2018 IEEE 16th Int. Conf. on Dependable, Autonomic & Secure Comp., 16th Int. Conf. on Pervasive Intelligence & Comp., 4th Int. Conf. on Big Data Intelligence & Comp., and 3rd Cyber Sci. & Tech. Cong.

The Implementation of CNN-based Object Detector on ARM Embedded Platforms

Yingjie Zhang, Sheng Bi*, Min Dong and Yunda Liu

School of Computer Science and Enginerring
South China University of Technology
Guangzhou, China
picy@scut.edu.cn, csyjzh@mail.scut.edu.cn

***Abstract*—Deep convolutional neural network (DCNN) based computer vision methods have great progress in object detection tasks. And object detection is essential to many applications such as autonomous robots and vehicles. However, it is difficult for object detection system to be deployed on embedded platforms due to its intensive computing architecture. We present an object detector MobileNet-SSD which can be deployed on embedded platforms. The object detector presented in this work based on Single Shot Detector (SSD) framework and replace the feature extractor with a more light weight network MobileNetV1. General matrix to matrix multiplication is used to simplify the calculation of convolution operation and is further optimized using ARM NEON technology. Finally, the object detector MobileNet-SSD is deployed on embedded platform and achieves 1.13FPS with 0.72 mean average precision on PASCAL VOC dataset.**

Keywords—object detection, convolutional neural network, embedded system, NEON optimization

I. INTRODUCTION

Object detection is an important computer vision technology used in many applications such as autonomous robots[1] and vehicles[2]. Deep convolutional neural networks (DCNNs) have made a lot of progress in the field of computer vision especially on the task of object detection in recent years. As a result, a lot of object detection networks, such as Single Shot Detector (SSD)[3], Faster R-CNN[4], You Only Look Once (YOLO)[5] and DeepLab[6], are accurate enough to be used in real-life detection applications. In almost all of these networks, a convolutional feature extractor need to be applied to the input image to obtain high level features. For example, VGG[7] is used in SSD and Resnet[8] is used in Faster R-CNN.

However, VGG and Resnet are not suitable for many embedded applications because of their requirements of high computational resources. It is a daunting task to deploy one of these object detectors on mobile or embedded device due to its limited computation resources. In order to achieve the goal of deploying to embedded devices, the object detection networks are expected to be not only accurate enough but also fast and with small size. Several efficient mobile neural networks for common object recognition have been proposed in the year recently, such as SqueezeNet[9], MobileNetV1[10], ShuffleNet[11], and MobileNetV2[12]. Therefore, it is a straight-forward way to use these accurate light weight CNNs as feature extractor to extract high level features for object detectors such as SSD.

This paper gives an overview of an object detector implementation on ARM embedded platform. An light weight object detector network is proposed, and implemented on an ARM embedded platform. The general matrix to matrix multiplication (GEMM) is used for the convolutional layer in object detector network and is optimized using ARM NEON technology.

The rest of this paper is organized as follows. Section II describes the work related to deployment of CNNs on embedded platforms. In Section III, the implementation of an object detector is presented. The deployment results on a specific ARM embedded platform are reported in Section IV. Finally, Section V concludes the paper and elaborates on the future work.

II. RELATED WORK

The deployment of CNNs on embedded platforms has attracted many researcher to study. Many different approaches can be generally categorized into either hardware optimizing or light weight CNN designing.

Due to the high computational complexity of CNNs, there has been a lot of work related to specific hardware design [13], [14], [15], [16], [17] for CNN and CNN-based computer vision tasks. In [13], Chen et al. proposed the DianNao, which achieved high throughput and low energy consumption when processing CNNs, by carefully optimizing the impact of memory on accelerator design, performance and energy consumption. In [14], Farabet et al. implemented an efficient CNN as well as an efficient CNN-based face detection system, using a single Field-Programmable Gate Array (FPGA) with an external memory module. In [15], Zhang et al. proposed a CNN accelerator on FPGA platform using many optimizing techniques, such as loop tiling and transformation.

As a contrast, designing a light weight CNN and implementing it on commodity embedded hardware has been considered as an alternative solution. Despite the fact that almost all the object detection networks are designed for desktop platform, there are many feature extractor [9], [10], [11], [12], specially designed for embedded and mobile platform. In [9], Forrest N. Iandola et al. used a bottleneck approach to design a very small but efficient network, which achieves AlexNet-level accuracy on ImageNet with much fewer parameters and can be compressed to less than 0.5MB. MobileNetV1 using depthwise separable convolutions is presented in [10], which is efficient for embedded vision applications and has two simple hyper-parameters that can efficiently trade off between latency and accuracy. In [11], ShuffleNet utilizes channel shuffle operation and pointwise

Yingjie Zhang, Sheng Bi*, Min Dong and Yunda Liu are with the South China University of Technology, Guangzhou, 51006. (Corresponding author: Sheng Bi, e-mail:picy@scut.edu.cn)

978-1-5386-7518-2/18/$31.00 ©2018 IEEE
DOI 10.1109/DASC/PiCom/DataCom/CyberSciTec.2018.00074

group convolution to reduce computation cost and achieve higher accuracy than MobileNetV1. And, MobileNetV2 further improves the performance of light weight CNNs due to the inverted residual structure with linear bottleneck in [12].

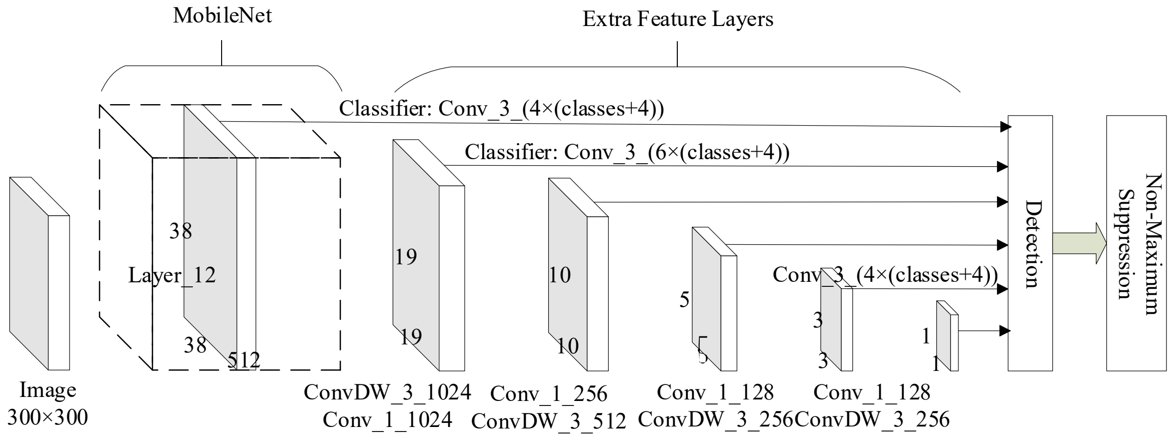

Figure 1. MobileNet-SSD

III. Implementation Of Object Detector

In this section, we firstly describe a light weight object detector network using SSD framework. And, the general method to calculate convolution operations is introduced in the second part of this section. Finally, the reasons for us to use NEON intrinsic to optimize the general matrix to matrix multiplication are presented.

A. The object detector MobileNet-SSD

SSD as one of the most popular object detectors, can feasibly detect objects of various sizes by combining anchor box proposal system of Faster R-CNN and using multi-scale features to do detection.

We design an object detector structure MobileNet-SSD using SSD framework, which is similar to original SSD-VGG-300 framework. The main difference is that rather than using VGG network, now the backbone for feature extractor is MobileNetV1. And also, the convolutional layers added in extra feature layer are replaced with depthwise separable convolution which is used in MobileNetV1 to reduce the computation complexity of convolution.

The classifiers in MobileNet-SSD use multiple feature layers, where each feature map is evaluated by a set of different (aspect ratio) default boxes at each location in the feature map, and each classifier predicts class scores and position offset relative to the default boxes.

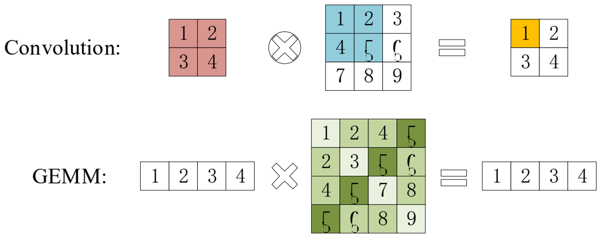

Figure 2. Convolution to GEMM

B. Convolution to general matrix to matrix multiplication

It is an obvious choice to use GEMM for the convolutional layer, since CPUs can use parallelization techniques such as Single Instruction Multiple Data (SIMD) to perform the multiply-and-accumulate operations in parallel.

Fig. 2 shows how a matrix multiplication is used for the convolutional layer. The first step is to turn the input image into a 2-D array that we can treat it like a matrix. We take each one of cubes of input image which will be applied to each kernel as a single column into the result matrix. This is known as image-to-column. Then, the kernel's weights will be transformed as the same way. After the matrix multiplication, the result of convolutional layer can be easily get.

However, the expansion in memory size will be caused by image-to-column operation, which means that pixels that included in overlapping kernel sites will be duplicated in the matrix. It seems inefficient but, image-to-column operation is still widely used because matrix to matrix multiplications are more suitable to understand and optimize than general convolutional operations.

C. NEON optimization for object detector

ARM NEON technology is an advanced SIMD architecture for the ARM Cortex-A series processors which is developed for the acceleration of multimedia processing algorithms such as image processing. In this paper, NEON technology is used to optimize the performance of object detector especially to optimize general matrix to matrix multiplication for convolutional layers. There are four ways to use NEON technology in programming: NEON optimized libraries, vectorizing compiler, NEON intrinsic, NEON assembly.

NEON optimized libraries cannot used in the application of object detector due to the lack of convolution implementation in open source project such as Ne10. We also do not utilize vectorizing compiler because of the unpredictable behavior of vectorizing compiler.

The reasons for us to use NEON intrinsic to optimization the object detector rather than NEON assembly are given below.

- Portability: Due to the different Instruction Set Architectures (ISAs) used in ARMv7-A/ARMv8 AArch32 and ARMv8 AArch64, NEON assembly codes need to be rewrite in order to run at all the platforms. In contrast, if the codes are programmed using NEON intrinsic, they can run across different platforms directly.
- Maintainability: NEON assembly is more demanding to read and write than NEON intrinsic, whether NEON assembly is write in assembly files or inline assembly. NEON intrinsic is similar to C code which significantly reduces the maintenance effort.
- Performance: NEON assembly can show better performance for the specified platform after careful fine-tune. It is time consuming, however, and NEON assembly need to be optimized for all the platforms. But, NEON intrinsic can meet the requirements of the application due to the development of modern compilers. By using NEON intrinsic, the benefits of hardware and compiler upgrade can be obtained without any reprogramming.

IV. Deployment on Embedded Platform

We train the MobileNet-SSD object detector with Caffe[18], which is a popular framework for deep learning. The dataset we use in training phase is PASCAL VOC[19], and the MobileNet-SSD object detector achieves the mean average precision (mAP) of 0.72 on the dataset PASCAL VOC after training. The pretrained model of MobileNetV1 is used, which make the training phase more easier than we train MobileNet-SSD from scratch.

After training, the trained model of MobileNet-SSD object detector is deployed on the NanoPi2 module. The NanoPi2 is a high performance ARM board developed by FriendlyARM. And NanoPi 2 uses the Samsung Cortex-A9 Quad Core S5P4418@1.4GHz SoC and 1GB 32bit DDR3 RAM.

During the deployment, the convolutional layer (include depthwise separable convolution layer) in MobileNet-SSD is transformed into general matrix to matrix multiplication and further optimized using ARM NEON intrinsic as described in Section III.

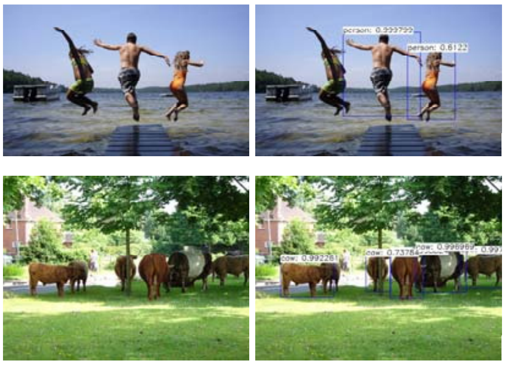

Figure 3. Detection result

Fig. 3 shows some detection results of MobileNet-SSD, which running on NanoPi2 module. Note that, the result of first image is not ideal because of the shadow on the body of the left-most person. Another reason of this result is that there is a tradeoff between computation complexity and accuracy in design of MobileNet-SSD object detector.

MobileNet-SSD, after deployed on NanoPi2, can run at 1.13FPS. The speed is not very fast, because the hardware NanoPi2 we used only contains Cortex-A9 cores with general performance. However, the MobileNet-SSD does improve the inference speed by reducing computation complexity. We haven't run the model SSD-VGG-300 on NanoPi2 because the number of weights of SSD-VGG-300 is very large so that the memory of NanoPi2 cannot satisfy this requirment. But the speed of SSD-VGG-300 will be much slower than MobileNet-SSD due to its computational complex feature extractor.

V. Conclusion and Future Work

In this paper, an object detector MobileNet-SSD is proposed and deployed on embedded platform NanoPi2. The MobileNet-SSD uses MobileNetV1 as the feature extractor which is a light weight CNN architecture designed for mobile and embedded application. General matrix to matrix multiplication is used to simplify the convolution operation. And ARM NEON technology is used to optimize GEMM. After the deployment of MobileNet-SSD on NanoPi2, MobileNet-SSD achieves the trade-off between speed and accuracy with 1.13FPS.

Another interesting aspect of this work that can be explored in the future would be to replace GEMM with winograd algorithm for convolutional layer to further reduce the computation complexity.

Acknowledgment

This research work is supported by Guangdong province science and technology plan projects (2017A010101031,2017A030310163). National Natural Science Foundation of China (61703168). The Fundamental Research Funds for the Central Universities (2017MS048).Shenzhen basic research projects (JCYJ20160429161539298). Shenzhen peacock project (KQTD20140630154026047).

References

[1] A. Coates and A. Y. Ng, "Multi-camera object detection for robotics," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010, pp. 412–419.

[2] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. R. Bradski, "Self-supervised Monocular Road Detection in Desert Terrain.," in *Robotics: science and systems*, 2006, vol. 38.

[3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, 2016, pp. 21–37.

[4] 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*, 2015, pp. 91–99.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[6] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets,

atrous convolution, and fully connected crfs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, 2018.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv Prepr. arXiv1409.1556*, 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*, 2016, pp. 770–778.

[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 Prepr. arXiv1602.07360*, 2016.

[10] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv Prepr. arXiv1704.04861*, 2017.

[11] M. G. Hluchyj and M. J. Karol, "Shuffle Net: An application of generalized perfect shuffles to multihop lightwave networks," *J. Light. Technol.*, vol. 9, no. 10, pp. 1386–1397, 1991.

[12] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation," *CoRR*, vol. abs/1801.04381, 2018.

[13] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," *ACM Sigplan Not.*, vol. 49, no. 4, pp. 269–284, 2014.

[14] C. Farabet, C. Poulet, J. Y. Han, and Y. LeCun, "Cnp: An fpga-based processor for convolutional networks," in *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, 2009, pp. 32–37.

[15] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2015, pp. 161–170.

[16] C. Farabet, B. Martini, B. Corda, P. Akselrod, E. Culurciello, and Y. LeCun, "Neuflow: A runtime reconfigurable dataflow processor for vision," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, 2011, pp. 109–116.

[17] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun, and others, "Dadiannao: A machine-learning supercomputer," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, 2014, pp. 609–622.

[18] 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 Prepr. arXiv1408.5093*, 2014.

[19] M. Everingham, L. Van~Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, 2010.