Differentiable Training for Hardware Efficient LightNNs

Ruizhou Ding, Zeye Liu, Ting-Wu Chin, Diana Marculescu, R.D. (Shawn) Blanton
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
{rding,zeyel,tingwuc,dianam,rblanton}@andrew.cmu.edu

Abstract

To reduce runtime and resource utilization of Deep Neural Networks (DNNs) on customized hardware, LightNN has been proposed by constraining the weights of DNNs to be a sum of a limited number (denoted as $k \in \{1, 2\}$) of powers of 2. LightNNs can therefore replace the multiplication between activations and weights with a shift operation or two shifts and an add operation. To provide a more continuous Pareto-optimal curve of accuracy and runtime so that hardware designers can have more flexible options of DNN configurations, one can customize the k for each convolutional filter. In this paper, we formulate the selection of k to be differentiable, and train the model weights and per-filter k in an end-to-end fashion. Since flexible-k LightNNs (FLightNNs) fully utilize the hardware resources on Field Programmable Gate Arrays (FPGAs), our experimental results show that FLightNN can achieve $2\times$ speedup on FPGAs when compared to LightNN-2, with only 0.1% accuracy degradation. In addition, compared to a 4-bit fixed-point quantization, FLightNN can achieve slightly higher accuracy and $1.4\times$ speedup, due to its lightweight shift operations.

1 Introduction

Emerging vision, speech and natural language applications have widely adopted deep learning models and, as a result, have achieved state-of-the-art accuracy [6, 17, 4]. Furthermore, recent industrial effort has focused on implementing the models on mobile devices [1]. However, real-time applications based on these deep models may incur unacceptably large latencies and can easily drain the battery on energy-limited devices. For example, smartphones can only run the AlexNet-based object detection for one hour [16]. Therefore, prior research has proposed model compression techniques including pruning and quantization to satisfy the stringent energy and latency requirements [10, 12, 3, 14].

As one of the recently proposed quantization approaches, LightNN [7] constrains the weights of DNNs to be a sum of k powers of 2, and therefore can use shift and add operations to replace the multiplications between activations and weights. For LightNN-1[1], all the multiplications of the DNNs will be replaced by a shift operation, while for LightNN-2, two shifts and an add replace the multiplication. Since shift operations are much more lightweight on customized hardware (*e.g.*, FPGA or ASIC), LightNNs can achieve faster speed and lower energy consumption, and generally maintains accuracy for over-parameterized models [7]. Although LightNNs provide better energy-efficiency, they lack the flexibility to provide fine-grained energy/delay and accuracy curve. As explained in Fig. 1, the energy efficiency for these models also exhibits gaps, making the Pareto-optimal curve of accuracy and energy discrete. However, a continuous accuracy and energy/delay trade-off is an important feature for vendors to target different market segments (*e.g.*, IoT devices, edge devices, and mobile devices).

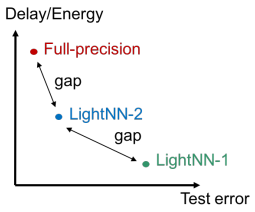


Figure 1: A discrete Pareto-optimal curve for LightNN models.

[1]LightNN-k indicates LightNN whose weights are quantized to be the sum of k powers of 2.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

To provide a more flexible Pareto curve for the LightNN framework, we propose to equip each convolutional filter with the freedom to use a different number of shift-and-add operations to approximate multiplications. Specifically, we introduce a set of free variables $\mathbf{k} = \{\mathbf{k}_1, \ldots, \mathbf{k}_F\}$ where each dimension represents the number of shift-and-add for the corresponding convolutional filter. As a result, a more contiguous Pareto curve can be achieved. For example, if we constrain $\mathbf{k} \in \{1, 2\}^F$, then the delay and energy consumption of the new model will sit between LightNN-1 ($\mathbf{k} = \{1\}^F$) and LightNN-2 ($\mathbf{k} = \{2\}^F$). Formally, we are solving $\min_{\mathbf{w},\mathbf{k}} \mathcal{L}(\mathbf{w}, \mathbf{k})$, where $\mathcal{L}$ is the loss function and $\mathbf{w}$ is the weights. However, the commonly adopted stochastic gradient descent (SGD) does not apply in this case since $\mathcal{L}$ is non-differentiable $w.r.t.$ $\mathbf{k}$. In this paper, we propose a differentiable training algorithm which enables end-to-end optimization with standard SGD. The resulting network is dubbed ***FLightNN*** for its flexible $\mathbf{k}$ values.

2 Differentiable Training for FLightNN

In this section, we first define the quantization function, and then introduce the end-to-end training algorithm for FLightNN, equipped with a regularization loss to penalize large $\mathbf{k}$ values.

2.1 Quantization function

We first denote the i^{th} filter of the network as $\mathbf{w}_i$ and the quantization function for the filter $\mathbf{w}_i$ as $\mathcal{Q}_k(\mathbf{w}_i|\mathbf{t})$, where $k = \max_i \mathbf{k}$ is the maximum number of shifts used for this network, and vector $\mathbf{t}$ is a latent variable that controls the approximation (*e.g.*, some threshold values). Also, we denote the residual resulting from the approximation as $\mathbf{r}_{i,k} = \mathbf{w}_i - \mathcal{Q}_k(\mathbf{w}_i|\mathbf{t})$. Then, we formally define the quantization function as follows:

$$\mathcal{Q}_k(\mathbf{w}_i|\mathbf{t}) = \begin{cases} 0, & \text{if } k = 0 \\ \sum_{j=0}^{k-1} \mathbb{1}(||\mathbf{r}_{i,j}||_2 > \mathbf{t}_j) R(\mathbf{r}_{i,j}), & \text{if } k \geq 1 \end{cases}$$

where $R(x) = sign(x) \times 2^{[log(|x|)]}$ rounds the input variable to a nearest power of 2, and $[.]$ is a rounding-to-integer function. To interpret the thresholds $\mathbf{t}$, $\mathbf{t}_0$ determines whether this filter is pruned out, and $\mathbf{t}_1$ determines whether one shift is enough, etc. Then, the number of shifts for the i-th filter is $\mathbf{k}_i = \sum_{j=0}^{k-1} \mathbb{1}(||\mathbf{r}_{i,j}||_2 > \mathbf{t}_j)$. Therefore, choosing $\mathbf{k}_i$ per filter is equivalent to finding optimal thresholds $\mathbf{t}$.

2.2 Differentiable training

Instead of picking the threshods t by hand, we consider them as trainable parameters. Therefore, the loss function $\mathcal{L}(\mathbf{w}, \mathbf{t})$ is a function of both weights and thresholds. Similar to prior work on DNN quantization [19, 5], we use the straight-through estimator (STE) [2] to compute $\frac{\partial \mathcal{L}}{\partial \mathbf{w}_i}$. By defining $\frac{\partial \mathbf{w}_i^q}{\partial \mathbf{w}_i} = 1$ where $\mathbf{w}_i^q = \mathcal{Q}_k(\mathbf{w}_i|\mathbf{t})$ is the quantized $\mathbf{w}_i$; therefore, we have $\frac{\partial \mathcal{L}}{\partial \mathbf{w}_i} = \frac{\partial \mathcal{L}}{\partial \mathbf{w}_i^q} \cdot \frac{\partial \mathbf{w}_i^q}{\partial \mathbf{w}_i} = \frac{\partial \mathcal{L}}{\partial \mathbf{w}_i^q}$, which becomes a differentiable expression.

To compute the gradient for thresholds, *i.e.*, $\frac{\partial \mathbf{w}_i^q}{\partial \mathbf{t}_j}$, we relax the indicator function $g(x, \mathbf{t}_j) = \mathbb{1}(x > \mathbf{t}_j)$ to a sigmoid function [9], $\sigma(.)$, when computing gradients, *i.e.*, $\hat{g}(x, \mathbf{t}_j) = \sigma(x - \mathbf{t}_j)$. In addition, we use STE to compute the gradient for $R(x)$. Thus, the gradient $\frac{\partial \mathbf{w}_i^q}{\partial \mathbf{t}_j}$ can be computed by:

$$\begin{aligned} \frac{\partial \mathcal{Q}_{\mathbf{k}_i}(\mathbf{w}_i|\mathbf{t})}{\partial \mathbf{t}_j} &= \sum_{l=0}^{\mathbf{k}_i - 1} \frac{\partial \sigma(||\mathbf{r}_{i,l}||_2 - \mathbf{t}_l)}{\partial \mathbf{t}_j} R(\mathbf{r}_{i,l}) + \sigma(||\mathbf{r}_{i,l}||_2 - \mathbf{t}_l) \frac{\partial R(\mathbf{r}_{i,l})}{\partial \mathbf{t}_j} \\ &= \sum_{l=0}^{\mathbf{k}_i - 1} \sigma'(||\mathbf{r}_{i,l}||_2 - \mathbf{t}_l)(\frac{\partial ||\mathbf{r}_{i,l}||_2}{\partial \mathbf{t}_j} - \frac{\partial \mathbf{t}_l}{\partial \mathbf{t}_j}) R(\mathbf{r}_{i,l}) + \sigma(||\mathbf{r}_{i,l}||_2 - \mathbf{t}_l) \frac{\partial \mathbf{r}_{i,l}}{\partial \mathbf{t}_j} \end{aligned} \tag{1}$$

where $\frac{\partial ||\mathbf{r}_{i,l}||_2}{\partial \mathbf{t}_j}$ and $\frac{\partial \mathbf{r}_{i,l}}{\partial \mathbf{t}_j}$ are 0 for $l < j$; otherwise, they can be computed with the result of $\frac{\partial \mathcal{Q}_l(\mathbf{w}_i|\mathbf{t})}{\partial \mathbf{t}_j}$. $\frac{\partial \mathbf{t}_l}{\partial \mathbf{t}_j} = \mathbb{1}(l = j)$.

2.3 Regularization

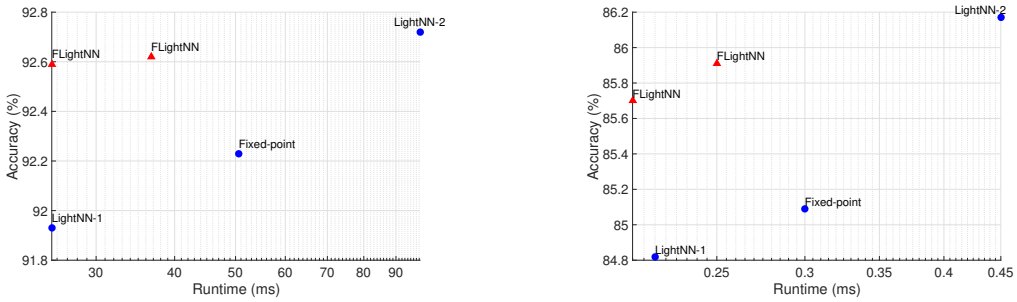
To encourage smaller $\mathbf{k}_i$ for the filters, we also add regularization loss: $\mathcal{L}_{reg,k}(\mathbf{w}) = \sum_{j=0}^{k-1} \lambda_j \sum_i ||\mathbf{r}_{i,j}||_2$ where λ_j performs as a handle to balance accuracy and model sparsity. This regularization loss is the sum of several group Lasso losses, since they can introduce structural sparsity [15]. The first item $\lambda_0 \sum_i ||\mathbf{r}_{i,0}||_2 = \lambda_0 \sum_i ||\mathbf{w}_i||_2$ is used to prune the whole filters out, while the other items ($j > 0$) regularize the residuals.

3 Experimental Results

We conduct experiments on both large and small CNNs for CIFAR-10 datasets. The configuration for Network-1 is: $(2 \times 128C3) - MP2 - (2 \times 256C3) - MP2 - (2 \times 512C3) - 10C3 - GP$, where $2 \times 128C3$ means two convolutional layers each with 128 3×3 filters; MP2 means max pooling with a 2x2 kernel; GP means global pooling. Network-2 reduces the number of filters per convolutional layer (except the last layer) of Network-1 by a factor of 8. For both networks, each convolutional layer is followed by a batch normalization layer and a Leaky ReLU activation function [13]. We use the Adam optimizer [11] to train the network. The learning rate starts at 5e-3, and decrease by 0.1 every 80 epochs. All the models are trained for 200 epochs. For FLightNN, we initialize the thresholds $\mathbf{t}$ to $\mathbf{0}$, and set the largest shifts k as 2. For all, except the 32-bit full-precision model, we use 8-bit fixed-point quantization for the activations. By varying λ, we can have different accuracy-runtime trade-offs for FLightNN. The results are shown in Table 1, where for each network we train two FLightNNs with different λ. We also implement and synthesize the models on Xilinx ZC706 FPGA, and measure the runtime of batched inference. In Table 1, the per-image runtime and speedup for the largest convolutional layer is reported. The accuracy-runtime trade-offs are also shown in Fig. 2.

Table 1: Accuracy and FPGA runtime for CIFAR-10. xWyA indicates x bits for weights and y bits for activations. The speedup compared to the full-precision model is shown in the last column.

ID	Param.	Model	Accuracy (%)	Storage (MB)	Runtime (ms)	Speedup
1	4.6M	Full-precision	92.85	18.5	769.2	1×
		LightNN-2_{8W8A}	92.72	4.6	98.3	7.8×
		LightNN-1_{4W8A}	91.93	2.3	25.5	30.2×
		Fixed-point$_{4W8A}$	92.23	2.3	50.6	15.2×
		FLightNN-2	92.59	2.3	25.5	30.2×
		FLightNN-2	92.62	3.3	36.7	21.0×
2	0.08M	Full-precision	86.36	0.31	3.16	1×
		LightNN-2_{8W8A}	86.17	0.08	0.45	7.0×
		LightNN-1_{4W8A}	84.82	0.04	0.22	14.4×
		Fixed-point$_{4W8A}$	85.09	0.04	0.30	10.5×
		FLightNN-2	85.70	0.04	0.21	15.0×
		FLightNN-2	85.91	0.06	0.25	12.6×



(a) (b)

Figure 2: Accuracy and runtime for (a) Network-1 and (b) Network-2 with different quantized models on CIFAR-10.

As shown in Table 1 and Fig. 2, FLightNN shows the advantage of flexible accuracy-speed trade-offs. For both networks, FLightNNs can achieve the accuracy close to LightNN-2, but have much higher speedup. Also, FlightNNs have higher accuracy and a lower speedup than LightNN-1. Thus, FLightNNs provide continuous trade-off curves for accuracy and speed. Compared to the fixed-point quantization, FLightNN can achieve higher accuracy, and up to $1.4\times$ speedup. This is because the multiplication is replaced by shift operators, which require much fewer resources on FPGA than multipliers. Therefore, the computation for FLightNN can be unrolled more times than that of fixed-point DNNs. By comparing the first FLightNN-2 of Network-2 with LightNN-1, we find that FLightNN-2 can achieve higher accuracy even with the same storage as LightNN-1. This is because initially FLightNN-2 quantizes all the filters with two shifts (since $\mathbf{t}$ is initialized as $\mathbf{0}$), and gradually adds constraints to the filters. This gradual quantization may be better than training a network with only one shift from scratch, as LightNN-1 does. The benefit of gradual quantization has also been observed by prior work [18, 8] which shows that gradually imposing quantization constraints can achieve better accuracy than directly quantizing with a strict constraint.

References

[1] Nnapi. `https://developer.android.com/ndk/guides/neuralnetworks/`. Accessed: 2018-10-15.

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[3] Ting-Wu Chin, Cha Zhang, and Diana Marculescu. Layer-compensated pruning for resource-constrained convolutional neural networks. *arXiv preprint arXiv:1810.00518*, 2018.

[4] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*, 2017.

[5] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

[6] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.

[7] Ruizhou Ding, Zeye Liu, Rongye Shi, Diana Marculescu, and RD Blanton. Lightnn: Filling the gap between conventional deep neural networks and binarized networks. In *Proceedings of the on Great Lakes Symposium on VLSI 2017*, pages 35–40. ACM, 2017.

[8] Yinpeng Dong, Renkun Ni, Jianguo Li, Yurong Chen, Jun Zhu, and Hang Su. Learning accurate low-bit deep neural networks with stochastic quantization. *arXiv preprint arXiv:1708.01001*, 2017.

[9] Jun Han and Claudio Moraga. The influence of the sigmoid function parameters on the speed of back-propagation learning. In *International Workshop on Artificial Neural Networks*, pages 195–201. Springer, 1995.

[10] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–800, 2018.

[11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] Darryl Lin, Sachin Talathi, and Sreekanth Annapureddy. Fixed point quantization of deep convolutional networks. In *International Conference on Machine Learning*, pages 2849–2858, 2016.

[13] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[14] Dimitrios Stamoulis, Anand Krishnan Prakash, Haocheng Fang, Sribhuvan Sajja, Mitchell Bognar, Diana Marculescu, et al. Designing adaptive neural networks for energy-constrained image classification. *arXiv preprint arXiv:1808.01550*, 2018.

[15] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2074–2082, 2016.

[16] Tien-Ju Yang, Yu-Hsin Chen, and Vivienne Sze. Designing energy-efficient convolutional neural networks using energy-aware pruning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6071–6079. IEEE, 2017.

[17] Yu Zhang, William Chan, and Navdeep Jaitly. Very deep convolutional networks for end-to-end speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 4845–4849. IEEE, 2017.

[18] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044*, 2017.

[19] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.