Design and Implementation of the Vehicular Camera System using Deep Neural Network Compression

Beomjun Kim, Yongsu Jeon, Heejin Park, Dongheon Han, Yunju Baek
Pusan National University
San-30, Jangjeon-dong,Geumjeong-gu
Busan, Republic of Korea
{beomjun.kim, yongsu.jeon, heejin.park, dongheon.han}@eslab.re.kr, yunju@pusan.ac.kr

ABSTRACT

In recent years, there is a growing interest in advanced driver assistance systems, which can reduce the risk of accidents on various roads. Many vehicular technologies use camera information to collect roadside information. But research and development of image recognition in embedded environments is challenging. Therefore, there is a requirement for research to apply open source image recognition technology to embedded platform. Deep learning, a technology that has been on the spotlight recently, shows excellent performance in image recognition. However, deep learning has a problem that the network size and amount of computation are too large.

In this paper, we design and implement a deep learning based object recognition system to recognize vehicles on the road. We recognize vehicles using Faster-RCNN, which has excellent object recognition capabilities. However, it is problematic to apply it to an embedded device due to the feature of deep learning. Therefore, we propose and evaluate the performance of object recognition with quantization and pruning. Through the evaluation, we will show that the proposed system reduces the network size to 16% and reduces the operating time to 64% without a significant decrease in recognition accuracy.

Keywords

Embedded Deep Learning; Pruning; Quantization

1. INTRODUCTION

Smart automotive technology has been greatly developed with the help of various telecommunication technologies. As society has developed, there are many vehicles on the road and the road condition is getting more complicated with growing risk of accidents. Therefore, Advanced Driver Assistance Systems (ADAS) for the safety of the driver has been studied [2, 3]. In most cases, ADAS devices use cameras or various sensors to obtain surrounding information. However, it is not easy to recognize the surroundings through the camera information in the vehicle environment. Most image recognition techniques have limitations in operation in embedded devices due to the complexity of recognition algorithms. In recent years, deep learning based image recognition technologies have shown high performance. For example, the Faster-RCNN [12], which is a representative technology, has excellent object recognition performance. In addition, the source code of Faster-RCNN is shared open source and is open to many users. We use Faster-RCNN for the proposed system for good image recognition performance. And, we use ZF [15] as a training model for Fater-RCNN. When ZF is used, Faster-RCNN extracts the features of the image from the five convolution layers and constructs the feature vector. The feature vector is fully connected layer via an Region Proposal Network (RPN) that can detect the area of the object. In the fully connected layer, it judge the area and kind of object. The last two fully connected layers have connection weights of over 90% of the total.

In this paper, we propose a deep learning based camera system in a vehicle environments. Since the proposed system operates in an embedded environment, we try to reduce the connection weight of the fully connected layer to reduce the size of the learning result data. In addition, we intend to increase the recognition speed by reducing the amount of computation of the fully connected layer occupying the most computation time. We implemented our system in the embedded platform to evaluate the performance of it. We have confirmed the performance of the proposed system by applying two well known techniques.

The contents of this paper are as follows. In Chapter 2, we will briefly discuss background information on related technologies. Chapter 3 introduces the structure of the proposed system and two weight-reduction techniques for deep learning. In Section 4, performance evaluation of the proposed system is performed. Finally, Chapter 5 presents conclusions and future research topics.

2. BACKGROUND

In this chapter, ADAS for driver safety and existing deep learning acceleration technique and Faster-RCNN, a object detection algorithm used in the proposed system, will be briefly described.

2.1 ADAS

ADAS is technology developed for cars to drive the road more safely. Along with the development of the Information Technology (IT) industry, technology to recognize the surrounding situation to prevent traffic accidents using ultra-

EMDL'17, June 23, 2017, Niagara Falls, NY, USA

DOI: http://dx.doi.org/10.1145/3089801.3089803

sound, infrared or image sensor have been developed. The system that recognizes the situation around the road is divided into a method using a sensor and a method using a camera. Although the method of using the sensor can measure the distance between the vehicles relatively accurately and simply, there is a disadvantage that the price of the sensor mounted on the vehicle is expensive. In the case of using a camera, it is somewhat less accurate, but recognizes various objects, and the installation cost is also relatively inexpensive. Because of this feature, many camera based ADAS technologies have been studied and several results have been produced. Recently, the study of ADAS technology based on cameras has been developed using deep learning.

2.2 Deep learning acceleration technique

Deep Neural Network (DNN) has a disadvantage of high computational complexity due to a large number of connection weights. To solve this problem, many researches have been conducted on acceleration of deep learning.

An example is a research that has achieved good performance by converting the fully connected layer with a large number of connection weights into a global average pooling layer [10]. In addition, SqueezeNet [9] reduced the number of connection weights by a factor of 50 by converting the convolution layer with a 3x3 kernel into a convolution layer with a 1x1 kernel. ResNet [8] has reduced the computational complexity and improved the accuracy by using a residual function that skips over multiple layers and links the previous input to the later output. Vanhoucke [14] converted the connection weights to 8-bit integers and obtained about twice the performance improvement by using the Single Instruction Multiple Data (SIMD) instruction of the CPU. Rastegari, M [11] reduced the memory usage by 32 times and increased the computation speed by 58 times by only configuring the connection weights as -1 and 1. Han's research [5, 6] has reduced the size of the learning result data by more than 90% by eliminating the connection weights that are smaller than the threshold value. Also, Bhattacharya's research [1] is addressing deployment of deep neural networks on embedded platforms that uses sparse matrix representation techniques to lower memory requirements of DNNs/CNNs.

2.3 Faster-RCNN

Faster-RCNN is an object detection algorithm that is a winner in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [13]. Unlike existing object detection algorithms [4, 7], Faster-RCNN overcomes the existing limitations by adding RPN in one multi-artificial neural network and integrated learning and object detection into one. The RPN receives the final feature map from the convolution layer and performs a sliding window convolution operation to generate a 256-dimensional or 512-dimensional vector. In each sliding window, a total of k object candidates are recommended. Each candidate consists of a combination of sizes and ratios based on the center of the sliding window. The vector generated by the RPN is connected to a box classification layer, which indicates the presence or absence of an object, and a box regressor layer, which forms the coordinates of the candidate region, to obtain the final object position. Faster-R CNN supports various training models. In this paper, we use ZF which is suitable for embedded platform as training model.

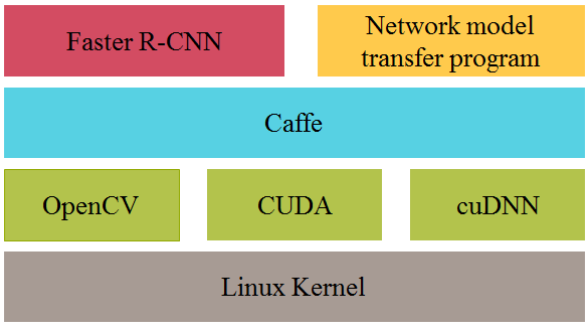


Figure 1: Software architecture of system

3. VEHICULAR CAMERA SYSTEM

We proposed a vehicular camera system to develop a technology for image sensor based smart cars. The proposed system analyzes the surrounding vehicles in real time using video. First, we explain the architecture of the proposed system and show how to modify the neural network to operate smoothly using embedded platform.

3.1 Architecture

In this paper, we implemented the system and measured the deep learning performance through NVIDIA 's Tegra X1 System on chip (SoC) based embedded platform. The operating system of the proposed system is Linux, and we installed the Compute Unified Device Architecture (CUDA) library, which enables us to use NVIDIA's GPU, and cuDNN, which can accelerate the deep learning operation using NVIDIA's GPU. We also installed the Caffe framework, which facilitates the use of deep learning, to run Faster-RCNN. Figure 1 shows the software architecture for implementing the functions of the vehicular camera system.

Deep learning can continuously improve performance. It allows the smart camera to receive the trained network from the server. There are three steps between the server and the vehicle to communicate over broadband communication. Each step is represented in Figure 2. First, the server learns

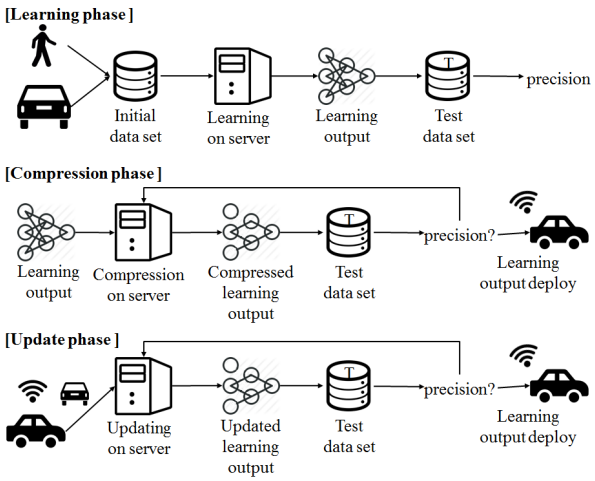


Figure 2: Process of proposed system

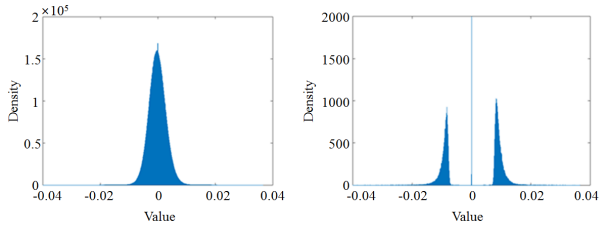


Figure 3: Change in weight values after applying pruning

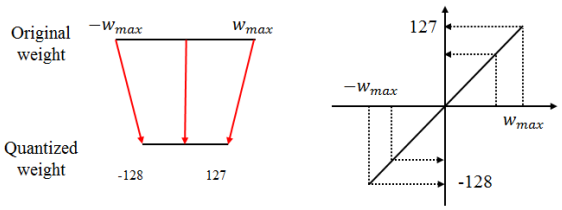


Figure 4: The process of linear quantization of weights

from the data and outputs the first result. Learn the shape of the vehicle through the learning data and verify the accuracy through the test data set. At this stage, the server reduces the size of the network, making it suitable for operation on an embedded platform and allowing data to be transmitted over broadband communications. If the network has a certain level of accuracy, it is transmitted to the smart camera of the vehicle through broadband communication. The vehicle periodically transmits the road image to the server. On the server side, the image received from the vehicle is added to the learning data, and then the learning is performed using it. Through this process, the accuracy of image recognition can be improved.

3.2 Pruning

In this chapter, we explain how to delete non-critical weight values through the pruning process used in this paper. In this process, we find appropriate thresholds for the data we are learning, rather than using a fixed threshold. First, the ratio of the values whose absolute values are smaller than the threshold value in one layer is calculated. Perform the learning process again so that the pruned weights do not significantly affect the accuracy. This process is repeated three times to ensure that the accuracy is not significantly reduced by pruning large quantities of weight values.

In our proposed system, pruning has two purposes: to reduce the memory required to store connection weights and to increase the computational speed. To save memory space required to store weights, pruned weights are stored as sparse matrices. The size of weights with most values set to 0 is reduced when expressed as a sparse matrix. Sparse matrices are expressed in Compressed Sparse Row (CSR) format among various expressions. In order to represent a sparse matrix using a smaller amount of space than a general matrix, at least half of the matrix values must be zero. By expressing the weights as a sparse matrix, the computation speed can be improved by using an algorithm different from the multiplication between general matrices. Multiplication between matrices in Faster-RCNN is implemented in cuBLAS, a linear algebra library in CUDA. In this paper, we perform sparse matrix multiplication using cuSPARSE, a sparse matrix library of CUDA. The distribution of the weighted distribution in FC6 after pruning is shown in Figure 3.

3.3 Quantization

The weights required for the deep learning operation are stored as 32-bit real numbers and the deep learning algorithm operates through the calculation of real numbers. However, when performing inference without learning, the precision of the weight is not very important [11]. Using this feature, many methods of transforming the values of connection weights into various types instead of real numbers have been researched. In this paper, we convert 32 bit real numbers to 8 bit integers to save the size of learning result data and speed up the computation speed. There are two ways to convert a 32-bit real number to an 8-bit integer: linear conversion and nonlinear conversion. When nonlinear methods are used, the information space of integers can be maximized to minimize information loss. However, when nonlinear transformation is performed, it is slower than linear transformation in the process of realization after multiplication between integer matrices. In order to minimize the non-precision problem of the linear transformation, the output weights of the FC6 layer are quantized from 0 to the maximum value. As shown in Figure 4, the original weight values are converted to 8-bit integers. The output of the FC6 layer is connected to ReLU and does not have a negative value. The realization of the linear transformation can be defined by a simple formula. In the embedded system used in the proposed system, there are video instructions which function as SIMD. This instruction can process a single 32-bit operand at a time, with one 32-bit integer, two 16-bit integers, or four 8-bit integers. When this instruction is used to perform multiplication between matrices, the multiplication between 8 bit integer matrices is faster than the multiplication between 32 bit real matrices.

4. EVALUATION

In order to evaluate the proposed system, performance tests of pruning and quantization were performed on the embedded platform. In this section, we will describe the evaluation environment and analyze the results of each experiment.

4.1 Environments

In our proposed system, vehicle side and rear photographs were used to evaluate the target detection performance. If the rear and side of the vehicle can be recognized, it is helpful to implement ADAS such as collision avoidance system and detection of the surrounding traffic. We collected black box

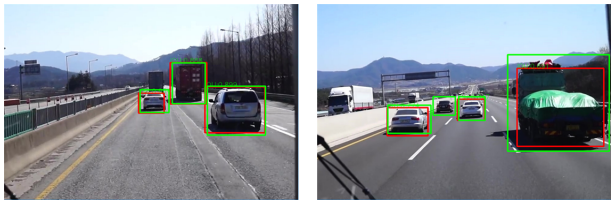
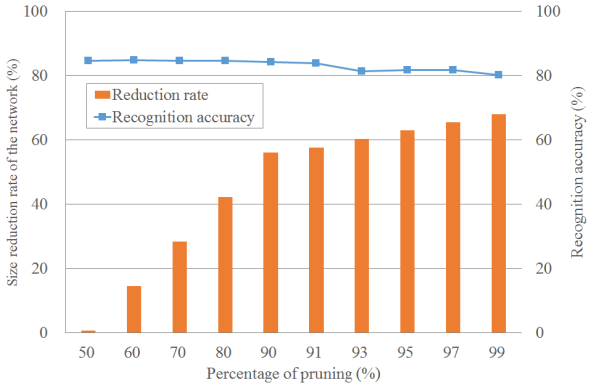
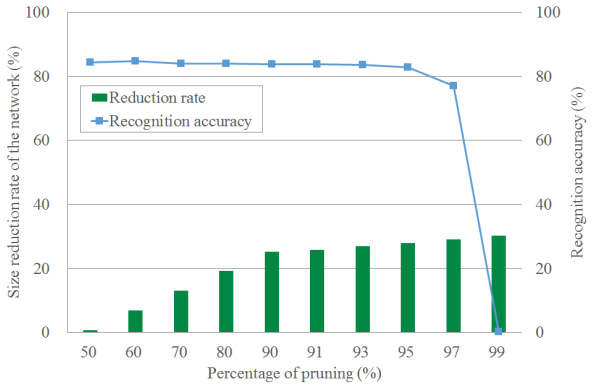


Figure 5: Data set for evaluation

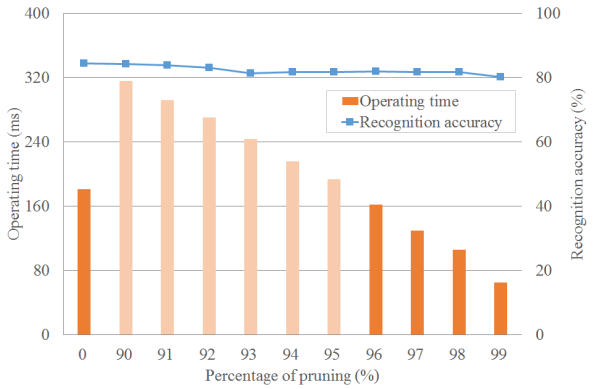


(a) Apply pruning to the FC6 layer only

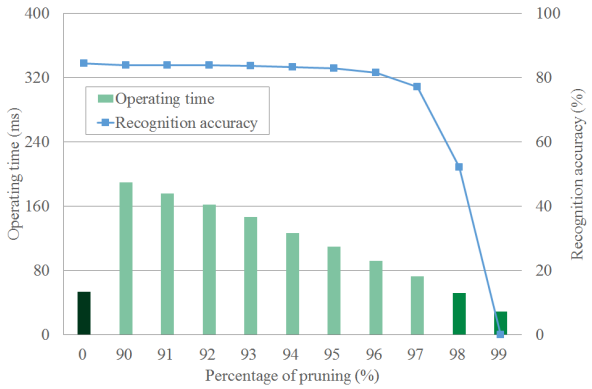


(b) Apply pruning to the FC7 layer only

Figure 6: Influence of pruning on total size of network and recognition accuracy



(a) Apply pruning to the FC6 layer only



(b) Apply pruning to the FC7 layer only

Figure 7: Influence of pruning on operating time and recognition accuracy

videos for evaluating object detection and extracted 2,929 images from the video with different weather and time zones. The vehicle images were divided into two groups, 2,000 images used as training data sets, and the remaining 929 images left as test sets. An example of a vehicle image data set is shown in Figure 5. In order to evaluate the recognition performance of the system, we made x86 program which is easy to create correct data for all pictures manually. The NVIDIA Tegra X1 device used in the experiment has a Maxwell 256-core GPU, a 4-core Cortex CPU, and 4GByte of LPDDR4 memory. The learning of vehicle image data was done on a server computer using the GTX1080 as a GPU, and the average learning time was 4 hours and 30 minutes. The inference time shown in the experiment of this paper was measured on the embedded platform, which took up to 10 times longer than the server computer.

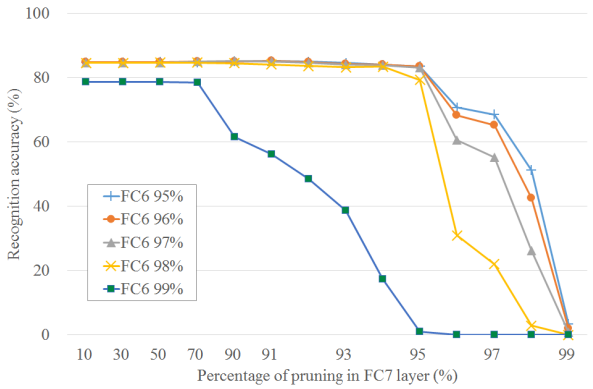
4.2 Performance Results

4.2.1 *Impact of pruning on data size*

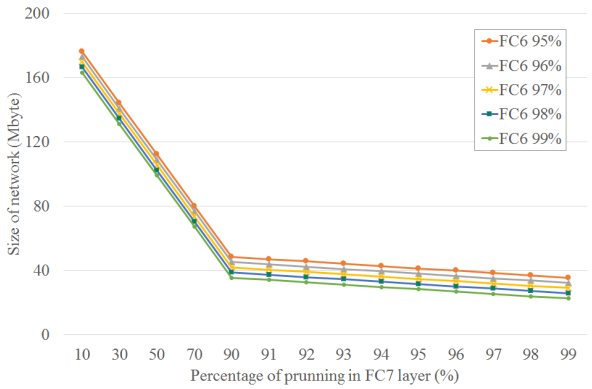
Theoretically, pruning more than half of the connection weights can save memory space, but it is less accurate. We could confirm this through actual experiments, and the experimental results had different characteristics depending on the layer. Figure 6 shows the size reduction rate of the network according to the degree of pruning in the fully connected layer. As suggested by Figure 6a, FC6 layer showed no significant difference in accuracy even when many connection weights were removed. At this time, the size of the learning result data of FC6 decreased from 260 Mbyte to 87 Mbyte which is about 40%. In the FC7 layer, the accuracy was drastically lowered when too much pruning was performed (Figure 6b). The FC7 layer is more influenced by pruning because it has a smaller number of connection weights than the FC6 layer and is the last layer. In order to ensure the recognition accuracy, the pruning ratio in FC7 should be adjusted appropriately. When pruning the FC7 layer connection weights by 96%, the accuracy is reduced by 3% and the size is reduced to 166 Mbytes.

4.2.2 *Impact of pruning on computation time*

If the degree of pruning is high as shown in Figure 7, the computation speed becomes faster due to the difference in characteristics between the sparse matrix and the general matrix. In general, performance is better with multiplication of sparse matrices than our experimental results. Since our embedded device does not support multiplication of GPU-optimized sparse matrices, it can not be performed efficiently. Nevertheless, pruning at a high rate has reduced the operating time of each fully connected layer. The FC6



(a) Accuracy of recognition when pruning both layer



(b) Total size of network when pruning both layer

Figure 8: Performance when pruning are applied at FC6 and FC7

layer operates faster than the existing operation when the pruning is 96% or more, and the FC7 layer operates efficiently when the pruning is 98% or more. In the Figure 7, dark color is used only for efficient operation.

4.2.3 *Impact of pruning on both layers*

As shown as Figure 8, we applied pruning to both layers at the same time to find a way to apply pruning appropriately. When pruning the FC6 layer to 96% or more, the computation speed is improved. Therefore, the ratio of FC6 is increased from 96%. When the FC6 layer is less than 97% and the FC7 layer pruning is less than 96%, the recognition accuracy does not drop significantly (Figure 8a). Figure 8b shows the size of network generated when pruning is applied to two layers. The size of the network data is reduced in proportion to the degree of pruning. As a result of examining several graphs, we found that FC6 layer is more suitable for pruning than FC7 layer. Performing pruning on the FC7 layer will result in damage in terms of operating speed or recognition accuracy.

4.2.4 *Impact of quantization*

Table 1 shows the size of the network and the operating speed of recognition according to the quantization. Quantization on both layers reduced the computation time and size of the network. However, the two layers have different rates of reduction in recognition accuracy. Quantization of FC6 results in a significant reduction in accuracy, but FC7 has the advantage of quantization because of similar accuracy. The FC6 layer has a problem that it is not accurate enough to be used.

Table 1: Performance comparison through quantization

	FC6		FC7	
	Before	After	Before	After
Accuracy (%)	84.5	57.0	84.5	83.6
Time (ms)	181.1	64.4	53.7	30.0
Network (Mbyte)	225	134	225	184

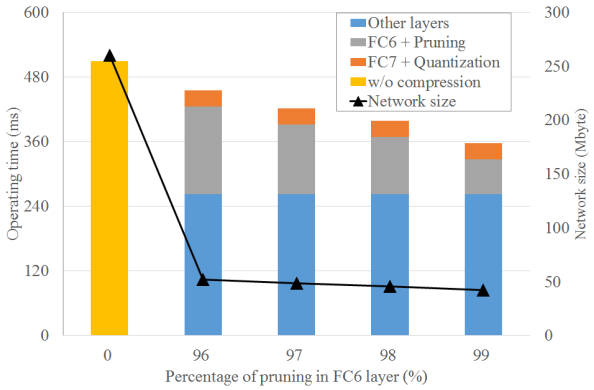


Figure 9: Performance when using both methods

4.2.5 *Performance experiment with both methods*

Figure 9 shows the total operating time and network size when both quantization and pruning are applied. Since quantization and pruning apply only to the fully connected layer, they do not affect the operating time of the other layers. FC7 improves the computation speed and reduces the size of the network through quantization, and FC6 achieves the same effect through pruning. By applying both techniques, the operation speed is reduced from 508 ms to 327 ms, and the size of the network is reduced from 260 Mbyte to 42 Mbyte.

5. CONCLUSIONS

In this paper, we propose a vehicle camera system that can acquire images and recognize other vehicles using camera installed on the front of the vehicle. We performed pruning and quantization for DNN based image recognition on embedded platform. DNN has excellent performance to recognize objects using image information, but it is not suitable for embedded system due to many connection weights. We performed performance evaluations on quantization and pruning for the proposed embedded system. Through these methods, we were able to effectively change the behavior of the fully connected layer, which accounts for half of the total computation time. As a result of the evaluation, the

size of the network was reduced to about 42 Mbyts, and the operation time was also reduced to about 327 ms. In the future, we plan to research and develop smart cameras for use in real environments. We will also try to optimize multiplication of sparse matrices on the embedded platform.

6. ACKNOWLEDGMENTS

This work was supported by the Center for Integrated Smart Sensors funded by the Ministry of Science, ICT & Future Planning as Global Frontier Project (CISS-2011-0031863), Corresponding author: Yunju Baek (yunju@pusan.ac.kr)

7. REFERENCES

[1] S. Bhattacharya and N. D. Lane. Sparsification and separation of deep learning layers for constrained resource inference on wearables. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, SenSys '16, pages 176–189, New York, NY, USA, 2016. ACM.

[2] S. Borhade, M. Shah, P. Jadhav, D. Rajurkar, and A. Bhor. Advanced driver assistance system. In *Sensing Technology (ICST), 2012 Sixth International Conference on*, pages 718–722. IEEE, 2012.

[3] L. D. Burns. Sustainable mobility: a vision of our transport future. *Nature*, 497(7448):181–182, 2013.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[5] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[6] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, pages 1135–1143, 2015.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision*, pages 346–361. Springer, 2014.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[9] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[10] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[11] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[12] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[13] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[14] V. Vanhoucke, A. Senior, and M. Z. Mao. Improving the speed of neural networks on cpus. In *Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop*, volume 1, page 4. Citeseer, 2011.

[15] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.