Compressing YOLO Network by Compressive Sensing

Yirui Wu[†,‡], Zhouyu Meng[‡], Shivakumara Palaiahnakote[§], Tong Lu[‡,*]
[†] College of Computer and Information, Hohai University, China
[‡]National Key Lab for Novel Software Technology, Nanjing University, Nanjing, China
[§] Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia
wuyirui@hhu.edu.cn;misskanagi@gmail.com;shiva@um.edu.my;lutong@nju.edu.cn

Abstract

Object detection is one of the fundamental challenges in pattern recognition community. Recently, convolutional neural networks (CNN) are increasingly exploited in object detection, showing their promising potentials of generatively discovering patterns from quantity of labeled images. Among CNN-based systems, we focus on one state-of-the-art architecture designed for fast object detection, named as YOLO. However, YOLO, as well as CNN-based systems are hard to deploy on embedded systems due to their computationally and storage intensive. In this paper, we propose to compress YOLO network by compressive sensing, which exploits inherent redundancy property of parameters in layers of CNN architecture, leading to decrease the computation and storage cost. We firstly convert parameter matrix to frequency domain through discrete cosine transform (DCT). Due to the smooth property of parameters when processing images, the resulting frequency matrix are dominated by low-frequency components. Next, we prune high-frequency part to make the frequency matrix sparse. After pruning, we sample the frequency matrix with distributed random Gaussian matrix. Finally, we retrain the network to finetune the remaining parameters. We evaluate the proposed compress method on VOC 2012 dataset and show it outperforms one latest compression approach.

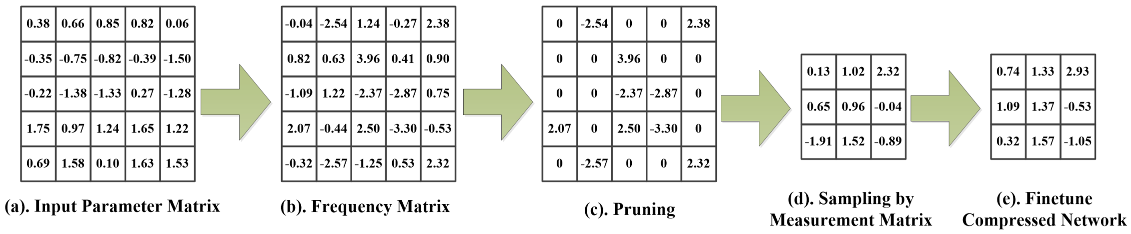
1. Introduction

Category-level object detection, i.e. spatially separates bounding boxes and associate class probabilities, has been one of the most active areas in pattern recognition and computer vision. In recent years, Convolutional Neural Networks (CNN) have attracted lots of attentions due to its impressive results. Various CNN-based approaches for feature extraction and classifier learning have been applied to solve the problem of object detection [19, 18, 17, 21]. Among them, YOLO system [19, 18] is designed to address the issue of computational cost, which could process 67 frames per second and achieve 76.8 mean Average Precision (mAP) simultaneously on VOC 2007 dataset [9]. It outperforms most of object detection methods including Deformable part-based model (DPM) [11] and R-CNN [13].

Impressed by the outstanding results of YOLO system, the operating systems to deploy YOLO are still restricted as servers and workstations. The main difficulty to shift YOLO from computers to embedded systems lies in its computational and storage intensity [15]. Specifically, YOLO system is over 230MB, consisted by 30 layers and $6.74*10^8$ parameters. Running YOLO with the tremendous number of parameters consumes large storage and computational resources, which are often limited in embedded systems. This problem occurs for other CNN-based systems as well. Therefore, how to compress the size of parameters and probably shift CNN architectures to embedded system has become one of the most challenging topics in pattern recognition commuting.

In this paper, we propose a novel approach to compress parameters of YOLO network with compressive sensing. Compressive sensing is well-known as a highly efficient signal compression method, which exploits the sparsity property of signal to recover it from far fewer samples than required by the Shannon-Nyquist sampling theorem. Due to the nature of local pixel correlation in images (i.e. spatial locality), filters in YOLO system tend to be smooth. Inspired by the theory of image compression, we first adopt discrete cosine transform to convert these filters into frequency domain [1]. After pruning high-frequency part, we achieve a sparse version of the frequency matrix. Next, we use distributed random Gaussian matrix to sample the frequency matrix. After the three steps, we retrain the network to finetune the remaining parameters. Using compressive sensing to compress parameters of YOLO network could help decrease the storage consumption. In running time, sparse distribution of YOLO parameters would greatly speed up the calculation.

Our approach has two major contributions:



0.38	0.66	0.85	0.82	0.06
-0.35	-0.75	-0.82	-0.39	-1.50
-0.22	-1.38	-1.33	0.27	-1.28
1.75	0.97	1.24	1.65	1.22
0.69	1.58	0.10	1.63	1.53

(a). Input Parameter Matrix

-0.04	-2.54	1.24	-0.27	2.38
0.82	0.63	3.96	0.41	0.90
-1.09	1.22	-2.37	-2.87	0.75
2.07	-0.44	2.50	-3.30	-0.53
-0.32	-2.57	-1.25	0.53	2.32

(b). Frequency Matrix

0	-2.54	0	0	2.38
0	0	3.96	0	0
0	0	-2.37	-2.87	0
2.07	0	2.50	-3.30	0
0	-2.57	0	0	2.32

(c). Pruning

0.13	1.02	2.32
0.65	0.96	-0.04
-1.91	1.52	-0.89

(d). Sampling by Measurement Matrix

0.74	1.33	2.93
1.09	1.37	-0.53
0.32	1.57	-1.05

(e). Finetune Compressed Network

Figure 1. The framework of the proposed method: (a) input parameter matrix sampled from filters of YOLO, (b) transforming parameter matrix to frequency domain by DCT, (c) pruning frequency matrix, (d) sampling frequency matrix by setting distributed random Gaussian matrix as measurement matrix and (e) finetuning the compressed network to get the modified version of sampling frequency matrix.

- Introduction of a compression method for CNN network based on compressive sensing technique. Compressive sensing is designed to compress the signal under the theoretical guidance of sparse coding. CNN filters for images processing are often sparse and smooth, which coincides with the idea of compressive sensing. To the best of our knowledge, this is the first work compressing CNN network with compressive sensing.

- Construction of a much smaller YOLO system (nearly 100MB) that supports to deploy in an embedded system. Practically, the proposed compress approach for YOLO system could be easily modified and applied to any CNN-based system.

The rest of the paper is organized as follows. Section 2 reviews the related work. Details of the proposed compress method is discussed in Section 3. Section 4 presents the experimental results and discussions. Finally, Section 5 concludes the paper.

2. Related Work

The existing methods related to our work can be categorized into the following two types: deep compression and object detection methods.

2.1. Deep compression

Deep neural networks have demonstrated the state-of-the-art power for computer vision tasks. However, these models are hard to shift to embedded system due to the limited computational and memory resource. More researchers begin to focus on methods for compression of deep models.

We generally category them into two groups: compress during and after training. The methods in the first group compress weights, activations and gradients during training to obtain smaller and faster network. For example, Courbariaux *et al.* [5] train binarized neural networks (BNNs) with binary weights and activations at run-time, which achieves a high compression result on multiple datasets. However, these compression methods require the whole dataset for compression, which greatly increase compression time cost for existed deep model. The other kind of methods focuses on compressing the trained network. For example, Dendi *et al.* [6] use low-rank decomposition of the weight matrix to reduce the number of parameters int he network. In a similar way, Gong *et al.* [14] investigate information theoretical vector quantization to compress the parameters of CNNs. Recently, Han *et al.* [15] introduce a three-stage pipeline including pruning, trained quantization and Huffman coding, to reduce the storage requirement of networks without affecting their accuracy. Most relevant to our work, Chen *et al.* [4] introduce DCT and low-cost hash function to randomly group frequency parameters into hash buckets. Due to the lack of analysis of sparse property, compression results of this approach could be improved.

2.2. Object detection

Category-level object detection, i.e. predicts the bounding boxes and associated class probabilities, has been one of the most active areas in pattern recognition community. Classical methods like DPM [11], often use sliding window and classifier to detect whether objects exist inside the window. More recent methods, like R-CNN [13] first use region proposal to generate potential bounding boxes and then run a classifier on these proposed boxes, which greatly reduce the search space and time cost. Based on object proposal methods, Ouyang *et al.* [17] propose a deformable deep convolutional neural network for object detection, which jointly learn the feature representation and part deformation for a large number of object categories. It outperforms the winner of ILSVRC2014, GoogLeNet [20], by 6.1%. However, the computation cost of this method is high to deploy in embedded systems. To address the problem of computation cost, Redmon *et al.* [18] propose a fast and accurate system named as YOLO, which uses a single neural network to predict bounding boxes and class probabilities. Later on, Redmon *et al.* [19] modify YOLO to improve its speed and accuracy, which gets 78.6 mAP at 40 FPS on VOC 2007 dataset. Our compressed work is built on the modified version of YOLO system.

3. Methodology

In this section, we propose a novel compression method to compress YOLO network by compressive sensing, which decreases the computation and storage cost, and retain the detection accuracy simultaneously. Fig. 1 gives the overview of the proposed method, where (a) refers to the input parameter matrix, which is sampled from the filters of YOLO, (b) use DCT to transform input parameter matrix into frequency domain, (c) pruning frequency matrix, (d) sampling frequency matrix and (e) finetuning the compressed network to get the modified sampling frequency matrix.

3.1. DCT for Parameter Matrix

YOLO network are composed of convolutional layers (COV), batch normalization layers (BN), max-pooling layers (MP), re-organization layers (RO) and detection layer (DT). The parameters of MPs, ROs and DTs are determined manually, while parameters in COVs and BNs require training. Therefore, the proposed method constructs parameter matrix based on parameters of COVs and BNs. Specifically, each normalization layer follows a convolutional layer to improve convergence. In conclusion, YOLO network has 22 COVs and 22 BNs. The size of parameters of each COV varies greatly from hundreds to millions. For each COV, we thus sperate them by blocks determined as 15*15 by experiments. Since each BN has three parameters, there are only 3*22=66 parameters in total for all the BNs. We thus use one 15*15 matrix to store the parameters of BNs. There would be blanks when transforming parameters of filters to blocks. We use the mean value of blocks to fill up these blanks, since the mean value won't change the frequency distribution. After separating and filling blanks, we could get a parameter matrix for each COV and BN by concatenating blocks. The parameter matrix is represented as N_l, where l represents the index of layers varying from 0 to 22. Note that index 0 represents the parameter matrix constructed based on parameters of BNs.

DCT-based methods are widely used to compress images and movies [1]. The main reason for its popularity in compression area lies in three aspects: 1. DCT has the ability to pack most energy of images to the low-frequency part; 2. DCT and its inverse operation could be lossless for compression when employed without quantization or other compression operation; 3. DCT yields a real-valued frequency matrix, compared with FFT whose representation has imaginary components. DCT is suitable for compress of parameter matrix as well, since the weights in parameter matrices are typically smooth and low-frequency, due to the property of spatial locality of image pixels. Given an input matrix N, the corresponding frequency matrix M after DCT could be written as follows:

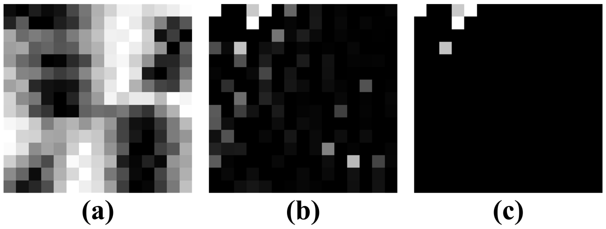


Figure 2. An example of a parameter matrix in spatial domain(a), frequency domain (b) and after pruning (c).

$$M = ANA^T, where\ A(i,j) = c(i)\cos[\frac{(j+0.5)\pi}{d}i] \quad (1)$$

where d is defined as the length of input matrix N and its value is 15 in our method, i and j are the row and column index respectively, $c(i) = \sqrt{\frac{1}{d}}$ when $i = 0$ and $c(i) = \sqrt{\frac{2}{d}}$ when $i \neq 0$. Representing Eq. 1 as f_{dct}, the transforming formula can be rewritten as $M_l = f_{dct}(N_l)$, which converts the parameter matrix from spatial domain to frequency domain.

3.2. Pruning Frequency Matrix

Since compressive sensing requires the input signal to be sparse, in this subsection we prune frequency matrices to make them sparse. Meanwhile pruning could help save computation and storage cost.

Fig. 2 (a) and (b) show an example of frequency matrix in spatial and frequency domain, respectively. In the spatial domain, the frequency matrices are smooth due to the local pixel smoothness in natural images. In the frequency domain, the upper left part with small indices (i, j), known as low-frequency components, have larger magnitude values than other parts named as high-frequency components. Based on this observation, we could conclude the energy of frequency matrix is dominated by low-frequency part. In other words, the upper left frequency values are more important than other values in constructing filters of YOLO. To decrease storage and computation cost and maintain detection results for YOLO, we should prune the high-frequency part and retain the low-frequency part.

Essentially, network pruning has been widely studied to compress CNN models. Early pruning methods [16] were proposed to reduce the network complexity and over-fitting. The proposed method build on top of these approaches by setting an adaptive threshold α for pruning:

$$\begin{cases} M_l(i,j) = M_l(i,j) & if\ M_l(i,j) \geq \alpha \\ M_l(i,j) = 0 & if\ M_l(i,j) < \alpha \end{cases} \quad (2)$$

where l represents layer index and α is defined as

$$\alpha = \gamma \cdot f_{sort}(M_l) \quad (3)$$

Prune Ratio	SNR^1	$PSNR^1$	$mAP^1(\%)$	SNR^2	$PSNR^2$	mAP^2	SNR^3	$PSNR^3$	$mAP^3(\%)$
0.9	51.70	62.44	83.80	41.29	53.17	78.60	10.41	9.27	5.2
0.8	46.70	58.65	82.81	40.29	52.68	74.48	6.41	5.97	8.33
0.7	42.48	53.31	81.88	35.55	47.59	70.45	6.93	5.72	**11.43**
0.6	39.21	49.95	71.94	18.23	30.25	65.91	**20.98**	**19.7**	6.03
0.5	14.64	25.53	64.36	12.42	21.12	59.76	2.22	4.41	4.6
Average	38.95	49.98	76.96	29.56	40.96	69.84	9.39	9.02	7.12

Table 1. Performance comparison between the proposed method and HashedNets [3] on VOC 2012 dataset. Note that the superscripts 1, 2 and 3 correspond to the proposed method, HashedNets and difference between both methods, respectively.

where function $f_{sort}()$ represents the sort operation and γ refers to the prune ratio. Based on different values of γ, users could achieve YOLO systems with different sizes to fulfill there requirement for storage consumption. After removing smaller frequency values, we could get pruned and sparse result as shown in Fig. 2 (c). In fact, pruning mainly occurs in high-frequency components due to the truth that high-frequency components have lower magnitude values. Therefore, pruning doesn't affect the construction of filters in YOLO and decreases computation and storage cost at the same time.

3.3. Compress and Recover by Compressive Sensing

In this subsection, we will discuss how to use compressive sensing [12, 2] to compress and recoverty the sparse frequency matrix M computed by Eq. 2. After that, we will give a short discussion on finetune steps.

A signal x could be expressed as $x = \sum_{i=1}^{N} s_i \Psi_i$, where s_i represents the coefficients and $\{\Psi_i\}$ consist of an orthogonal basis. Compressive sensing focuses on sparse signals in the sense that there exists a basis where coefficients s_i could have just a few large values and many small values. Considering sparse frequency matrix M as a signal, our goal is thus to design a measurement matrix Φ to represent the orthogonal basis vectors $\{\Psi_i\}$.

Specifically, we design Φ that ensures the salient information in frequency matrix M is saved during the dimensionality reduction, i.e. compression. To fulfill this requirement, the construction of Φ shouldn't depend on the frequency matrix M in any way. In other words, Φ should be designed as combinations of independent randomly weighted linear vectors. The distributed random Gaussian matrix [7] has been proved to be a good choice for Φ, since it satisfies all the requirements. The proposed method thus adopts the distributed random Gaussian matrix as Φ, which is defined as $H(i,j) = \frac{1}{\sqrt{m}} h_{i,j}$, where m is the length of Φ and $h_{i,j}$ refers to the normal distribution between 0 and 1. After defining $\Phi = H$, we could compute compressed matrix $C = M \cdot \Phi^{-1}$.

For recovery based on compressed matrix C and measurement matrix Φ, there are many possibilities due to the location of large coefficient components contained in the signal is uncertain. So the recovery solution is often high in computational complexity. We firstly try several classical algorithms for recovery and then choose the fast and accurate one as our recovery algorithm, which is named as approximate message-passing (AMP) [8]. AMP is an iterative algorithm achieving reconstruction performance in one important sense identical to LP-based reconstruction while running dramatically faster.

In running time, the method firstly recovers the frequency matrix M from compressed frequency matrix C by AMP algorithm and then do inverse DCT to convert M from frequency domain back to spatial domain, which could be represented as

$$\tilde{N} = f_{iDCT}(f_{AMP}(M)) \tag{4}$$

After pruning and compressing, the recovery parameter matrix $\tilde{N}$ could be different from the original one. To obtain better performance with $\tilde{N}$, we will finetune the Compressed Network with a few examples.

4. Experiments

To evaluate the proposed method, we consider one benchmark databases, VOC 2012 [10]. VOC 2012 is challenging due to its diversity in object categories and complexity in the layout of images. To measure results of compression, we use SNR, PSNR, mAP. SNR is defined as

$$SNR = 10 \cdot \lg \frac{\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} N(i,j)^2}{mn \cdot MSE} \tag{5}$$

where MSE represents mean squared error defined as $MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \|\tilde{N}(i,j) - N(i,j)\|^2$, and N and $\tilde{N}$ represent the input and compressed parameter matrix, respectively. PSNR is defined as

$$PSNR = 10 \cdot \lg \frac{\max(N)}{MSE} \tag{6}$$

where $\max(N)$ refers to the maximal value in input parameter matrix. When comparing between two methods, the performance is considered better if SNR, PSNR and mAP

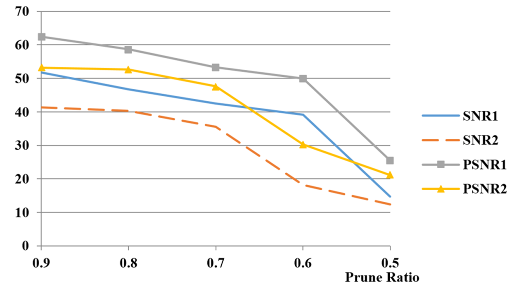


Figure 3. Comparison of SNR and PSNR between our method and HashedNets, which are represented by label 1 and 2.

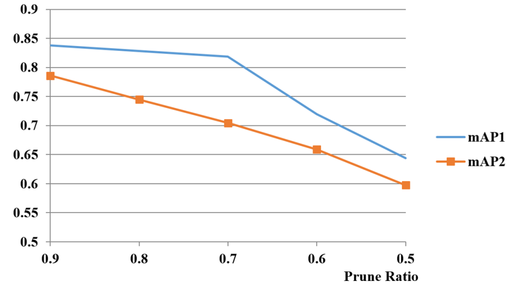


Figure 4. Comparison of mAP between our method and HashedNets, which are represented by label 1 and 2.

are larger values. In fact, SNR and PSNR are two measurements to judge the loss of parameter matrix during compression, while mAP focuses on the detection performance of YOLO.

Table. 1 gives the detailed statics of the proposed method and HashedNets [3] for the VOC 2012 dataset, measured on a PC with 2.5GHz i7 CPU and 6GB RAM. HashedNets is a novel network architecture to reduce and limit the memory overhead of neural networks. To compare with HashedNets, we implement its algorithm according to the instructions in paper. Note that the prune ratio in Table. 1 refers to the compression ratio in our experiment. Our method could slightly decrease the computation cost at 10% when the prune ratio is set at 60%. It is evident by average value of SNR^3, $PSNR^3$ and mAP^3 that the proposed method achieves better results than HashedNets in terms of three measurements for VOC 2012 database. The comparison advantage of the proposed method is more obvious when the prune ratio is set as 0.6, where the proposed method gets far better performance than HashedNets in SNR and $PSNR$. However, the corresponding difference of mAP is only 6.03. On the contrary, the largest difference of mAP value 11.43 appears when the prune ration is set at 0.5. This inconsistent performance between SNR, PSNR and mAP is caused by the fact that the performance of YOLO is affected by many factors including parameters and the relation between parameters and detection accuracy is non-linear.

We show the comparison of SNR, PSNR and mAP between the proposed method and HashedNets in Fig. 3 and 4, respectively. We could find that our method gets stable performance in SNR and PSNR when the prune ratio decreases from 0.9 to 0.6, while HashedNets gets stable performance between 0.9 and 0.7. The mAP value of our methods drops greatly when the prune ratio decreases from 0.7 to 0.5, while HashedNets is stable is mAP during compression. The unstable performance of the proposed method could be explained by the fact that mAP would drop fast if the proposed method prunes the low-frequency part of parameter matrix.

After compression, we shift YOLO to mobile phones and test its performance. Sample qualitative results of the shifted YOLO are shown in Fig. 5, where the first and second row represent detection results of real-life scene images and images in VOC 2012 dataset. From these sample results, we can see the shifted version of YOLO could detect object accurately and fast, even facing challenges of diversity in object categories and complexity in the layout of images.

5. Conclusion

In this paper, we propose a novel method to compress YOLO network by compressive sensing, result in decreasing the computation and storage cost. After converting parameter matrix of YOLO network to frequency domain by DCT, we could achieve a low-frequency dominated matrix due to the inherent smooth property of parameters when processing images. Then, we prune high-frequency components to make the frequency matrix sparse. Next, we sample the sparse frequency matrix with distributed random Gaussian matrix. Finally, we retrain the network to finetune the remaining parameters. Experiment results on VOC 2012 show that the proposed method outperforms a relevant baseline method. We also show examples of object detection results after shifting compressed YOLO to mobile phones. Our future work includes the exploration on compressing other types of CNN and RNN network with the proposed method.

References

[1] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE Trans. Computers*, 23(1):90–93, 1974. 1, 3

[2] R. G. Baraniuk, V. Cevher, M. F. Duarte, and C. Hegde. Model-based compressive sensing. *IEEE Trans. Information Theory*, 56(4):1982–2001, 2010. 4

[3] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen. Compressing neural networks with the hashing trick. In *ICML 2015*, pages 2285–2294, 2015. 4, 5

[4] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen. Compressing convolutional neural networks in the

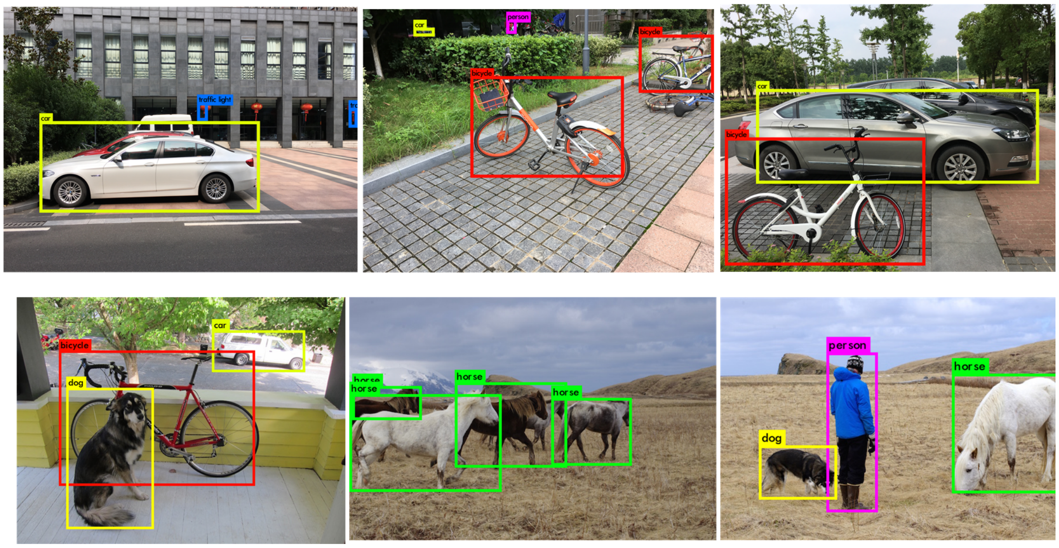


Figure 5. Detection examples of the proposed method on real-life scene images and images from VOC 2012 dataset.

frequency domain. In *SIGKDD 2016*, pages 1475–1484, 2016. 2

[5] M. Courbariaux and Y. Bengio. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. *CoRR*, abs/1602.02830, 2016. 2

[6] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. de Freitas. Predicting parameters in deep learning. In *NIPS 2013*, pages 2148–2156, 2013. 2

[7] T. T. Do, T. D. Tran, and L. Gan. Fast compressive sampling with structurally random matrices. In *ICASSP 2008*, pages 3369–3372, 2008. 4

[8] D. L. Donoho, A. Maleki, and A. Montanari. Message passing algorithms for compressed sensing. *CoRR*, abs/0907.3574, 2009. 4

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html. 1

[10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. 4

[11] P. F. Felzenszwalb, R. B. Girshick, D. A. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1627–1645, 2010. 1, 2

[12] S. Foucart and H. Rauhut. *A Mathematical Introduction to Compressive Sensing*. Applied and Numerical Harmonic Analysis. Birkhäuser, 2013. 4

[13] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR 2014*, pages 580–587, 2014. 1, 2

[14] Y. Gong, L. Liu, M. Yang, and L. D. Bourdev. Compressing deep convolutional networks using vector quantization. *CoRR*, abs/1412.6115, 2014. 2

[15] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR*, abs/1510.00149, 2015. 1, 2

[16] S. J. Hanson and L. Y. Pratt. Comparing biases for minimal network construction with back-propagation. In *NIPS 1988*, pages 177–185, 1988. 3

[17] W. Ouyang, X. Zeng, X. Wang, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, H. Li, K. Wang, J. Yan, C. C. Loy, and X. Tang. Deepid-net: Object detection with deformable part based convolutional neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(7):1320–1334, 2017. 1, 2

[18] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR 2016*, pages 779–788, 2016. 1, 2

[19] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016. 1, 2

[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. 2

[21] L. Wang, Y. Wu, T. Lu, and K. Chen. Multiclass object detection by combining local appearances and context. In *ACM MM 2011*, pages 1161–1164, 2011. 1