Training Bit Fully Convolutional Network for Fast Semantic Segmentation

He Wen and **Shuchang Zhou** and **Zhe Liang** and **Yuxiang Zhang** and **Dieqiao Feng** and **Xinyu Zhou** and **Cong Yao**
Megvii Inc.
{wenhe, zsc, liangzhe, zyx, fdq, zxy, yaocong}@megvii.com

Abstract

Fully convolutional neural networks give accurate, per-pixel prediction for input images and have applications like semantic segmentation. However, a typical FCN usually requires lots of floating point computation and large run-time memory, which effectively limits its usability. We propose a method to train Bit Fully Convolution Network (BFCN), a fully convolutional neural network that has low bit-width weights and activations. Because most of its computation-intensive convolutions are accomplished between low bit-width numbers, a BFCN can be accelerated by an efficient bit-convolution implementation. On CPU, the dot product operation between two bit vectors can be reduced to bitwise operations and popcounts, which can offer much higher throughput than 32-bit multiplications and additions.

To validate the effectiveness of BFCN, we conduct experiments on the PASCAL VOC 2012 semantic segmentation task and Cityscapes. Our BFCN with 1-bit weights and 2-bit activations, which runs 7.8x faster on CPU or requires less than 1% resources on FPGA, can achieve comparable performance as the 32-bit counterpart.

Introduction

Deep convolutional neural networks (DCNN), with its recent progress, has considerably changed the landscape of computer vision (Krizhevsky, Sutskever, and Hinton 2012) and many other fields.

To achieve close to state-of-the-art performance, a DCNN usually has a lot of parameters and high computational complexity, which may easily overwhelm resource capability of embedded devices. Substantial research efforts have been invested in speeding up DCNNs on both general-purpose (Vanhoucke, Senior, and Mao 2011; Gong et al. 2014; Han et al. 2015) and specialized computer hardware (Farabet et al. 2009; Farabet et al. 2011; Pham et al. 2012; Chen et al. 2014b; Chen et al. 2014c; Zhang et al. 2015a).

Recent progress in using low bit-width networks has considerably reduced parameter storage size and computation burden by using 1-bit weight and low bit-width activations. In particular, in BNN (Kim and Smaragdis 2016) and XNOR-net (Rastegari et al. 2016), during the forward pass the most computationally expensive convolutions can be done by combining xnor and popcount operations, thanks to the following equivalence when x and y are bit vectors:

$$\sum_i^n x_i y_i = n - 2\,\mathrm{popcount}(\mathrm{xnor}(x_i, y_i)),\ x_i, y_i \in \{-1, 1\}, \forall i.$$

Specifically, an FPGA implementation of neural network can take more benefit from low bit-width computation, because the complexity of a multiplier is proportional to the square of bit-widths.

However, most of previous researches on low bit-width networks have been focused on classification networks. In this paper, we are concerned with fully convolutional networks (FCN), which can be thought of as performing pixelwise classification of the input images and have applications in tasks like semantic segmentation (Long, Shelhamer, and Darrell 2015). Techniques developed in this paper can also be applied to other variants like RPN (Ren et al. 2015), FCLN (Johnson, Karpathy, and Fei-Fei 2015) and Densebox (Huang et al. 2015). Compared to a typical classification network, the following properties of FCN make it a better candidate to apply low bit-width quantizations.

1. An FCN typically has large feature maps, and some of them may need to be stored for later combination, which pushes up its peak memory usage. As BFCN uses low bit-width feature maps, the peak memory usage is significantly reduced.
2. An FCN usually accepts large input image and taps into a powerful classification network like VGGNet (Simonyan and Zisserman 2014) or ResNet (He et al. 2015) to boost performance. The acceleration offered by exploiting bit-convolution kernel, together with memory savings, would

network	VOC12	Cityscapes	speedup
32-bit FCN	69.8%	62.1%	1x
2-bit BFCN	67.0%	60.3%	4.1x
1-2 BFCN	62.8%	57.4%	7.8x

Table 1: Summary results of our BFCNs. Performance measure in mean IoU.

allow BFCN to be run on devices with limited computation resources.

Considering the method of training a low bit-wdith network is still under exploration, it remains a challenge to find a way to train a BFCN efficiently as well.

Our paper makes the following contributions:

1. We propose BFCN, an FCN that has low bit-width weights and activations, which is an extension to the combination of methods from Binarized Neural Network (Courbariaux, Bengio, and David 2014), XNOR-net (Rastegari et al. 2016) and DoReFa-net (Zhou et al. 2016).
2. We replace the convolutional filter in reconstruction with residual blocks to better suit the need of low bit-width network. We also propose a novel bit-width decay method to train BFCN with better performance. In our experiment, 2-bit BFCN with residual reconstruction and linear bit-width decay achieves a 67.0% mean intersection-over-union score, which is 7.4% better than the vanilla variant.
3. Based on an ImageNet pretrained ResNet-50 with bounded weights and activations, we train a semantic segmentation network with 2-bit weights and activations except for the first layer, and achieves a mean IoU score of 67.0% on PASCAL VOC 2012 (Everingham et al. 2015) and 60.3% on Cityscapes (Cordts et al. 2016), both on validation set as shown in 1. For comparison, the baseline full-precision model is 69.8% and 62.1% respectively. Our network can run at 5x speed on CPU compared to full-precision, and can be implemented on FPGA with only few percents resource consumption.

Related Work

Semantic segmentation helps computer to understand the structure of images, and usually serves as a basis of other computer vision applications. Recent state-of-the-art networks for semantic segmentation are mostly fully convolutional networks (Long, Shelhamer, and Darrell 2015) and adopt the architecture of encoder-decoder with multi-stage refinement (Badrinarayanan, Handa, and Cipolla 2015). In order to achieve best performance, powerful classification models are often embedded as part of the FCNs, which pushes up computational complexity together with large decoders.

To further refine the results from neural networks, CRFs are widely used in post-processing to improve local predictions (Chen et al. 2014a) by reconstructing boundaries more accurately. Since CRF can be integrated with most methods as post-processing step, which contributes little to our main topic, it will not be discussed in this paper.

Recent success of residual network has shown that very deep networks can be trained efficiently and performs better than any other previous network. There also exists successful attempt (Wu, Shen, and Hengel 2016) to combine FCN and ResNet, which achieves considerable improvement in semantic segmentation.

To utilize scene-parsing network in low-latency or real-time application, the computational complexity need to be significantly reduced. Some methods (Paszke et al. 2016; Kim et al. 2016) are proposed to reduce demand of computation resources of FCN by simplifying or redesigning the architecture of network.

We also note that our low bit-width method can be integrated with almost all other speed-up methods and achieves even further acceleration. For example, low-rank approaches (Zhang et al. 2015b) is orthogonal to our approach and may be integrated to BFCN.

Method

In this section we first introduce the design of our bit fully convolutional network, and then propose our method for training a BFCN.

Network design

A standard approach to perform semantic segmentation includes a feature extractor to produce feature maps from input image, and convolutions with upsampling operations to predict per-pixel labels from those feature maps. We use ResNet as feature extractor and adopt the multi-resolution reconstruction structure from Laplacian Reconstruction and Refinement (Ghiasi and Fowlkes 2016) to perform per-pixel classification over feature maps in different scales (see Figure 1).

However, while the network works well in full-precision, we observe a great loss in accuracy while converting it to low bit-width, indicating that this architecture is not suitable for low bit-width network. To address this issue, we evaluate different variations of BFCN, so as to figure out the cause of performance degeneration. As shown in Table 2, low bit-width network with single convolution in reconstruction structure suffer great performance degeneration. We also discover that adding more channels in reconstruction filter helps improve performance considerably, indicating a low bit-width convolution is not enough to extract spatial context from feature maps. In short, we need a more low bit-width friendly architecture in reconstruction to eliminate the bottleneck.

model	mean IoU	Ops in reconstruction
32-bit model	66.4%	3.7 GOps
baseline	59.6%	3.7 GOps
2x filter channel	63.3%	7.8 GOps
residual block	67.0%	6.9 GOps

Table 2: Comparison of different variations of low bit-width reconstruction structures on PASCAL VOC 2012. All variants except "residual block" use single convolution as filter in reconstruction.

Intuitively, we may add more channels to the filter in reconstruction. But it also pushes up computational complexity a lot. Fortunately, ResNet (He et al. 2015) allows us to go deeper instead of wider. It has been shown that a deeper residual block can often performs better than a wider convolution. Therefore, we address the issue by replacing the

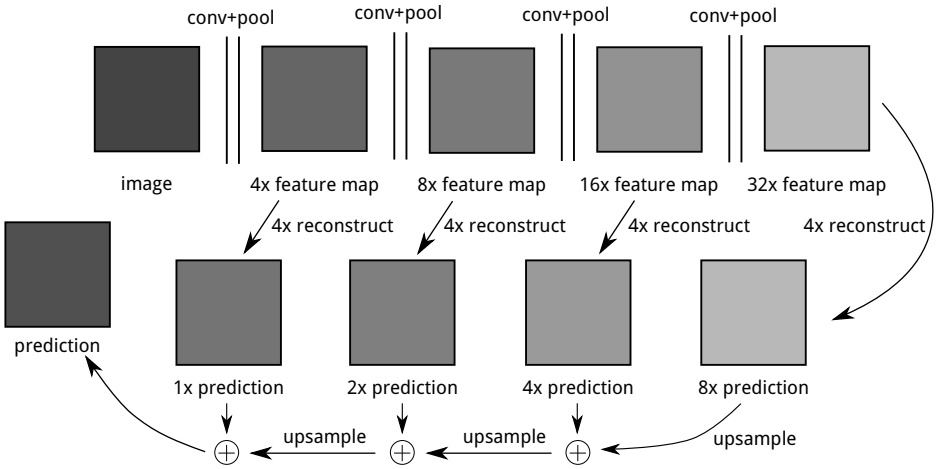


Figure 1: Network Architecture.

linear convolution with residual blocks. As shown in Table 2, our residual block variant even outperforms the original full-precision network.

In our approach, residual reconstruction structure can not only achieve better performance with similar complexity to a wide convolution, but also accelerate training by reduce the length of shortest path in reconstruction.

Bit-width allocation

It is important to decide how many bits to allocate for weights and feature maps, because bit-width has a crucial impact on both performance and computational complexity of a network. Since our goal is to speed-up semantic segmentation networks without losing much performance, we need to allocate bit-width carefully and wisely.

First we note it has been observed (Gupta et al. 2015) that 8-bit fixed point quantization is enough for a network to achieve almost the same performance as 32-bit floating point counterpart. Therefore, we focus our attention to bit-widths less than eight, which can provide us with further acceleration.

In order to extend bit-convolution kernel to m-bit weights and n-bit feature maps, we notice that:

$$\begin{aligned} A \times B =& (A_0 + 2A_1 + ... + 2^m A_m)(B_0 + ... + 2^n B_n) \\ =& A_0 B_0 + ... + 2^{i+j} A_i B_j + ... + 2^{m+n} A_m B_n \end{aligned}$$

where A_i, B_i represent the i-th bit of A and B. Therefore, it is pretty straightforward to compute the dot product using $m \cdot n$ bit-convolution kernels for m-bit weights and n-bit feature maps. It shows that the complexity of bit-width allocation, which is our primary goal to optimize, is proportional to the product of bit-widths allocated to weights and activations. Specifically, bit-width allocation becomes vital on FPGA implementation since it is the direct restriction of network size.

With fixed product of bit-widths, we still need to allocate bits between weights and activations. Intuitively, we would allocate bits equally as it keeps a balance between weights and activations, and error analysis confirms this intuition.

We first note that the error of a number introduced by k-bit quantization is $1/2^k$. As the errors are accumulated mainly by multiplication in convolution, it can be estimated as follow:

$$E = \frac{1}{2^{k_W}} + \frac{1}{2^{k_A}} = \frac{2^{k_W} + 2^{k_W}}{2^{k_W + k_A}} \tag{1}$$

When $c = k_W \cdot k_A$ is constant, we have the following inequality:

$$E \geq \frac{2 \times 2^{\sqrt{c}}}{2^{2\sqrt{c}}} \tag{2}$$

The equality holds iff $k_W = k_A = \sqrt{c}$, thus a balanced bit-width allocation is needed so as to minimize errors.

For the first layer, since the input image is 8-bit, we also fix bit-width of weights to 8. The bit-width of activations is still the same as other layers.

Route to low bit-width

initialization	mean IoU
low bit-width ResNet	63.5%
32-bit FCN	65.7%
8-bit BFCN	65.8%

Table 3: Results of different routes of training 2-bit BFCN on PASCAL VOC 2012 val set.

As shown in Figure 2, there are two ways to adapt the procedure of training a full-precision fully convolutional network to produce BFCN, denote as `P1` and `P2`.

The only difference between `P1` and `P2` is the initialization. `P1` uses full-precision FCN as initialization while

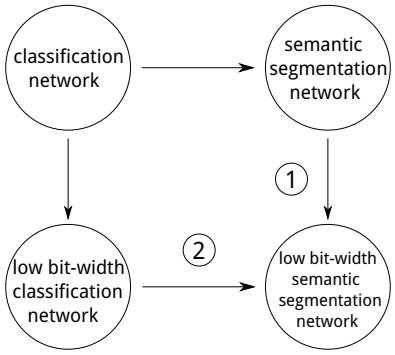


Figure 2: Routes of training BFCN.

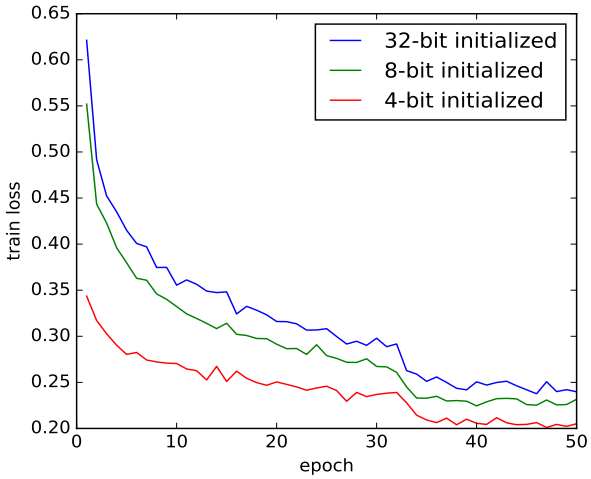


Figure 3: Training loss with different initializations of 2-bit BFCN. Experiments are conducted on PASCAL VOC 2012. The 4-bit model is initialized from 8-bit model.

`P2` uses low bit-width classification network. Here full-precision FCN serves as a intermediate stage in the procedure of training.

We evaluate the two routes and find the former one performs significantly better as the mean IoU scores indicate in Table 3. We then add one more intermediate stage to the procedure, the 8-bit BFCN, and achieve a slightly better result. We conjecture that utilizing intermediate network helps to preserve more information in the process of converting to low bit-width.

Bit-width decay

We notice that cutting off bit-width directly from full-precision to very low bit-width will lead to significant performance drop. To support this observation, we perform a simple experiment by training a 2-bit network initialized by a pretrained network of different number of bits. The training process (Figure 3) shows that networks initialized from lower bit-width converge faster.

This phenomenon can be explained by looking at the errors in quantization. Obviously, with higher original precision, a quantization step introduced larger error, and as as result the model benefit less from the initialization. However, introducing intermediate stages can help resolve it since networks with closer bit-widths tend to be more similar, hence more noise-tolerant when cutting off bit-width.

Our experiments show that BFCN can not recover from the quantization loss very well, if directly initialized from full-precision models. To extend the idea of utilizing intermediate models during training low bit-width network, we add more intermediate steps to train BFCN. We propose a method called bit-width decay, which cuts off bit-width step-by-step to avoid the overwhelming quantization error caused by large numeric precision drop.

We detail the procedure of bit-width decay method as follow:

1. Pretrain a full-precision network N_1.
2. Quantize N_1 to produce N_2 in 8-bit, which has been proved to be lossless, and fine-tune until its convergence.
3. Initialize N_3 with N_2.
4. Decrease bit-width of N_3, and fine-tune for enough iterations.
5. Repeat step 4 until desired bit-width is reached.

In this way, we can reduce the unrecoverable loss of quantization and the adverse impact of quantization can be mostly eliminated.

Experiments

In this section, we first describe the datasets we evaluate on and the experiment setup, then demonstrate the results of our method. Note that we conduct most of our experiments in our in-house machine learning system.

Datasets

We benchmarked the performance of our BFCN on PASCAL VOC 2012 and Cityscapes, two popular datasets for semantic segmentation.

The PASCAL VOC 2012 dataset on semantic segmentation consists of 1464 labelled images for training, and 1449 for validation. There are 20 categories to be predicted, including aeroplane, bus, chair, sofa, etc. All images in the dataset are not larger than 500x500. Following the convention of literature (Long, Shelhamer, and Darrell 2015; Wu, Shen, and Hengel 2016), we use the augmented dataset from (Hariharan et al. 2011), which gives us 10582 images for training in total. We also utilized reflection, resizing and random crop to augment the training data.

The Cityscapes dataset consists of 2975 street photos with fine annotation for training and 500 for validation. There are 19 classes of 7 categories in total. All images are in resolution of 2048x1536. In our experiment, the input of BFCN is random-cropped to 1536x768 due to GPU memory restriction, while validation is performed in its original size. We train our models with fine-annotated images only.

method	mean	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motor	person	plant	sheep	sofa	train	tv
32-bit FCN	**69.8**	**82.4**	38.9	**82.7**	64.3	66.4	**86.9**	**83.3**	**86.5**	**31.5**	**73.0**	48.9	**78.6**	**65.4**	**77.3**	81.0	**55.7**	79.3	41.3	77.8	65.1
4-bit BFCN	68.6	82.3	39.1	79.4	**67.4**	**66.5**	85.9	79.6	84.9	29.9	69.3	**50.1**	75.9	63.2	74.9	**81.1**	55.6	**82.3**	37.5	**78.0**	**66.2**
2-bit BFCN	67.0	80.8	**39.7**	75.4	59.0	63.2	85.2	79.5	83.6	29.7	71.3	44.2	75.6	63.8	73.1	79.7	48.5	79.6	**43.4**	74.4	65.4

Table 4: Class-wise results on PASCAL VOC 2012 val set.

For performance evaluation, we report the mean class-wise intersection-over-union score (mean IoU), which is the mean of IoU scores among classes.

Experiment Setup

All experiments are initialized from a ImageNet pretrained ResNet-50 with bounded activations and weights. We then use stochastic gradient descend with momentum of 0.9 to fine-tune the BFCN on semantic segmentation dataset.

Since the prediction on higher resolution feature maps in laplacian reconstruction and refinement structure depends on the prediction on lower resolutions, we use stage-wise losses to train the network. At first, we only define loss on 32x upsampling branch and fine-tune the network until convergence. Then losses of 16x, 8x and 4x upsampling branches are added one by one.

In order to overcome the huge imbalance of classes in Cityscapes dataset, we utilize a class weighing scheme introduced by ENet, which is defined as $W_{class} = 1/\ln(c + p_{class})$. We choose $c = 1.4$ to bound class weights in $[1, 3]$.

Results of different bit-width allocations

bit-width (W / A)	mean IoU	Complexity
32 / 32	69.8%	-
8 / 8	69.8%	64
4 / 4	68.6%	16
3 / 3	67.4%	9
2 / 2	65.7%	4
1 / 4	64.4%	4
4 / 1	diverge	4
1 / 2	62.8%	2

Table 5: Results of different bit-width allocated to weight and activation on PASCAL VOC 2012 val set. W represents weight and A for activation. Complexities are measured in terms of the number of bit-convolution kernels needed to compute low bit-width convolution.

First we evaluate the impact of different bit-width allocations on PASCAL VOC 2012 dataset (see Table 5).

We observe the performance of network degenerates while bit-width is decreasing, which correspond to our intuition. While 8-8 model performs exactly the same as the full-precision model, decreasing bit-width from 4-4 to 2-2 continuously incurs degeneration in performance. The performance degeneration is at first minor compared to bit-width saving, but suddenly becomes non-negligible around 4-4. We also discover that allocating different bit-widths to weights and activations harms performance compared to equally-allocated model with the same complexity.

From the results we conclude that 4-4 and 2-2 are favorable choices in different scenes. The 4-4 model can offer comparable performance with full-precision model but with considerable 75% resource savings compared to 8-8 on FPGA. In a more resource-limited situation, the 2-2 model can still offer good performance with only 6.25% hardware complexity of 8-8 model.

Results of bit-width decay

decay rate	mean IoU
1	67.0%
2	66.8%
3	66.1%
no decay	65.8%

Table 6: Results of different decay rates.

We then show how bit-width decay affects performance of networks on PASCAL VOC 2012.

It can be seen from Table 6 that bit-width decay does help to achieve a better performance compared to directly cutting off bit-width.

Besides, we evaluate the impact of "decay rate", which is the number of bits in a step. For a decay rate of r, we have $k_W = c - r \cdot t$ and $k_A = c - r \cdot t$ after t steps of decay, where $c = 8$ is the initial bit-width. The results of different decay rates are also presented in Table 6.

We discover with decay rate less than 2 we can achieve almost the same performance, but increasing it to 3 leads to a sudden drop in performance. It indicates network with 3 less bits starts diverging from its high bit-width couterpart.

Analysis of class-wise results

We demonstrate our class-wise results of PASCAL VOC 2012 and Cityscapes in Table 4 and 7.

As can be observed that most performance degeneration occur in classes which are more difficult to classify. In PASCAL VOC 2012, we observe that on fine-grained classes like

method	mean	road	sidewalk	building	wall	fence	pole	light	sign	veg	terrain	sky	person	rider	car	truck	bus	train	moto	bicycle
32-bit FCN	**62.1**	**95.8**	**73.5**	**88.2**	**31.4**	**38.2**	**52.6**	**49.6**	**65.8**	**89.8**	**52.7**	**90.1**	**72.8**	**47.6**	**89.9**	**40.8**	**57.3**	**37.2**	**38.2**	**69.1**
2-bit BFCN	60.3	95.3	71.2	87.6	25.9	36.2	51.8	49.0	63.3	89.1	51.7	89.9	71.4	44.5	**89.9**	40.1	53.8	32.4	33.8	67.5
1-2 BFCN	57.4	94.4	70.1	86.5	22.7	33.9	49.9	44.3	62.2	87.9	44.9	89.3	69.5	40.0	88.1	35.1	51.8	21.0	32.6	65.9

Table 7: Class-wise results on Cityscapes val set.

car and bus, cat and dog, BFCN is less powerful than its 32-bit counterpart, however on classes like sofa and bike, 2-bit BFCN even outperforms the full-precision network.

It can be seen more clearly on Cityscapes dataset: classes with low mean IoU scores in full-precision network become worse after quantization (like wall and train), while those large-scale, frequent classes such as sky and car remain in nearly the same accuracy.

The observation correspond to our intuition that a low bit-width quantized network is usually less powerful and thus harder to train on difficult tasks. It also suggest that we may use class balancing or bootstrapping to improve performance in these cases.

Analysis of run-time performance

We then analyze the run-time performance of BFCN on Tegra K1's CPU. We have implemented a custom runtime on arm, and all our results on CPU are measured directly in the runtime.

We note that 1 single precision operation is equivalent to 1024 bitOps on FPGA in terms of resource consumption, and roughly 18 bitOps on CPU according to the inference speed measured in our custom runtime. Thus, a network with m-bit weights and n-bit activations is expected to be $\frac{18}{m \cdot n}$ faster than its 32-bit counterpart ignoring the overheads.

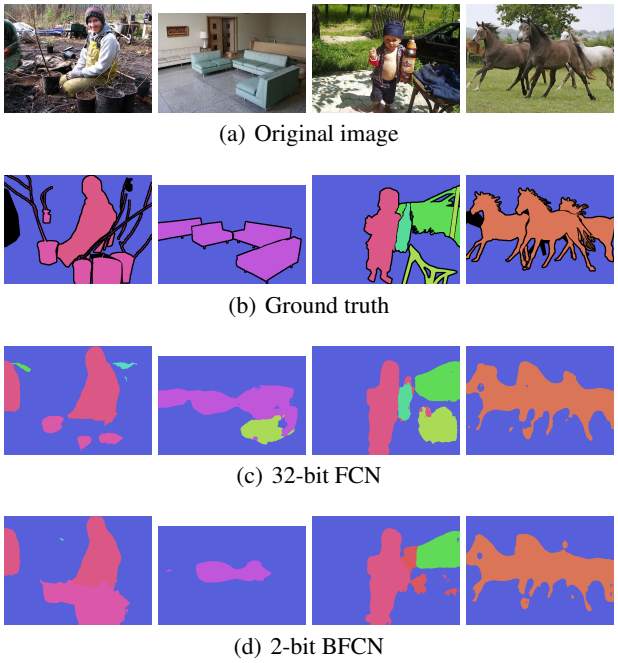
As shown in Table 8, our 1-2 BFCN can run 7.8x faster than full-precision network with only 1/32 storage size.

method	**run time**	**parameter size**
32-bit FCN	9681 ms	137.7 MB
1-2 BFCN	1237 ms	4.3 MB

Table 8: Comparison in hardware requirements of Cityscapes models. Run times are measured on Tegra K1 using single core for input size of 256x160x3.

Discussion

We present some example outputs on PASCAL VOC 2012 in Figure 4. From predictions we can see that BFCNs perform well on easy tasks. But on difficult tasks, which mostly consist of small objects or rare classes like bottle and sofa, BFCN will fail and have worse boundary performance. It also seems that BFCN has difficulties in reconstructing fine structures of the input image. However, low bit-width networks seldom misclassify the whole object, which effectively allow them to be used in real applications.



(a) Original image

(b) Ground truth

(c) 32-bit FCN

(d) 2-bit BFCN

Figure 4: Examples on PASCAL VOC 2012.

Conclusion and Future Work

In this paper, we propose and study methods for training bit fully convolutional network, which uses low bit-width weights and activations to accelerate inference speed and reduce memory footprint. We also propose a novel method to train a low bit-width network, which decreases bit-width step by step to reduce performance loss resulting from quantization. As a result, we are able to train efficient low bit-width scene-parsing networks without losing much performance. The low bit-width networks are especially friendly to hardware implementations like FPGA as low bit-width multipliers usually require orders of magnitude less resources.

As future work, a better baseline model can be used and CRF as well as other techniques can be integrated into BFCN for even better performance. We also note that our methods of designing and training low bit-width network

can also be applied to other related tasks such as object detection and instance segmentation.

References

[Badrinarayanan, Handa, and Cipolla 2015] Badrinarayanan, V.; Handa, A.; and Cipolla, R. 2015. Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. *arXiv preprint arXiv:1505.07293*.

[Chen et al. 2014a] Chen, L.-C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; and Yuille, A. L. 2014a. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*.

[Chen et al. 2014b] Chen, T.; Du, Z.; Sun, N.; Wang, J.; Wu, C.; Chen, Y.; and Temam, O. 2014b. Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. In *ACM Sigplan Notices*, volume 49, 269–284. ACM.

[Chen et al. 2014c] Chen, Y.; Luo, T.; Liu, S.; Zhang, S.; He, L.; Wang, J.; Li, L.; Chen, T.; Xu, Z.; Sun, N.; et al. 2014c. Dadiannao: A machine-learning supercomputer. In *Microarchitecture (MICRO), 2014 47th Annual IEEE/ACM International Symposium on*, 609–622. IEEE.

[Cordts et al. 2016] Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; and Schiele, B. 2016. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Courbariaux, Bengio, and David 2014] Courbariaux, M.; Bengio, Y.; and David, J.-P. 2014. Training deep neural networks with low precision multiplications. *arXiv preprint arXiv:1412.7024*.

[Everingham et al. 2015] Everingham, M.; Eslami, S. M. A.; Van Gool, L.; Williams, C. K. I.; Winn, J.; and Zisserman, A. 2015. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision* 111(1):98–136.

[Farabet et al. 2009] Farabet, C.; Poulet, C.; Han, J. Y.; and LeCun, Y. 2009. Cnp: An fpga-based processor for convolutional networks. In *2009 International Conference on Field Programmable Logic and Applications*, 32–37. IEEE.

[Farabet et al. 2011] Farabet, C.; LeCun, Y.; Kavukcuoglu, K.; Culurciello, E.; Martini, B.; Akselrod, P.; and Talay, S. 2011. Large-scale fpga-based convolutional networks. *Machine Learning on Very Large Data Sets* 1.

[Ghiasi and Fowlkes 2016] Ghiasi, G., and Fowlkes, C. C. 2016. Laplacian reconstruction and refinement for semantic segmentation. *CoRR* abs/1605.02264.

[Gong et al. 2014] Gong, Y.; Liu, L.; Yang, M.; and Bourdev, L. 2014. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*.

[Gupta et al. 2015] Gupta, S.; Agrawal, A.; Gopalakrishnan, K.; and Narayanan, P. 2015. Deep learning with limited numerical precision. *arXiv preprint arXiv:1502.02551*.

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

[Hariharan et al. 2011] Hariharan, B.; Arbeláez, P.; Bourdev, L.; Maji, S.; and Malik, J. 2011. Semantic contours from inverse detectors. In *2011 International Conference on Computer Vision*, 991–998. IEEE.

[He et al. 2015] He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

[Huang et al. 2015] Huang, L.; Yang, Y.; Deng, Y.; and Yu, Y. 2015. Densebox: Unifying landmark localization with end to end object detection. *arXiv preprint arXiv:1509.04874*.

[Johnson, Karpathy, and Fei-Fei 2015] Johnson, J.; Karpathy, A.; and Fei-Fei, L. 2015. Densecap: Fully convolutional localization networks for dense captioning. *arXiv preprint arXiv:1511.07571*.

[Kim and Smaragdis 2016] Kim, M., and Smaragdis, P. 2016. Bitwise neural networks. *arXiv preprint arXiv:1601.06071*.

[Kim et al. 2016] Kim, K.-H.; Cheon, Y.; Hong, S.; Roh, B.; and Park, M. 2016. Pvanet: Deep but lightweight neural networks for real-time object detection. *arXiv preprint arXiv:1608.08021*.

[Krizhevsky, Sutskever, and Hinton 2012] Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

[Long, Shelhamer, and Darrell 2015] Long, J.; Shelhamer, E.; and Darrell, T. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3431–3440.

[Paszke et al. 2016] Paszke, A.; Chaurasia, A.; Kim, S.; and Culurciello, E. 2016. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*.

[Pham et al. 2012] Pham, P.-H.; Jelaca, D.; Farabet, C.; Martini, B.; LeCun, Y.; and Culurciello, E. 2012. Neuflow: Dataflow vision processing system-on-a-chip. In *Circuits and Systems (MWSCAS), 2012 IEEE 55th International Midwest Symposium on*, 1044–1047. IEEE.

[Rastegari et al. 2016] Rastegari, M.; Ordonez, V.; Redmon, J.; and Farhadi, A. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. *arXiv preprint arXiv:1603.05279*.

[Ren et al. 2015] Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, 91–99.

[Simonyan and Zisserman 2014] Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *CoRR* abs/1409.1556.

[Vanhoucke, Senior, and Mao 2011] Vanhoucke, V.; Senior, A.; and Mao, M. Z. 2011. Improving the speed of neural

networks on cpus. In *Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop*, volume 1.

[Wu, Shen, and Hengel 2016] Wu, Z.; Shen, C.; and Hengel, A. v. d. 2016. High-performance semantic segmentation using very deep fully convolutional networks. *arXiv preprint arXiv:1604.04339*.

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

[Zhang et al. 2015b] Zhang, X.; Zou, J.; He, K.; and Sun, J. 2015b. Accelerating very deep convolutional networks for classification and detection.

[Zhou et al. 2016] Zhou, S.; Wu, Y.; Ni, Z.; Zhou, X.; Wen, H.; and Zou, Y. 2016. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*.