EVOLVING BOXES FOR FAST VEHICLE DETECTION

Li Wang[1,2]*, Yao Lu*[3]*, Hong Wang*[2]*, Yingbin Zheng*[2]*, Hao Ye*[2*]*, Xiangyang Xue*[1]

[1]Shanghai Key Lab of Intelligent Information Processing, School of Computer Science, Fudan University, Shanghai, China
[2]Shanghai Advanced Research Institute, Chinese Academy of Sciences, Shanghai, China
[3]School of Computer Science and Engineering, University of Washington, Seattle, USA
wangli16@fudan.edu.cn, luyao@cs.washington.edu,
{wang_hong,zhengyb,yeh}@sari.ac.cn, xyxue@fudan.edu.cn

ABSTRACT

We perform fast vehicle detection from traffic surveillance cameras. A novel deep learning framework, namely Evolving Boxes, is developed that proposes and refines the object boxes under different feature representations. Specifically, our framework is embedded with a light-weight proposal network to generate initial anchor boxes as well as to early discard unlikely regions; a fine-turning network produces detailed features for these candidate boxes. We show intriguingly that by applying different feature fusion techniques, the initial boxes can be refined for both localization and recognition. We evaluate our network on the recent DETRAC benchmark and obtain a significant improvement over the state-of-the-art Faster RCNN by 9.5% mAP. Further, our network achieves 9-13 FPS detection speed on a moderate commercial GPU.

Index Terms— Vehicle detection, deep neural networks

1. INTRODUCTION

Vehicle detection is essential in various computer vision applications including traffic surveillance [1] and auto-driving. Classic vehicle detectors [2, 3] have achieved promising detection results. Notably, the cascade object detector [2] applies a set of weak-classifiers and filters background objects early. More recently, the region-based RCNN [4] has gained considerable attention in detecting generic objects. Faster RCNN [5] achieves state-of-the-art performance by generating potential object boxes with an embedded region proposal network (RPN). Detecting vehicles robustly and efficiently under different pose, scale, occlusion, and lighting conditions is nevertheless challenging. Figure 1 demonstrates two examples for vehicle detection with real-world traffic camera feeds.

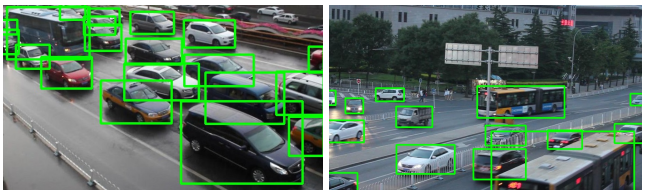

Fig. 1: Vehicle detection in real traffic surveillance cameras is challenging. There is often a large variation in occlusion and lightning conditions as well as vehicle pose and scale patterns. We demonstrate two examples from the DETRAC vehicle detection dataset [6]; heavy traffic, different weather conditions and different vehicle types are shown in these examples.

To boost the accuracy as well as efficiency for vehicle detection, we are inspired by the classic cascade object detection framework; a novel deep learning framework, namely Evolving Boxes (EB), is proposed in which the object boxes are being refined along with our algorithmic pipeline.

Specifically, we set up two neural networks in a single deep learning framework after a pre-trained deep convolutional network. The first proposal network (PN) employs a set of small convolutional layers to generate candidate boxes while discarding the unlikely ones. Potential boxes are sent to the second fine-turning network (FTN) in which sophisticated convolutional layers are leveraged. Intriguingly, we find that applying different feature fusion techniques yields promising results. That is, by combining the features from PN, FTN and different convolutional layers, the initial object boxes can be refined in terms of both localization and class regression. We demonstrate a considerable improvement for the evolved boxes over those regressed directly from the sophisticated features.

Given such evolving detection structure, we are able to accelerate the overall deep learning networks for vehicle detection. This is especially useful for real-world scenarios where background objects are often the majority. On the other hand, training and using our deep neural networks is end-to-end; it does not require frequently and densely turning the hyper-

*Corresponding author.

This work was supported in part by grants from NSFC (#61602459, #61572138, and #U1611461) and STCSM's program (#16511104802, #16JC1420401, and #17YF1427100).

978-1-5090-6067-2/17/$31.00

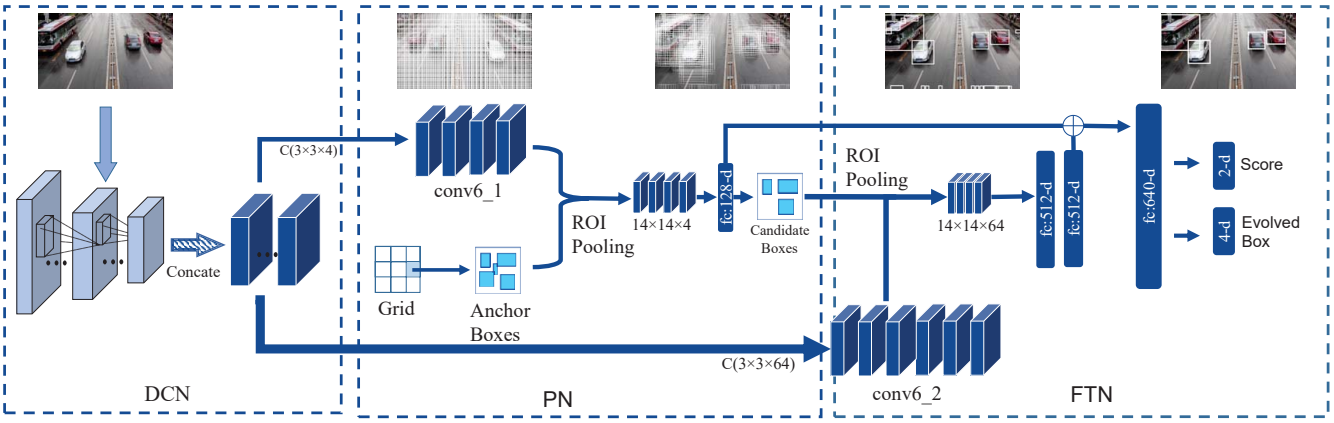

Fig. 2: The evolving architecture of our framework. Three networks are involved including the Deep Convolutional Network (DCN), the Proposal Network (PN) and the Fine-Tuning Network (FTN). DCN is responsible for generating rich features from images. PN produces anchor proposals and filters the ones that are unlikely to be vehicles. FTN further fine-tunes the candidates and generates refined localization and recognition results. Concate: feature concatenation from different conv layers. $\oplus$: feature concatenation from different networks.

parameters or in multiple training stages.

To evaluate our proposed framework, we report the evaluations on the recent DETRAC vehicle detection dataset [6], and we compare with several recent approaches. Notably, we achieve a significant 9.5% mAP improvement over the state-of-the-art Faster RCNN [5]. Meanwhile, a 9-13 FPS detection rate is obtained on an Nvidia Titan X GPU (Maxwell).

The rest of this paper is organized as follows. Section 2 introduces the background and related work for vehicle detection. Section 3 discusses our evolving framework in detail, while Section 4 demonstrates the experiments.

2. RELATED WORKS

Vehicle detection has been studied for decades with numerous real applications [7, 8, 9]; it has drawn considerable attention when the cascade methods [9] and the deformable part models (DPM) detectors [3] were introduced. The cascaded vehicle detection was initially proposed by Viola and Jones [2] with a set of weak classifiers to early filter image patches that are not target objects. Later works [10, 11, 12] extend this cascade pipeline and achieve good performance. On the other hand, DPM detectors successfully detect the target objects by retrieving the object parts and finding their spatial combinations [7]. So far, Convolutional Neural Networks (CNNs) have shown their rich representative power in object detection [4]. To employ the cascade detection strategy and to reduce candidate objects, the Faster RCNN [5] achieves state-of-the-art performance by generating potential object locations using RPN. Inspired by these works, we extend the cascade strategy and propose a novel evolving framework for vehicle detection. Recently, several popular object detection frameworks utilize anchors or grids to propose candidate object locations, which yields end-to-end object detections without explicitly cropping out candidate objects for further recognition. YOLO [13] directly regresses the object locations for each grid in the image. SSD [14] generates several candidate objects for each anchor in the image. We find that directly localizing the objects in a single shot often yields unsatisfactory results. Instead, our proposed boxes are evolving and being fine-tuned by different networks within our framework.

3. FRAMEWORK

We illustrate the evolving architecture and different networks of our deep learning framework in Figure 2 and as follows:

- *Deep Convolutional Network* (DCN). The entire images are initially fed to several convolutional and max-pooling layers to generate rich feature representations. Feature concatenation from different convolutional layers are used here. The output feature maps are sent to the proposal network (PN) and the fine-turning network (FTN) afterwards.

- *Proposal Network* (PN). As shown in Figure 2, the PN network takes the feature maps from the previous convolutional network (DCN) as input. A small convolutional layer with 4 filters later is connected afterwards (conv6_1). The image and the output feature map are divided into grids; for each grid, a fixed number of anchor proposals are generated. For each anchor proposal, the ROI pooling layer crops out a $14 \times 14 \times 4$ feature vector, based on which a fully connected (fc) layer of 128-d directly regresses the initial bounding box (x, y, w, h) and its object score s. Candidate boxes with low object scores are discarded.

- *Fine-tuning Network* (FTN). The FTN network similarly takes the feature maps from the previous convolutional network (DCN) as input. A large convolutional layer with 64 filters is connected afterwards (conv6_2). For each candidate box produced by the PN network, another ROI pooling layer crops out a $14 \times 14 \times 64$ feature vector, which is sent to a set of fc layers for fine-turning. The fc layer feature from the PN network is concatenated to the fc layer feature (512-d) herein. The final output is evolved object boxes (x', y', w', h') and object scores s'.

We describe below each individual network in detail.

Deep Convolutional Network (DCN). Rich feature representations are produced by this network. We leverage the convolutional and max-pooling layers from the VGG-16 [15] network and load the weights pre-trained on the ImageNet dataset [16]. This is a common practice among many state-of-the-art deep learning frameworks [13, 5] to ensure that the detection can benefit from evolving big datasets.

As discussed by [17], high convolutional layers perform better on the classification but lack insight to precise object localization, because the feature weights have been summarized by multiple convolutional layers. On the contrary, low convolutional layers have a better scope to localizing objects as they are closer to raw images. Prior works [18, 19] have shown the gain of combining the feature maps from different convolutional layers. To mirror this, we leverage concatenation of different convolutional layers in DCN to feed into the later PN and FTN networks. Figure 2 demonstrates the multi-feature map concatenation (denote as 'concate' in Figure 2) and we will show in our experiments how this affects the detection accuracy and speed. We leverage down- and up-sampling methods when combining different feature maps: we down-sample low layer feature maps to align with high layer feature maps when combining the two layers; to combine three layers, we down-sample the low layer and up-sample the high layer to align with the middle layer feature map. After aligning the feature maps, we normalize these feature maps using a batch normalize layer [20], and then concatenate them to form a hyper feature map. In our experiments, we concatenate the feature maps from the 1st, 3rd, and 5th convolutional layers.

Proposal Network (PN). As background objects are always the majority, filtering out pool candidates is essential to build an efficient vehicle detection system. To this end, we leverage a small PN network to propose candidate boxes as well as to eliminate background regions. Specifically, a small convolutional layer with 4 filters is connected to the DCN output (conv6_1) and the 256×144 feature map is obtained. We divide the feature maps into 64×36 grids while for each grid, a set of *anchor boxes*, or initial object boxes, are generated with fixed sizes of 32×32, 64×64, 128×128, 256×256 and 512×512, and different aspect ratios of 1:2, 2:1, and 1:1. Unlike the Faster RCNN, we directly take the anchor boxes to propose candidate objects, instead of using an additional fc layer. A ROI pooling layer later crops out a $14 \times 14 \times 4$ feature vector from each anchor box. The cropped feature vector is sent to a fc layer with 128-d to regress the candidate object box (x, y, w, h) and object score s. We show later in our experiments that using this simple but effective proposal network, roughly 98% of the background regions can be discarded, leading to a significant speed boost.

Fine-tuning Network (FTN). The FTN is responsible for fine-tuning the remaining object boxes. The structure is similar to the previous PN network, except that a convolutional layer with 64 filters are used (conv6_2) to produce the 256×144 feature map. The ROI pooling layer extracts a $14 \times 14 \times 64$ feature vector for each vehicle candidate produced by the PN. A fc layers with 640-d is connected afterwards that is concatenated from two fc layers (the concatenation is denoted by the $\oplus$ operator in Figure 2); the first has 512-d produced by the current FTN, while the second 128-d is generated by the PN. The output of the fc layers is a 5-d vector, (x', y', w', h') for evolved object box and s' for refined object score. We will show in our experiments that this evolving architecture is superior to directly regressing the object boxes, which is done in many state-of-the-art detection frameworks such as YOLO and Faster RCNN. Further, the concatenation of the two-stage features further improves the detection results.

Networks Training. As stated above, our networks are partly initialized by the pre-trained ImageNet model VGG-16 [15] for the DCN part, and the other networks are randomly initialized from a zero-mean Gaussian distribution with standard deviation of 0.01. We set the initial learning rate to 10^{-3} and then decrease to 10^{-4} after 50k iterations. We totally train 70k iterations. The benchmark and comparisons are reported with training using the entire training and validation set, while to evaluate different algorithmic components, only the training set is used.

Our model is trained end-to-end using stochastic gradient descent. We use the mini-batch size of 256. For the PN network, we assign positive anchor proposals that overlap the ground truth for more than 0.5 in intersection over union (IOU) [3]. Anchor proposals that overlap the ground truth for less than 0.3 in IOU are assigned as negative examples. We run non-maximum suppression (NMS) [3] with threshold 0.7 to eliminate redundant boxes and keep 800 of them. For the FTN network, we assign positive candidates that overlap the ground truth for IOU$\geq$0.45, while candidates with $0.1 \leq$IOU$\leq$0.3 are assigned with negative examples. We also apply hard mining [3] during training the FTN; we sort the classification loss in descending order, and pick the top 70% samples to participate in the back propagation and we ignore easy examples.

Multi-stage Loss. The fc layers in PN and FTN generate (x, y, w, h, s) and (x', y', w', h', s') respectively, denoting the

bounding box locations and object scores produced by the two networks. Similar to [4], we use $t = (t_x, t_y, t_w, t_h)$ to parameterize the bounding box generated by the first stage PN:

$$\begin{aligned} t_x &= (x - \hat{x})\ /\hat{w},\ t_w = \log(w/\hat{w}), \\ t_y &= (y - \hat{y})\ /\hat{h},\ \ t_h = \log(h/\hat{h}) \end{aligned} \tag{1}$$

where $\hat{x}, \hat{y}, \hat{w}, \hat{h}$ are the location of the initial anchor box. Meanwhile, we use $t' = (t'_x, t'_y, t'_w, t'_h)$ to parameterize the evolved box generated by the second stage FTN:

$$\begin{aligned} t'_x &= (x' - x)\ /w,\ t'_w = \log(w'/w), \\ t'_y &= (y' - y)\ /h,\ \ t'_h = \log(h'/h) \end{aligned} \tag{2}$$

Therefore, a multi-stage loss L is used to jointly train the two-stage classification and regression:

$$\begin{aligned} L &= \alpha L_{pn} + (1-\alpha) L_{ftn}, \\ L_{pn}(t, s) &= L_{cls}(s) + \lambda L_{loc}(t, s), \\ L_{ftn}(t', s') &= L_{cls}(s') + \lambda L_{loc}(t', s'), \end{aligned} \tag{3}$$

where L_{pn} and L_{ftn} are the loss for PN an FTN respectively. α balances the two stages. The class regression loss $L_{cls}(s) = -\log s$ is a logarithmic loss upon class scores. To regress bounding boxes locations, we follow [21] and use the localization loss L_{loc} defined as:

$$L_{loc}(t, s) = \sum_{i \in \{x,y,w,h\}} \sigma(t_i - s), \tag{4}$$

where

$$\sigma(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1, \\ |x| - 0.5 & \text{otherwise.} \end{cases} \tag{5}$$

We consider the PN and FTN stages equally important; $\alpha = 0.5$ is set to compute the multi-stage loss.

4. EXPERIMENTS

We trained our model on the recent DETRAC vehicle detection dataset [6] with 140K captured frames and 1.2M labeled vehicles. It contains 84K images for training and we further split it into the training set with 56K images and the validation set with 28K image. The image resolution is 960×540, and user marked rectangles exist to represent non-detection regions. The DETRAC dataset is challenging due to its large variation; the cameras are mounted on traffic poles in Beijing, while the video frames are captured in different scenarios including sunny, cloudy, rainy and night. By mean each video frame contains 8.6 vehicles and occlusion happens frequently. Figure 1 shows two examples of this dataset.

To evaluate the effectiveness of our proposed framework, we focus on answering (1) how different algorithmic components affects the vehicle detection performance, and (2) how our overall framework compares with state-of-the-art vehicle detection methods. For a fair comparison, results for the first experiment is reported on the validation set while benchmarks on the test set is reported to compare with other methods. The following sections discuss each experiment respectively.

Algorithmic Setting	Overall	Sunny	Cloudy	Rainy	Night
Faster RCNN [5]	68.58	63.64	70.04	81.56	60.53
PN+FTN+Fusion	73.96	69.70	73.90	82.12	67.91
PN+FTN+Concat	83.84	87.09	84.80	85.95	70.21
PN+FTN+Fusion+Concat	**84.43**	**87.48**	**85.88**	**85.65**	**70.86**

Table 1: Control experiments on switching off different algorithmic components of our framework. We illustrate mean average precisions (mAP) on the DETRAC validation set as well as different subsets.

Algorithmic Setting	Runtime Speed (ms)
Faster RCNN [5]	87
PN+FTN+Fusion	75
PN+FTN+Concat	100
PN+FTN+Fusion+Concat	110

Table 2: Inference time on different algorithmic combinations.

4.1. Control Experiments

Table 1 demonstrates the control experiments of switching off different components of our proposed framework. Detection performances including the overall mAP and mAPs under different scenarios are reported. PN+FTN+Fusion indicates that we do not use the multi-layer feature map concatenation from the 1st, 3rd, and the 5th convolutional layers and only the final convolutional layer features are used. PN+FTN+Concat turns off the multi-stage feature concatenation from different networks, while our full model is PN+FTN+Fusion+Concat. We also compare with the Faster RCNN [5], which can be seen as using the PN to propose candidate vehicles but directly regressing the vehicle detection results; fine-tuning the detections boxes and class scores are not involved in their approach. Besides, multi-layer and multi-stage feature concatenation are not used in Faster RCNN either.

Quantitatively speaking, our full model performs the best among different algorithmic settings. As the Faster RCNN does not apply the proposal refinement or any other tricks discussed in this paper, its performance drops significantly for a 15.9% mAP compared with our full model. The multi-layer feature map fusion is critical, which introduces a 10.5% performance gain. This accords with the study in [17], and we consider this particularly useful for vehicle detection, as many vehicles from real traffic cameras are small in size. Ignoring the low level convolutional layer features prevents the network from finding any small object. Further, the multi-stage feature concatenation leads to another 0.6% performance gain. On the other hand, although feature fusion introduces additional runtime, the overall speed is still acceptable.

Method	Overall	Easy	Medium	Hard	Sunny	Cloudy	Rainy	Night	Speed	Environment
DPM [3]	25.70	34.42	30.29	17.62	24.78	30.91	25.55	31.77	6s/img	CPU@2.4GHz
ACF [9]	46.35	54.27	51.52	38.07	58.30	35.29	37.09	66.58	1.5s/img	CPU@2.4GHz
RCNN [4]	48.95	59.31	54.06	39.47	59.73	39.32	39.06	67.52	10s/img	GPU@K40
Faster RCNN [5]	58.45	82.75	63.05	44.25	62.34	66.29	45.16	69.85	0.09s/img	GPU@TitanX
CompACT [22]	53.23	64.84	58.70	43.16	63.23	46.37	44.21	71.16	4.5s/img	GPU@K40
Ours - EB	**67.96**	**89.65**	**73.12**	**54.64**	**72.42**	**73.93**	**53.40**	**83.73**	0.11s/img	GPU@TitanX

Table 3: Mean average precision (mAP) on the DETRAC test dataset produced by different state-of-the-art vehicle detection approaches. The runtime environment and speed are shown as well.

A 9-13 FPS detection rate is achieved using a single Nvidia Titan X GPU (Maxwell).

4.2. Comparing with State-of-The-Art

Figure 3 and Table 3 demonstrate the comparison of our full model with state-of-the-art vehicle detection approaches. The results can also be found at the DETRAC benchmark server[1]. Precision-recall curves and mAPs are reported. We compare with CompACT [22], RCNN [4], ACF [9], Faster RCNN [5], and DPM [3]. We achieve a significant overall improvement of 14.73% mAP over the state-of-the-art CompACT [22] and 9.5% mAP over Faster RCNN [5]. Notably, our method performs the best on all subcategories. Figure 3 further shows that our methods outperforms state-of-the-art approaches with different recall setting, indicating that our method achieves better detection coverage as well as accuracy.

Table 3 shows the environment and runtime speed for different approaches. Our proposed framework runs magnitude faster compared with CompACT (40x) and RCNN (90x), while on-par or slightly slower than the Faster RCNN.

Figure 4 and 5 demonstrate qualitative evaluations of our approach on the test set; successful and partially unsuccessful results are shown. We succeed in detecting most of the vehicles in different appearances, especially when heavy occlusion is happening or the vehicles are far away from the camera. However, there are also some failure cases where the vehicle detection is split into multiple boxes, or fails to identify multiple vehicles that are adjacent to each other. Generally speaking, the detection results are reasonable and high quality for further post-processing such as vehicle type and color recognition.

Our framework is implemented on Caffe [23], and we have released the code and trained model for future research[2].

5. CONCLUSIONS

We borrow the idea from cascade object detection and propose an evolving object detection framework in which the object boxes are generated and being refined by different networks within our proposed pipeline. We show that by leveraging different feature fusing techniques, good performance is achieved for both localization and class recognition. The runtime speed is 9-13 FPS on a moderate commercial GPU.

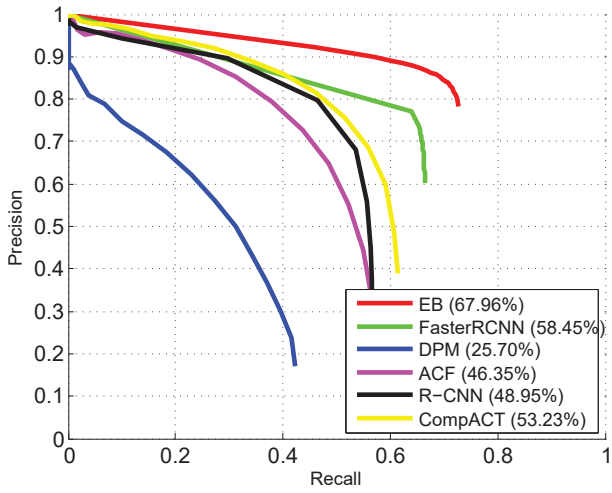

Fig. 3: Precision-recall curves of different vehicle detection algorithms on the DETRAC test set.

6. REFERENCES

[1] Y. Lu, A. Chowdhery, and S. Kandula, "Optasia: A relational platform for efficient large-scale video analytics," in *ACM SoCC*, 2016.

[2] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *CVPR*, 2001.

[3] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014.

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *NIPS*, 2015.

[1] http://detrac-db.rit.albany.edu/DetRet

[2] http://zyb.im/research/EB

Fig. 4: Successful detection results on the DETRAC test set.

Fig. 5: Partially unsuccessful detection results on the DETRAC test set.

[6] L. Wen, D. Du, Z. Cai, Z. Lei, M.-C. Chang, H. Qi, J. Lim, M.-H. Yang, and S. Lyu, "DETRAC: A new benchmark and protocol for multi-object detection and tracking," *arXiv preprint arXiv:1511.04136*, 2015.

[7] X. Song, T. Wu, Y. Jia, and S.-C. Zhu, "Discriminatively trained and-or tree models for object detection," in *CVPR*, 2013, pp. 3278–3285.

[8] R. Feris, R. Bobbitt, S. Pankanti, and M.-T. Sun, "Efficient 24/7 object detection in surveillance videos," in *IEEE AVSS*, 2015.

[9] P. Dollár, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 8, pp. 1532–1545, 2014.

[10] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, "A convolutional neural network cascade for face detection," in *CVPR*, 2015.

[11] H. Qin, J. Yan, X. Li, and X. Hu, "Joint training of cascaded cnn for face detection," in *CVPR*, 2016.

[12] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.

[13] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *CVPR*, 2016.

[14] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *ECCV*, 2016.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *CVPR*, 2009.

[17] A. Ghodrati, A. Diba, M. Pedersoli, T. Tuytelaars, and L. Van Gool, "DeepProposal: Hunting objects by cascading deep convolutional layers," in *ICCV*, 2015.

[18] R. Ranjan, V. M Patel, and R. Chellappa, "HyperFace: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition," *arXiv preprint arXiv:1603.01249*, 2016.

[19] T. Kong, A. Yao, Y. Chen, and F. Sun, "HyperNet: Towards accurate region proposal generation and joint object detection," in *CVPR*, 2016.

[20] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.

[21] R. Girshick, "Fast R-CNN," in *ICCV*, 2015.

[22] Z. Cai, M. Saberian, and N. Vasconcelos, "Learning complexity-aware cascades for deep pedestrian detection," in *ICCV*, 2015.

[23] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.