Hardware Oriented Vision System of Logistics Robotics

Feng Liang[1], Chun Zhang[2]
[1]Institute of Microelectronics, Tsinghua University, Beijing 100084, China
[2]Research Institute of Tsinghua University in Shenzhen, Shen Zhen 518055, China
liangf16@mails.tsinghua.edu.cn, zhangchun@tsinghua.edu.cn

***Abstract*—A qualified logistics robot is required to locate and identify the target item properly. The vision system of the robot is the way it perceives the world which demands high precision and low latency. Using the state-of-art deep convolutional neural network model, we present a folder detector to locate and identify the file folder. We implement several classic CNN models in the Faster RCNN framework. The average precision of the ideal MobileNet model is up to 0.966 with the GPU inference time 59ms. Besides the model design, we present a hardware oriented layer adaptive quantization method. Using this method, we condense the model into low bitwidth fixed point arithmetic which is more efficient and hardware friendly than the GPU widely used 32 floating point arithmetic. We can condense the model into 8-bit fixed point arithmetic without the precision drop.**

***Keywords*—Logistics Robotics; Deep Convolutional Neural Network; Object Detection; Model Compression**

I. Introduction

With the boom of the electronic commerce, the logistics and transportation cost is becoming more critical. With the fact that most warehouses are manually operated, the automation support in the warehouse is highly demanded. The basic module of a logistics robot is the sense system or the vision system. The robot estimates the environment through sensing, and responds to these surroundings through actions.

The vision system of a qualified logistics robot demands high precision and low latency. In order to reduce costs, we propose a camera-based vision system without the expensive 3D laser radar. In the past, researchers used handmade SIFT[1] HOG[2] features to interpret the digital images. In 2012, Krizhevsky came up with a deep convolution neural network(CNN) called AlexNet[3] which won the ImageNet[5] LSVRC-2010 contest by a very large margin compared to the second place. Since then, oceans of researches have done to advance the state-of-art CNN. CNN can be adpted to many specific tasks, such as object detection[4]. We implement a folder detector to locate and identify the file folder using the state-of-art CNN object detector. We achieve a very good precision with a tolerable latency.

AlexNet[3] won the ImageNet competition with 60M parameters. These heavy CNNs outperform the old handmade features and stand out in feature extraction, but they also have major shortcomings. The storage, memory bandwidth, and computational resources are all highly demanded[6]. Because of the high computational complexity, the CNN models are often implemented in GPU which is very suitable for parallel computing. In general, the GPU uses 32 floating point arithmetic which is not cost efficient. On the other hand, the fixed point arithmetic is widely used in the hardware devices. In order to transfer our folder detector to a robotic mobile device, we present a hardware oriented layer adaptive quantization method to condense the model. We condense the model into 8-bit fixed point arithmetic without the precision drop.

The rest of this report is arranged as follows. Section 2 summarizes the CNN, object detection, model compression related works. Section 3 introduces our folder dataset and evaluation method. We describe our folder detector and quantization method in Section 4,5 respectively. The conclusion is stated in Section 6.

II. Related Work

A. Deep Convolutional Neural Network

Yann LeCun proposed a CNN model called LeNet to recognize the handwritten document[7] in the early 1990s. Neural network got into slow development for decades. In 2012, Krizhevsky and his AlexNet[3] show the superior performance of the CNN if we have enough data and computation resources. In contrast to traditional handmade SIFT[1] and HOG[2] features, features learned by the CNN from large-scale data such as ImageNet[5] need little priori knowledge in the training phase. VGG[8], GoogLeNet[10], ResNet[9] illustrates the depth matters in the CNN architecture. Nowadays, a lot of mobile device oriented network architectures are proposed to balance the accuracy and performance, like SqueezeNet[11] MobileNet[12] MobileNetV2[13].

B. Object Detection

The CNNs are very powerful feature extractor and can be easily adopted into other recognition tasks like object detection. With or without candidate proposals, the CNN object detectors can be divided into two-stage detector and one-stage detector. The R-CNN framework[4] is a representative two-stage detector, a sparse set of candidate object proposals are generated in the first stage using the selective search and the classifiers in the second stage classify each candidate proposal into the accurate category. A sequence of advances have done to improve the accuracy and speed of the detector, like Fast RCNN[14], Faster RCNN[15], Feature pyramid networks(FPN)[16], Mask RCNN[17].Without sparse

978-1-5386-6063-8/18/$31.00 ©2018 IEEE

TABLE I: Dataset Overview

	Pictures	Objects
Train/Val	254	1868
Test	108	953

candidate proposals, One-stage detectors handle a regular, dense anchor boxes sampling from the original images. Recent work on one-stage detectors, such as YOLO[18−20], SSD[21], RetinaNet[22], shows that one-stage detectors can achieve a good balance between speed and accuracy.

C. Model Compression

The state-of-art CNN outperforms in many computer vision tasks. However, the CNN demands large storage space and computational resources. It's hard to implement a 170MB ResNet101 model in an embedded device. With the problem becoming critical, Han Song proposes deep compression[23−25], a method to compress deep neural network by three steps: pruning the useless connections, quantize the weights and Huffman encoding. Weight pruning in deep compression[23] is unstructured, which require sparse BLAS(Basic Linear Algebra Subprograms) libraries or even specialized hardware[25] to implement. Structured pruning, like structured sparsity learning (SSL) method[26], filters pruning[27], is more hardware friendly. Besides weight pruning, model quantization is another topic in model compression. The quantized model uses low bitwidth arithmetic. In particular, in BNN[28] and XNOR-Net[29], both weights and interlayer activations of convolutional layers are quantized into 0/1.

III. Dataset & Evaluation

A. Dataset

To simulate a logistics warehouse, we build a folder dataset. The dataset contains 362 pictures with 2821 target folders. The pictures are taken by iPhone 8 Plus and adjusted to 806*604 pixels. As indicated in TABLE I, the training/validation set contains 70% pictures with the training validation ratio 0.7:0.3.

We label the pictures using the open source LabelImg tool. Fig. 1 is an example of the dataset.

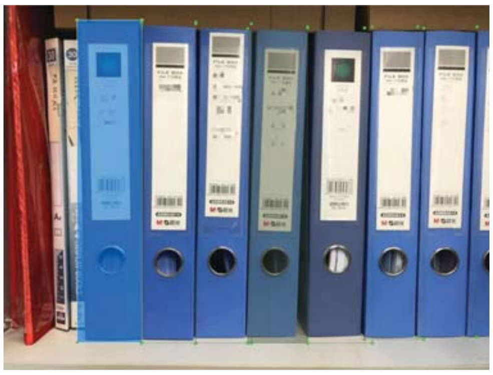

Fig. 1: Dataset Example

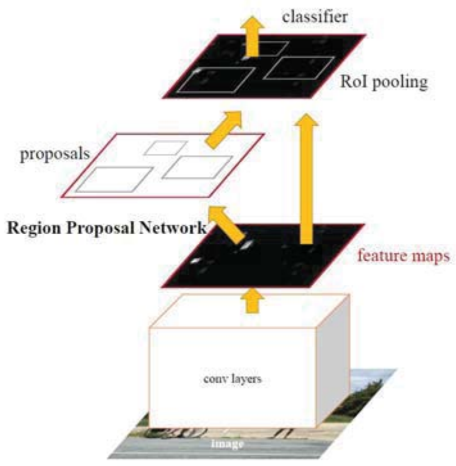

Fig. 2: Faster RCNN Architecture

B. Evaluation

In the object detection task, we use mAP(mean Average Precision) metric to measure the accuracy of object detectors. It is the average of the maximum precisions at different recall values.

IV. Improved Faster RCNN

We use Faster RCNN[15] as our object detector. Since Ross Girshick applies CNN in object detection, a sequence of advances have done to improve the accuracy and speed of the detector. Faster RCNN[15] has a great improvement in comparison with Fast RCNN[14]. It's a towards real-time object detector.

As shown in Fig. 2, Faster RCNN consists of two stages: Region Proposal Networks(RPN)[15] and Fast RCNN. These two modules share the same backbone CNN feature extractor which saves much computational complexity and speeds up the whole model. In the RPN stage, a set of candidate proposals are generated by the CNN based RPN. In the Fast RCNN stage, each candidate proposal is classified into the accurate category, the bounding boxes are refined.

We implement several backbone CNN models in Faster RCNN to evaluate the speed and the accuracy. The CNN models are pretrained using ImageNet[5]. We use PyTorch[33] framework. Evaluation results are in TABLE II.

According to the evaluation results, deep powerful networks like VGG[8] and ResNet[9] may be overqualified in our task.

TABLE II: Evaluation Results

Network	Inference(ms)	mAP(thr=0.8)	ckpt size(MB)
Res152	166	0.990	480
Res101	145	0.976	360
VGG16	104	0.983	1100
Mobile_1.0_224	73	0.978	43
Mobile_0.75_224	70	0.975	28
Mobile_0.5_224	67	0.971	16
Mobile_0.25_224	59	0.966	6.4

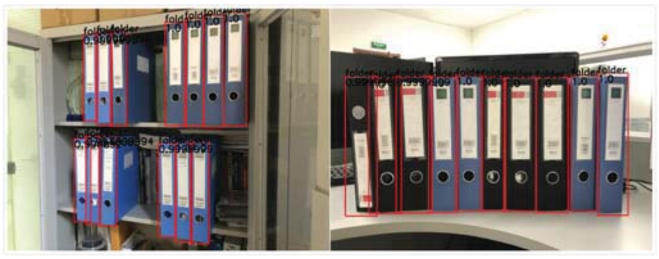

Fig. 3: Result of the Folder Detection

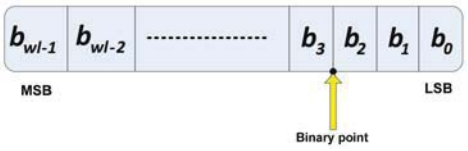

Fig. 4: Fixed Point Arithmetic

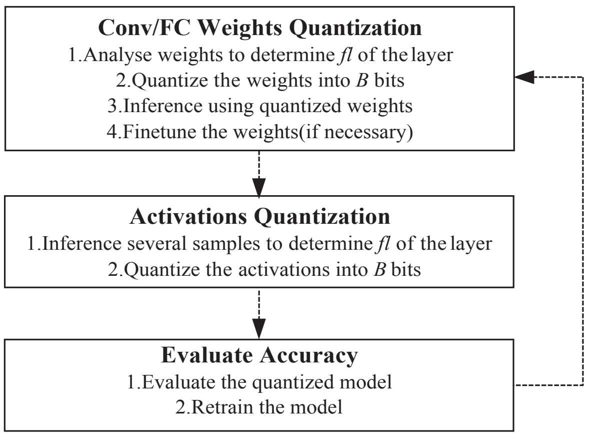
Conv/FC Weights Quantization
1.Analyse weights to determine *fl* of the layer
2.Quantize the weights into *B* bits
3.Inference using quantized weights
4.Finetune the weights(if necessary)
Activations Quantization
1.Inference several samples to determine *fl* of the layer
2.Quantize the activations into *B* bits
Evaluate Accuracy
1.Evaluate the quantized model
2.Retrain the model

Fig. 5: Quantization Flow

MobileNet[12] is designed to be deployed in the mobile device, it uses depthwise separable convolutions to build a light weight CNN. In our task, MobileNet can achieve high precision and is more embedded device friendly. In mobile_1.0_224, 1.0 224 are the width multiplier and the image resolution respectively.

We also do some experiment in the anchor mechanism by changing the anchor ratios and scales. The results seem to differ little. Fig. 3 is the result of folder detection.

V. Layer Adaptive Quantization

Our vision system is designed to be implemented in an embedded device. While GPU uses 32 floating point arithmetic which is not cost efficient, we condense the model into low bitwidth fixed point arithmetic which is more efficient and hardware friendly. Comparing with floating point, fixed point arithmetic can use bitwise operation to implement basic add and multiplication operations. The bit convolutions are much faster. Moreover, bit convolutions can be efficiently implemented on hardware,like CPU, FPGA, ASIC and GPU[31] .

A. *Layer Adaptive*

The weights in different layers of CNN have a significantly different range. We concentrate on the convolutional/linear layers and the following activation layers. The activation results are accumulated by the multiplication of the former activation and the current conv/fc layer. Thus the parameters in the activation layer are much larger than those in the conv/fc layer. While the 32 float point has an exceeded demand range, fixed point only has a limited dynamic range. Fortunately, it's more than sufficient. We adopt the dynamic fixed point[32] in our quantization method.

The quantization function is shown in (1).

$$x_{quan} = (-1)^s \cdot 2^{(-fl)} \sum_{i=0}^{B-2} 2^i \cdot x_i \tag{1}$$

Here s is the sign bit to meet the positive and negative range, *fl* is the fractional length that represents the precision, B denotes the quantized bit-width, and x the mantissa bits. As we have indicated before, different layer has different ranges. It means different layer has different constant *fl*. We should retain different constant *fl* for different later. On the hardware side, it is possible to implement dynamic fixed point arithmetic using bit shifters[30].

B. *Quantization Flow*

Once we have trained our full precision model which is indicated in part IV, we use layer adaptive quantization method to condense the full precision model into low bitwidth representation. The quantization flow is shown in Fig. 5.

Layer adaptive quantization flow has three stages to do the compression. In the first stage, we analyze the specific layer weights to find the appropriate *fl*. The fixed point representation should avoid saturation of the full precision parameter, so we use enough integer length to ensure the demand. It means the *fl* depends on the maximum absolute value(MBA) of the weights. In the stage of activations quantization, thing is a bit more complicated. Because activations of each conv/fc layer differ from sample to sample, we do some activations statistics for several samples before quantization. We try three statistical methods: MBA as in weights quantization, the percentile of MBA, exponential moving average (EMA). During our experiments, we find the 99.9 percentile of MBA has the best performance, followed by EMA, MBA.

C. *Retrain*

The accuracy will drop because of the quantization, so we also implement a finetune method to retrain the network. In the inference procedure, we get the quantized weights w_{quan} by sampling from full precision weights w by round nearest sampling. Since the quantized network can only have discrete weight/activation values, we should pay more attention to the back propagation procedure. We adopt the idea of previous work of Courbariaux[32]. We haven't implemented quantized gradient[31] in our algorithm. Although We use the quantized weights w_{quan} sampled from the full precision weights w in

TABLE III: QUANTIZATION RESULTS

Bitwidth	16-bits	8-bits	4-bits	2-bits
Without Retrain	0.962	0.971	0.003	0.000
Retrain	0.972	0.973	0.342	-

the inference phase, we update the full precision weights w using the full precision backward Δw.

D. Quantization Results

We implement our layer adaptive quantization method in the PyTorch[33] framework to simulate fixed point arithmetic. We redefine the conv/fc layers in the torch.nn.Module and add a new activation quantization layer. The quantization experiments are done in the Mobile_0.25_224 using Faster RCNN. Results(TABLE III) show that we can condense the model into 16-bit 8-bit fixed point without precision loss. After retraining, the 4-bit network is unsatisfactory. Maybe we should come up with some quantized training method.

VI. CONCLUSION

In this work, we present a hardware oriented vision system of logistics robotics. By using the state-of-art CNN, we present an improved Faster RCNN folder detector to locate and identify the file folder. We achieve a very good precision with a tolerable latency. The average precision of the ideal MobileNet model is up to 0.966 with the GPU inference time 59ms. Besides the model design, we present a hardware oriented layer adaptive quantization method. Using this method, we condense the model into 8-bit fixed point without the precision loss.

In the future, we may try other one-stage detectors, like YOLO[20], SSD[21], to help the logistic robot to identify the target. 4-bits or lower bitwidth quantization is another topic of our future work.

ACKNOWLEDGMENT

The authors would like to thank the Science, Technology and Innovation Commission of Shenzhen Municipality (No.GJHZ20170314153223728), and the National Natural Science Foundation of China under Grant No.61331003.

REFERENCES

[1] Lowe D G. Distinctive image features from scale-invariant keypoints[J]. International journal of computer vision, 2004, 60(2): 91-110.
[2] Dalal N, Triggs B. Histograms of oriented gradients for human detection[C]//Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. IEEE, 2005, 1: 886-893.
[3] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105.
[4] Girshick R, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2014: 580-587.
[5] Russakovsky O, Deng J, Su H, et al. Imagenet large scale visual recognition challenge[J]. International Journal of Computer Vision, 2015, 115(3): 211-252.
[6] Han S, Pool J, Tran J, et al. Learning both weights and connections for efficient neural network[C]//Advances in neural information processing systems. 2015: 1135-1143.
[7] LeCun, Yann, et al. ”Backpropagation applied to handwritten zip code recognition.” Neural computation 1.4 (1989): 541-551.
[8] Simonyan, Karen, and Andrew Zisserman. ”Very deep convolutional networks for large-scale image recognition.” arXiv preprint arXiv:1409.1556 (2014).
[9] He, Kaiming, et al. ”Deep residual learning for image recognition.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
[10] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015: 1-9.
[11] Iandola F N, Han S, Moskewicz M W, et al. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size[J]. arXiv preprint arXiv:1602.07360, 2016.
[12] Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.
[13] Sandler M, Howard A, Zhu M, et al. MobileNetV2: Inverted Residuals and Linear Bottlenecks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 4510-4520.
[14] Girshick R. Fast r-cnn[C]//Proceedings of the IEEE international conference on computer vision. 2015: 1440-1448.
[15] Ren S, He K, Girshick R, et al. Faster r-cnn: Towards real-time object detection with region proposal networks[C]//Advances in neural information processing systems. 2015: 91-99.
[16] Lin T Y, Dollr P, Girshick R B, et al. Feature Pyramid Networks for Object Detection[C]//CVPR. 2017, 1(2): 4.
[17] He K, Gkioxari G, Dollr P, et al. Mask r-cnn[C]//Computer Vision (ICCV), 2017 IEEE International Conference on. IEEE, 2017: 2980-2988.
[18] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 779-788.
[19] Redmon J, Farhadi A. YOLO9000: better, faster, stronger[J]. arXiv preprint, 2017.
[20] Redmon J, Farhadi A. Yolov3: An incremental improvement[J]. arXiv preprint arXiv:1804.02767, 2018.
[21] Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector[C]//European conference on computer vision. Springer, Cham, 2016: 21-37.
[22] Lin T Y, Goyal P, Girshick R, et al. Focal loss for dense object detection[J]. arXiv preprint arXiv:1708.02002, 2017.
[23] Han S, Pool J, Tran J, et al. Learning both weights and connections for efficient neural network[C]//Advances in neural information processing systems. 2015: 1135-1143.
[24] Han S, Mao H, Dally W J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding[J]. arXiv preprint arXiv:1510.00149, 2015.
[25] Han S, Liu X, Mao H, et al. EIE: efficient inference engine on compressed deep neural network[C]//Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on. IEEE, 2016: 243-254.
[26] Wen W, Wu C, Wang Y, et al. Learning structured sparsity in deep neural networks[C]//Advances in Neural Information Processing Systems. 2016: 2074-2082.
[27] Li H, Kadav A, Durdanovic I, et al. Pruning filters for efficient convnets[J]. arXiv preprint arXiv:1608.08710, 2016.
[28] Courbariaux M, Bengio Y. BinaryNet: Training deep neural networks with weights and activations constrained to+ 1 or? 1. arXiv: 1602.02830, 2016[J]. 2017.
[29] Rastegari M, Ordonez V, Redmon J, et al. Xnor-net: Imagenet classification using binary convolutional neural networks[C]//European Conference on Computer Vision. Springer, Cham, 2016: 525-542.
[30] Gysel P, Motamedi M, Ghiasi S. Hardware-oriented approximation of convolutional neural networks[J]. arXiv preprint arXiv:1604.03168, 2016.
[31] Zhou S, Wu Y, Ni Z, et al. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients[J]. arXiv preprint arXiv:1606.06160, 2016.
[32] Courbariaux M, Bengio Y, David J P. Training deep neural networks with low precision multiplications[J]. arXiv preprint arXiv:1412.7024, 2014.
[33] Paszke A, Gross S, Chintala S, et al. Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration[J]. 2017.