires the previous frame and the current frame, as well as the bounding box information on the previous frame [17]. It is able to track novel objects in images and estimate their movements and variance [2, 9]. Held et al. [17] propose a network that learns motion smoothness by making certain that the centers of the bounding boxes are described relative to the previous frame, and the model learns the changes in scale in width and height from the previous frame.

Most object trackers define tracking as the analysis of video sequences for the purpose of establishing the location of the target over a sequence of frames [2]. A common practice in the field is to start with the bounding boxes on the initial frame [17, 12]. 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, [9] focuses 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 looks at a variety of different parameters to both predict and track an activity [9].

From the literature study [2, 11, 13, 6], it can be seen that there are certain features of a neural network tracker and object detector that are extremely important such as the ability of the network to generalize well, handle occlusions, reflections, motion blur and transfer training.

III. Predictive Object Detector (POD) Network

We apply the object detection network implemented by the YOLOv2 network architecture [4] that is wrapped in a Time Distributed wrapper with additional convolutional 2D LSTM layer added on top of the network. The YOLO loss function is extremely important to the network as it manages to evaluate a variety of different aspects of the problem. The loss function (see Fig. 2) penalizes bad localization of cells, bounding boxes with inaccurate height and width values, low confidence scores for correct predictions, high confidence scores for incorrect predictions and a standard classification loss [16]. Weights can be assigned to mark object prediction as more important than classification or vice-versa. The network performs non-max suppression in order to stop overlapping predictions of the same class by applying a cap on the maximum number of boxes that need to be predicted. For the sake of computational efficiency, a maximum of ten boxes are chosen for final prediction.

The loss is calculated per batch, and a validation loss is calculated per epoch as seen in Fig. 3. This allows us to evaluate how the network performs in each epoch, and compare it to the previous epochs and only save the best network. The network stops training when it can no longer reduce the validation loss for 10 epochs, and the network with the lowest validation loss is the one that is saved.

PODv1 contains three object detector networks (YOLOv2). All the weights are shared between the three object detectors. This YOLOv2 network is trained on the COCO dataset and fine-tuned on the MOT 2017 dataset. Once this is done, the YOLOv2 detector is used to output predictions for three images. These predictions are fed into the convolutional 2D LSTM layer and a final prediction is made. The PODv1 network contains over 150 million parameters.

The PODv2 network, by comparison, only contains 50 million parameters. This network only has one object detector (YOLOv2) network which has been wrapped with a Time Distributed layer. The initial 6 images are fed into the convolutional 2D LSTM layer and the output feature set is passed into the Time Distributed object detector as in Fig. 4. This network is not only smaller, but faster than the PODv1 network.

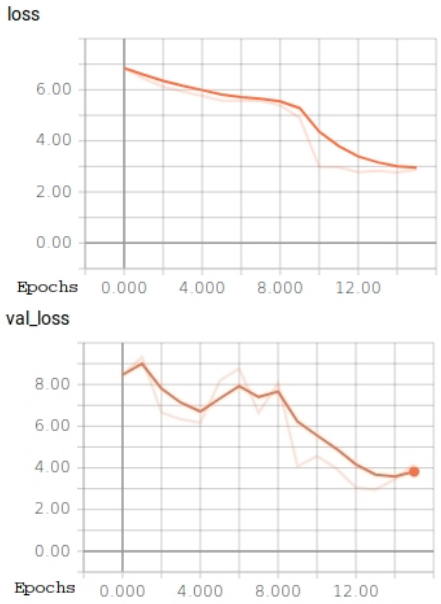

Fig. 3: Loss value and validation loss value per epoch.

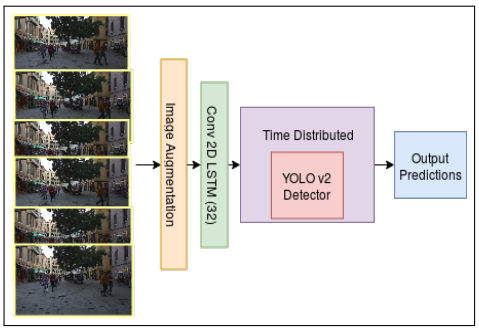

Fig. 4: Flow from video frame sequence to output of the PODv2 network.

Training was done on the Compute Canada Frontenac clusters with 24 core Intel Xeon CPU's. Final training was performed using the Graham Compute clusters and a Nvidia P100 GPU. Training for the PODv1 takes 10000-11000 iterations per class, which resulted in four to five days of training on the CPU. Training for the PODv2 network takes 9000-10000 iterations per class, which resulted in three to four days of training on the CPU or 10-12 hours on one Nvidia P100 GPU.

IV. Data

Some of the most common datasets that are used in object detection are the COCO dataset [7] and the MOT benchmark dataset [8]. The COCO dataset is a large-scale object detection, segmentation, and captioning dataset with several features. 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 training a classification neural network model. 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 Intersection Of Union (IOU) that it produces [4]. The IOU is a particular evaluation metric that is used to measure the accuracy of an object detector on a particular dataset. This is important so that when the network predicts a bounding box, it selects the type of anchor box and the location of the anchor box and the differences in width and height for the box.

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 labelled 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 benchmarking, leaderboards and evaluation techniques. Some of these evaluation techniques are standard in the object detection field, the most common of which are detailed in the evaluation section.

Each frame of the training video that is passed into the network is augmented using an image augmentation library. This allows us to add a variety of blurs, perform contrast normalization, and a variety of other augmentation techniques. This increases the variety of the data that is fed into the network and makes the network more robust. For validation and testing, image augmentation is turned off.

V. Results

A. Evaluation

The evaluation requires prediction of bounding boxes for each image and a calculation of the IOU against the ground truth using Eq. 1. The IOU is a particular evaluation metric that is used to measure the accuracy of an object detector on a particular dataset. The IOU is computed by dividing the area of overlap of the bounding boxes by the area of union of the bounding boxes. A good demonstration of the specific scenario is in Fig. 5. The straight lines represent the prediction, and the dotted lines represent the ground truth box. The IOU metric allows us to get a good representation of how well our prediction compares to the ground truth. This calculation can be seen as the percentage of the ground truth that is encompassed within the predicted bounding box as shown in Eq. 1. We use the IOU metric to calculate individual class average precisions (AP) as in Eq. 2. Finally, an additional evaluation metric called mAP (mean Average Precision) is calculated as in Eq. 3. There are a variety of different mAP values that can be used to compare our POD network against other networks. Since mAP is one of the most recorded values for object detectors, we used it for evaluation purposes.

Our initial implementation (PODv1) yielded a rather low mAP on the MOT dataset, while the second implementation (PODv2) yielded a 10% increase in mAP compared to PODv1 as shown in Table I. Khoreva et al. [13] discuss the idea of generating in-domain training data by synthesizing plausible future video frames. The POD network does not synthesize plausible future video frames however, the convolutional 2D

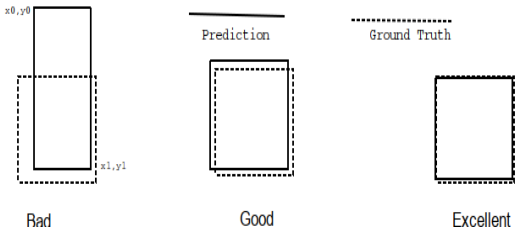

Fig. 5: The intersection of union calculation for bounding boxes.

$$
\begin{aligned}
box_a &= [x_0, y_0, x_1, y_1] \\
box_b &= [x_0, y_0, x_1, y_1] \\
max_{xy} &= minimum(box_a[2:], box_b[2:]) \\
min_{xy} &= maximum(box_a[:2], box_b[:2]) \\
diff_{xy} &= max_{xy} - min_{xy} \\
intersection &= diff_{xy}[0] * diff_{xy}[1] \\
area_a &= (box_a[2] - box_a[0]) * (box_a[3] - box_a[1]) \\
area_b &= (box_b[2] - box_b[0]) * (box_b[3] - box_b[1]) \\
union &= area_a + area_b - intersection \\
IOU &= \frac{intersection}{union}
\end{aligned} \tag{1}
$$

Above is the IOU calculation. Variables x_0, y_0, x_1, y_1 are the left, top, right, bottom values of a box.

$$
\begin{aligned}
AP_c &= \sum_i p_i * (r_i - r_{i-1}) \\
p_i &= \frac{tp_i}{tp_i + fp_i} \\
r_i &= \frac{tp_i}{tp_i + fn_i}
\end{aligned} \tag{2}
$$

Where c is the class, r is the recall, p is the precision, tp is the number of true positives, fp is the number of false positives and fn is the number of false negatives.

$$
mAP = \frac{\sum_{c \epsilon C} AP_c}{|C|} \tag{3}
$$

Where c is the class, C is the set of all classes. $|C|$ is the cardinality of the set of classes. AP is the value calculated from Eq. 2, which calculates the value for the average precision for class c.

LSTM layer allows the network to develop interesting feature maps as those in Fig. 1 that incorporate temporal patterns among the spatial features. This allows the network to perform object detection on the temporal and spatial patterns from the beginning, rather than attempting to finely separate them at the end. We applied this idea in PODv2 and it resulted in the increase in mAP. It should be noted that a higher value of mAP is better and often a higher value for frames per second is often considered better, as long as we do not affect mAP too severely [19].

If we look at Fig. 6 for an example of PODv2 network output, we can see that the network accurately detects the classes, locations and sizes of the bounding boxes. Generally, the network begins with a lower class confidence score, and increases the class score very quickly as the network learns to differentiate objects. When other objects in the scene occlude an object, it is still possible for PODv2 to remember the occluded object's location and predict it. This is illustrated in Fig. 7.

TABLE I: Comparison of FPS and mAP of a variety of different detectors

Detector	Sequence Length	mAP[8]	Frames Per Second
YTLAB	1	*0.89*	N/A
YOLOv2	1	0.64	*40*
PODv2 (pre-recurrent)	6	0.55	1.1
PODv1 (post-recurrent)	3	0.45	0.5

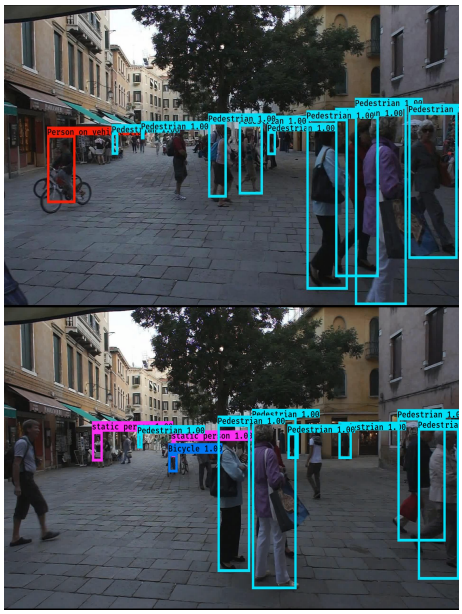

Fig. 6: Example of pre-detector network output.

B. Discussion

Both PODv1 and PODv2 performed worse than the state-of-the-art methods for object detection as in Table **??** however, it still generates interesting results compared to the other object detectors. The reason for this seems to be the fact that the POD network is entirely predictive, unlike the other object detectors. Object trackers such as those detailed in [13, 12] require initial bounding boxes in order to perform object predictions but the POD does not. These trackers tend to focus on object assignment inference and object velocity inference. Our POD network detailed in this paper performs object assignment inference, object velocity inference, and

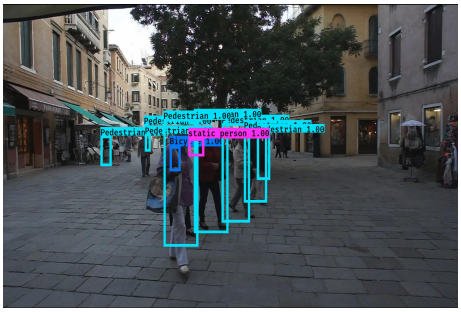

Fig. 7: The bicycle and static pedestrian are detected as they are still in memory.

object detection. Any additional step introduces a new loss vector causing the network to perform less efficiently.

Since the POD network has a maximum of 10 boxes, it will always miss some boxes during prediction since the MOT dataset has many more annotated boxes (often between 25-35 boxes) [8]. By increasing the number of bounding boxes for the network to predict, we can increase the capability of POD to identify more objects but it will also increase processing time and may introduce additional error in object detection. The error can be mitigated slightly by setting threshold values for confidence score and IOU. However, it would again incur additional compute cost and time.

VI. Conclusion

We propose a Predictive Object Detector (POD) that performs object prediction to detect objects in video data which is composed of a sequence of image frames. POD uses a sequence of previous frames to predict objects for classification. It uses the architecture from YOLOv2 paper and expands upon it using the novel Convolutional 2D LSTM layer. The Convolutional 2D LSTM layer is able to learn sequence-to-sequence object detection and tracking through prediction. Although the POD network under-performs compared to the state-of-the-art methods, it provides insights on the development of the field in terms of the predictive capabilities of these networks and provides interesting results compared to the state-of-the-art methods.

The use of the MOT dataset for video object detection was useful in terms of benchmarking and demonstrating the capabilities of the POD network, in comparison to the YOLOv2 detector.

For future work, we would like to use the objects relative velocity and object relative width and height changes instead of absolute x and y values. The output of the network should also be fed forward to the next input of the network to ease object tracking and allow the network to converge more easily. Finally, developping a loss function that takes into consideration the time-steps could also be beneficial.

The future of video object detection and tracking is bright. With advances in computational power and deep learning models, increasing research in image/video analysis is inevitable and interesting.

Acknowledgment

This project would not be possible without the help of the CAC (Canadian Advanced Computing) and Compute Canada. We would like to thank them for their help.

References

[1] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo, "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting," pp. 1–12, 2015. [Online]. Available: http://arxiv.org/abs/1506.04214

[2] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1442–1468, 2014.

[3] Y. Tian, P. Luo, X. Wang, and X. Tang, "Deep learning strong parts for pedestrian detection," Ph.D. dissertation, The Chinese University of Hong Kong, 2015.

[4] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," Ph.D. dissertation, University of Washington, 2016. [Online]. Available: http://arxiv.org/abs/1612.08242

[5] K. Kang, W. Ouyang, H. Li, and X. Wang, "Object Detection from Video Tubelets with Convolutional Neural Networks," Ph.D. dissertation, The Chinese University of Hong Kong, 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7780464/

[6] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. V. D. Hengel, "A Survey of Appearance Models in Visual Object Tracking," *ACM Trans. Intell. Syst. Technol.*, vol. 4, no. 4, pp. 58:1—-58:48, 2013. [Online]. Available: http://doi.acm.org/10.1145/2508037.2508039

[7] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8693 LNCS, no. PART 5, pp. 740–755, 2014.

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

[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 Transactions on Image Processing*, vol. 19, no. 10, pp. 2564–2579, 2010.

[10] Q. Wang and J. Liu, "Visual Tracking Using the Kernel Based Particle Filter and Color Distribution," Ph.D. dissertation, Zhejiang University, 2005.

[11] I. Posner and P. Ondruska, "Deep Tracking: Seeing Beyond Seeing Using Recurrent Neural Networks," Ph.D. dissertation, University of Oxford, 2016.

[12] 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,"

Ph.D. dissertation, Technical University Munich, 2017. [Online]. Available: http://arxiv.org/abs/1704.02781

[13] A. Khoreva, R. Benenson, E. Ilg, T. Brox, and B. Schiele, "Lucid Data Dreaming for Multiple Object Tracking," pp. 1–16, 2017. [Online]. Available: http://arxiv.org/abs/1703.09554

[14] P. Tokmakov, K. Alahari, and C. Schmid, "Learning Video Object Segmentation with Visual Memory," *Iccv*, pp. 4481–4490, 2017. [Online]. Available: http://arxiv.org/abs/1704.05737

[15] R. Yu, H. Wang, and L. S. Davis, "ReMotENet: Efficient Relevant Motion Event Detection for Large-scale Home Surveillance Videos," Ph.D. dissertation, University of Maryland, 2018. [Online]. Available: https://arxiv.org/pdf/1801.02031v1.pdf

[16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2015. [Online]. Available: http://arxiv.org/abs/1506.02640

[17] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 FPS with deep regression networks," Ph.D. dissertation, Stanford University, 2016.

[18] X. Chen, J. Yu, and Z. Wu, "Temporally Identity-Aware SSD with Attentional LSTM," pp. 1–13, 2018. [Online]. Available: http://arxiv.org/abs/1803.00197

[19] C. Sun, K. Murphy, T. Kong, A. Yao, Chen *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," *NIPS*, vol. 11-18-Dece, no. 2, pp. 185–192, 2017. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/6714453/