XNOR Neural Networks on FPGA

Fang Lin
flin4@stanford.edu

Abstract

We propose to implement the XNOR Neural Networks (XNOR-Net) on FPGA where both the weight filters and the inputs of convolutional layers are binary. XNOR-Net is regarded simple, accurate, efficient, and work on challenging visual tasks with portable devices and embedded systems. We also evaluate the high order quantization method which is expected to solve the loss of accuracy issue on XNOR-net.

1. Introduction

Convolutional neural networks (CNN) has reliable outcomes on object recognition, detection, and classification, etc. However, CNN-based recognition frameworks require very high computational power and large amounts of memory. Deep neural networks suffer from over-parameterization and very high redundancy in their models, which results in inefficient computation and memory usage. While GPU-based machines have performed well on expensive, they are often not suitable for portable devices and embedded systems. Currently, power consumption has drawn massive attentions from mobile devices. Tasks like real-time text detection, object detection, environment emergency alerting on devices, like glasses, would drain its battery quickly. We evaluated different methods to solve this issue and chose to implement XNOR-Net on FPGA since it's proved very efficient and resource saving. The input of our model is small datasets CIFAR-10 and MNIST for classifications.

2. Related Work

There are several approaches proposed to achieve efficient training and inference processing in deep neural works.

2.1. Using shallow networks

The most straightforward solution is using shallower model can reduce the size of network. [1] proves with CIFAR-10 that shallow nets are able to achieve the same function as deep nets. However, with a large dataset like SIFT feature classification on ImageNet, the shallow nets can't perform so well [2-3].

2.2. Compressing pre-trained deep networks

Cut-off or pruning extra, redundant, or non-informative weights in a trained network is able to reduce the size of the network while processing inference. Through pruning on some state-of-the-art networks, [4] is able decrease the number of parameters by an order of magnitude without affecting their accuracy by learning only the important connections. [5] reduces the number of activations for compression and acceleration. Also, deep compression [6] achieves less storage and energy to run inference on large networks which is able to be deployed on portable devices. As we can notice, this approach needs the network to be pre-trained and then pruning.

2.3. Quantizing parameters

According to researches, to achieve high performance in deep neural networks, it's not required to use high precision parameters. [7] quantized the weights of fully connected layers (FCC) using vector quantization techniques. Through only thresholding the weight values at zero would only drop the top-1 accuracy by less than 10 percent. [8] provided a new algorithm for training a sparse networks with only three (+1/0/-1) weights. [9] proposed a fixed-point implementation of 8-bit integer was compared with 32-bit floating point activations. Similarly, [10] also proposed another fixed-point network with ternary weights and 3-bits activations. [11] quantize a network with L2 error minimization getting better accuracy on MNIST and CIFAR-10 datasets. [12] proposed a back-propagation flow via quantizing the representations at each layer of the network.

2.4. Network binarization

There are several approaches attempt to binarize the weights and the activation functions in the network. [13] proposed the expectation backpropagation (EBP), which is proved to have high performance achieving through a network with binary weights and binary activations. While, in EBP the binarized parameters were only used during inference. [14] presented a fully binary network running real-time using a similar approach as EBP, which has improved a lot in efficiency. Introducing the probabilistic idea within the EBP, [15] proposed BinaryConnect, which uses the real-valued version of the weights as a key reference for the binarization process. While it can perform well on small datasets (e.g. CIFAR-10, SVHN), it can't behave well on large-scale datasets (e.g., ImageNet). [16] propose an extension of BinaryConnect, BinaryNet, where both weights and activations are binarized. Similarly, [17] proposed XNOR-Net, where both the filters and the input to convolutional layers are binary but has different binarization method and network architecture. It approximate convolutions using primarily binary operations which achieve 58x faster convolutional operations and 32x memory savings. Interestingly, [18] pointed out that the noise introduced by weight binarization provides a form of regularization, which can improve the accuracy. [20] combines the previously trained neural network with binary weights and binary inputs. It also replaces float multiplication with bit XNOR and float addition with bit counting. Specifically, we choose XNOR-Net which only need to do XNOR between inputs and weighs in one layer and the outpu to next layer is activated if the counts of 1s is greater than a threshold.

2.5. Our approach

Specifically, we choose XNOR-Net which only need to do XNOR between inputs and weighs in one layer and the outpu to next layer is activated if the counts of 1s is greater than a threshold. By this change, the workload pattern becomes very suitable and highly parallelizable for FPGA, at the expense of only a small decrease of accuracy.

3. Methods

Since we choose to implement XNOR-Net, the main reference is [20].

For FPGA implementation, we refer to [22-28] with our own optimizations.

The complexity of deep convolutional networks could be split into two major parts. First part, the convolutional layers which contain around 90% of the arithmetic operations. So, our target for the energy-efficient accelerator is 1) offer a large enough computational throughput and 2) offer a memory-bandwidth which is able to keep the processing elements running. First of all, we seek to decrease the number of interconnections of the fully-connected neural networks. The totally number of weights grows exponentially with the a number of nodes. If using traditional method, say our fully connected network layers have 1024 inner nodes for each hidden layer, which sums to more than a million interconnections in real hardware. This is definitely not acceptable and impractical. Thus, our first task is to reduce the number of interconnections. FCC layers and convolutional layers consist of additions and multiplications, while the latter needs a much larger chip area. Motivated by the former experiences, through using integer power of two weights is able to turn multiplications into bit shifts, which significantly decrease amount of energy. We choose the similar approach as [29], to use an indirect connection between two nodes instead of using direct connection. First rearrange the contribution of each node due to commutative property and then express computation procedure between different layers with a matrix multiplication operation. In this case, we can use shift and elementwise multiplication only instead of using matrix multiplication through rearranging the weights by diagonal. Though we choose limit the weights in positive, it's quite easy to adapt the approximation process to positive and negative both. As we all know, the computation of convolutions plus FCC mainly consist of multiply-and-accumulate process.

The methods are able to decrease the number of interconnections from more than a million to around 2K for each hidden layer, which can achieve 500 times resource saving similar as Song mentioned on the guest introduction of deep compression. Combining the optimization mentioned above with the XNOR-net mechanism provided by [20], our implementation on hardware is presented in Fig. 1, which inputs binary and outputs binary numbers.

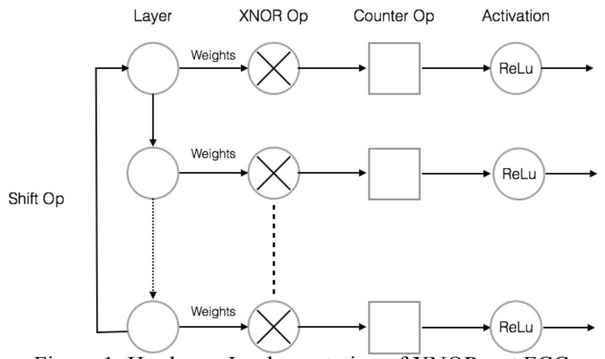

Figure 1: Hardware Implementation of XNOR-net FCC

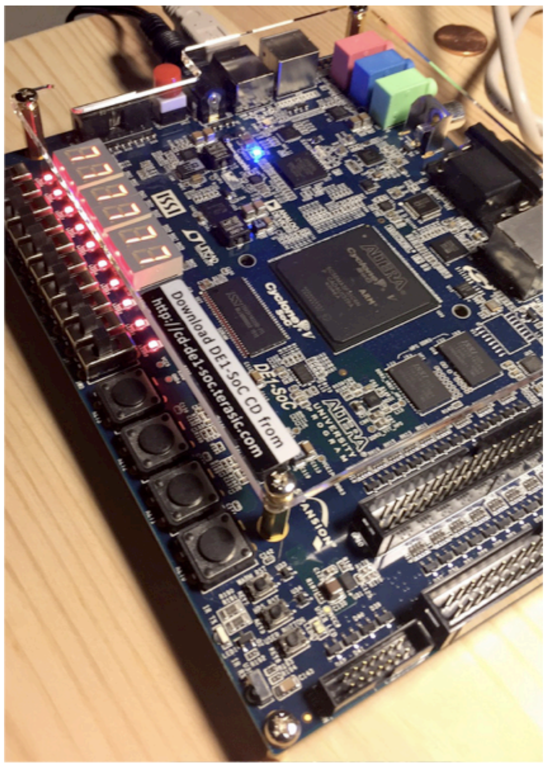
Figure 2: Hardware Setup

4. Dataset and Features

We choose to use same architecture as in [11] Theano experiments. Applying shift-based AdaMax and BN (with a mini batch of size 200) instead of the vanilla implementations to reduce the number of multiplications. Likewise, we decay the learning rate by using a 1-bit right shift every 50 epochs. BC and BNN showed near state-of-the-art performance on CIFAR-10, MNIST. BWN and XNOR-Net on CIFAR-10 using the same network architecture as BC and BNN achieve the error rate of 9.88% and 10.17%, respectively [20]. The prototype code is available on my github: https://github.com/PhoenixShield/XNOR-net.git

5. Results and Comparisons

For scientific comparing, we implemented the FCC version focusing on digit classification using MNIST dataset. The board we are using is Altera DE1-SoC which works on 50MHz clock. The standard development tool-chain starts with a Python-Theano based implementation of the algorithm, which then undergoes tons of testing to make sure it's function correctly. Based on the verified python prototype, we finished the hardware implementation using Verilog HDL.

Although our scheme is fully parametrical, we fix the number of hidden layer nodes as 1024 the same as the state-of-the art counterparts. Also, after bottleneck analysis, we found that our scheme is bounded by the bandwidth. In order for minimize the cost of loading data, we choose the batch process trick. That is to duplicate the scheme as many times as it can use the chip pins as much as possible. On our board, we duplicate twice which achieves 3x faster. It turns out that our approach behave better (use less resources) than others who uses faster FPGAs.

We also compare our results with the results from [30] who has the state-of-the-art approach similar as ours. It turns our that our optimized scheme performs better than theirs. Comparing our results on MNIST dataset, if regard to cycles instead of absolute time (since we want to avoid the difference between boards), our optimized scheme behave 3x more throughput than [30]. Remember that due to the parallel mechanism, we can always achieve higher through once more resources are available.

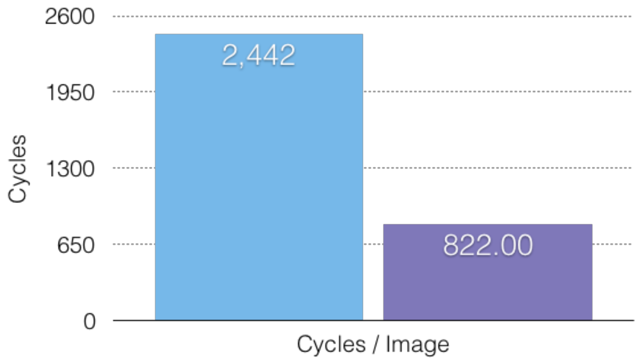

Figure 3: Comparison between our design and baseline (a)

Also, our design has a 25 times lower latency than [30].

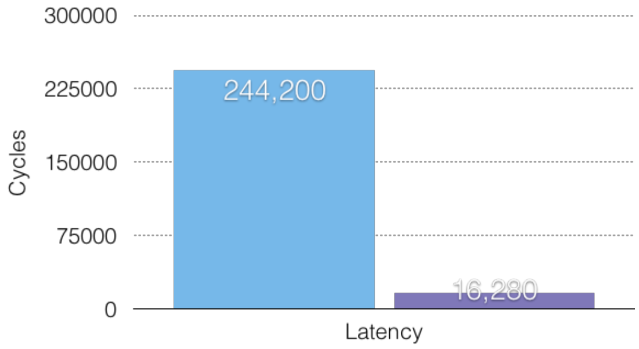

Figure 4: Comparison between our design and baseline (b)

Regards to power consumption, [30] has the real power measurement at 1mJ per image, while our optimized scheme shows 0.78mJ cost per image according to the Altera power simulator.

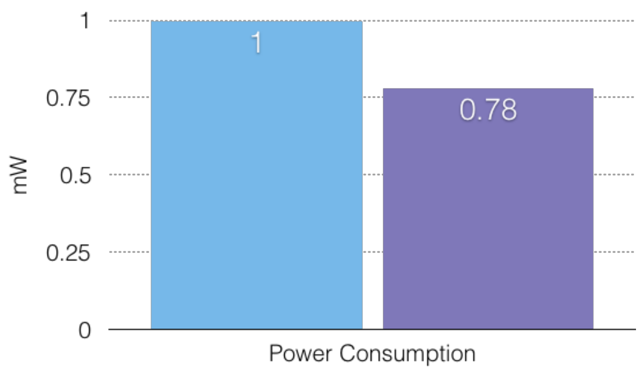

Figure 5: Comparison between our design and baseline (c)

According to the results shown above, our optimized implementation of neural networks saves the on-chip resources significantly through using XNOR-net and is able to achieve on-pair accuracy as non XNOR-net. Also, our optimized scheme cost less power than the state-of-the-art design.

6. Future Work

6.1. Network Pruning

Neural network pruning has been widely studied to compress CNN models [31] - tarting by learning the connectivity via normal network traning, and then prune the small-weight connections. As shown in [31], pruning is able to reduce the number of parameters by 9x and 13x for AlexNet and VGG-16 model. We believe combining with network pruning the XNOR-net, we can squeeze the network size much smaller into next level.

6.2. Trained Tenary Quantization

[21] proposed a trained tenary quantization (TTQ), which is able to reduce the precision of weights in neural networks to ternary values. It proves a very little accuracy degradation with a not much aggressive quantized weights than XNOR-Net. They claimed a 32x smaller model size improvement. We assume by applying a tenary quantization instead of binarization, we might be able to improve our model accuracy into next level.

References

[1] Ba, J., Caruana, R.: Do deep nets really need to be deep?. Advances in neural information processing systems. 2654–2662, 2014.

[2] Seide, F., Li, G., Yu, D. Conversational speech transcription using context-dependent deep neural networks. Interspeech,437–440, 2011.

[3] Dauphin, Y.N., Bengio, Y. Big neural networks waste capacity. arXiv preprint arXiv, 1301–3583, 2013.

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

[5] Van Nguyen, H., Zhou, K., Vemulapalli, R., Cross-domain synthesis of medical images using efficient location-sensitive deep network., Medical Image Computing and Computer-Assisted Intervention-MICCAI, 677-684, 2015.

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

[7] Gong, Y., Liu, L., Yang, M., Bourdev, L., Compressing deep convolutional networks using vector quantization., arXiv preprint arXiv:1412.6115, 2014.

[8] Arora, S., Bhaskara, A., Ge, R., Ma, T.: Provable bounds for learning some deep representations. arXiv preprint arXiv:1310.6343, 2013.

[9] Vanhoucke, V., Senior, A., Mao, M.Z.: Improving the speed of neural networks on cpus. In: Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop. Volume 1., 2011

[10] Hwang, K., Sung, W.: Fixed-point feedforward deep neural network design using weights+ 1, 0, and- 1. In: Signal Processing Systems (SiPS), 2014 IEEE Workshop on, IEEE (2014) 1–6

[11] Anwar, S., Hwang, K., Sung, W.: Fixed point optimization of deep convolutional neural networks for object recognition. In: Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on, IEEE (2015) 1131–1135

[12] Lin, Z., Courbariaux, M., Memisevic, R., Bengio, Y.: Neural networks with few multiplica- tions. arXiv preprint arXiv:1510.03009, 2015

[13] Soudry, D., Hubara, I., Meir, R.: Expectation backpropagation: parameter-free training of multilayer neural networks with continuous or discrete weights. In: Advances in Neural Information Processing Systems. 963–971, 2014.

[14] Esser, S.K., Appuswamy, R., Merolla, P., Arthur, J.V., Modha, D.S.: Backpropagation for energy-efficient neuromorphic computing. In: Advances in Neural Information Processing Systems., 1117–1125, 2015.

[15] Courbariaux, M., Bengio, Y., David, J.P.: Binaryconnect: Training deep neural networks with binary weights during propagations. In: Advances in Neural Information Processing Systems. 3105–311, 2015.

[16] Courbariaux, M., Bengio, Y.: Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. CoRR, 2016.

[17] Mohammad R., Vicente O., Joseph R., XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks, arXiv:1603.05279, 2016.

[18] Wan, L., Zeiler, M., Zhang, S., Cun, Y.L., Fergus, R.: Regularization of neural networks us- ing dropconnect. In: Proceedings of the 30th International Conference on Machine Learning (ICML-13).,1058–1066, 2013.

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

[20] Mohammad R., Vicente O., Joseph R., XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks, arXiv, 1603.05279, 2016.

[21] Chenzhuo Z., Song H., Huizi M., Trained Ternary Quantization,. arXiv: 1612.01064, 2017.

[22] Andri, R., Cavigelli, L., Rossi, D., & Benini, L. (2016). YodaNN: An Ultra-Low Power Convolutional Neural Network Accelerator Based on Binary Weights. arXiv preprint arXiv:1606.05487.

[23] Lukas Cavigelli, David Gschwend, Christoph Mayer, Samuel Willi, Beat Muheim, Luca Benini, "Origami: A Convolutional Network Accelerator", Proc. ACM/IEEE GLS-VLSI'15

[24] F. Conti, L. Benini, "A Ultra-Low-Energy Convolution Engine for Fast Brain-Inspired Vision in Multicore Clusters", Proc. ACM/IEEE DATE'15

[25] Yu-Hsin Chen, Tushar Krishna, Joel Emer, Vivienne Sze, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks", Proc. ISSCC'16.

[26] C. Farabet, B. Martini, B. Corda, P. Akselrod, E. Culurciello and Y. LeCun, "NeuFlow: A Runtime Reconfigurable Dataflow Processor for Vision", Proc. IEEE ECV'11@CVPR'11

[27] Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, Jason Cong, "Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks", Proc. FPGA'15

[28] L. Cavigelli, M. Magno, L. Benini, "Accelerating real-time embedded scene labeling with convolutional networks", Proc. ACM/IEEE/EDAC DAC'15

[29] Y, Yunfan, Huang, Yayun, http://arainhyy.github.io

[30] P. Jinhwan, S. Wongyong, Fpga Based Implementation of Deep Neural Network Using On-chip Memory Only, arXiv preprint arXiv: 1602.01616, 2016

[31] H., Song, M. Huizi, J. D Wiliam, Deep Compression: Compressing Deep Neural Networks With Pruning, Tained Quantization And Huffman Coding, arXiv: 1510.00149, 2016