BMPQ: Bit-Gradient Sensitivity-Driven Mixed-Precision Quantization of DNNs from Scratch

Souvik Kundu, Shikai Wang, Qirui Sun, Peter A. Beerel, Massoud Pedram
Electrical and Computer Engineering, University of Southern California, Los Angeles, USA
{souvikku, shikaiwa, qiruisun, pabeerel, pedram}@usc.edu

***Abstract*—Large DNNs with mixed-precision quantization can achieve ultra-high compression while retaining high classification performance. However, because of the challenges in finding an accurate metric that can guide the optimization process, these methods either sacrifice significant performance compared to the 32-bit floating-point (FP-32) baseline or rely on a compute-expensive, iterative training policy that requires the availability of a pre-trained baseline. To address this issue, this paper presents BMPQ, a training method that uses *bit gradients* to analyze layer sensitivities and yield *mixed-precision quantized* models. BMPQ requires a single training iteration but does not need a pre-trained baseline. It uses an integer linear program (ILP) to dynamically adjust the precision of layers during training, subject to a fixed hardware budget. To evaluate the efficacy of BMPQ, we conduct extensive experiments with VGG16 and ResNet18 on CIFAR-10, CIFAR-100, and Tiny-ImageNet datasets. Compared to the baseline FP-32 models, BMPQ can yield models that have $15.4\times$ fewer parameter bits with negligible drop in accuracy. Compared to the SOTA "during training", mixed-precision training scheme, our models are $2.1\times$, $2.2\times$, and $2.9\times$ smaller, on CIFAR-10, CIFAR-100, and Tiny-ImageNet, respectively, with an improved accuracy of up to 14.54%.**

***Index Terms*—Mixed-precision quantization, model compression, energy-efficient DNN training, one-shot quantization**

I. Introduction

The prohibitively large computation and storage costs of current deep neural network (DNN) models pose a significant challenges in deployment of many large DNN models on resource-constrained IoT and edge devices. To address these issues, researchers have primarily focused on reducing the parameter budgets through various model compression techniques [1]–[6]. In particular, *quantization* has proven to be a promising model compression scheme, that can essentially perform similar to the 32-bit floating-point (FP-32) parameter model and/or activation maps with low-precision, quantized, fixed-point values, requiring reduced storage and data-transfer costs than the FP-32 baselines.

Early works of of binary neural networks (BNNs) [7] and XNOR-net [8] with *homogeneous-precision* (HPQ) models[1] showed the potential benefits of quantization. To address the issue of significant accuracy sacrifice of the HPQ models, more recent works have demonstrated the *mixed-precision* quantization (MPQ) in which different layers can be assigned different bit widths based on the layer significance evaluated through various metrics, including Hessian spectrum [9], [10].

Most of the sensitivity-driven methods require the presence of a baseline FP-32 pre-trained model. Alternatively, quantization methods that rely on neural architecture search (NAS) [5], [11], to translate the model compression problem to a search problem of efficient bit-width assignment to different layers, require a compute-intensive search procedure that is added on top of the training. Recently, researchers [12] have used intermediate activation densities to estimate the layer sensitivity and assign quantization bit widths, however, have not re-evaluated bit-width assignments, limiting performance. Moreover, their quantization method did not necessarily yield models satisfying a target hardware constraint. Meanwhile, the increased demand for data-privacy has increased the need for on-device training and fine-tuning [13]. This trend makes many of these quantization-aware training methods impossible on resource-constrained devices. An alternative can be to rent costly GPU clusters, use quantization on private data, transfer the quantized model to resource-constrained device, and finally remove sensitive data from the server cluster before terminating the session. However, the prohibitively expensive training time may significantly increase the cluster cost, that generally charges on hourly basis. This motivates the development of efficient training solutions that can yield mixed-precision quantized models with no baseline pre-training or iterative training.

Our contribution. We present an efficient "during training" MPQ method that does not require a pre-trained model. To effectively search the large design space of MPQ, we decompose the core problem into two sub-problems. First, we use a novel *bit-gradient-analysis* driven layer-sensitivity evaluation to rank the layers based on their significance. Second, using this information and a specific target hardware constraint, we formulate an integer linear program (ILP) to decide the bit-precision of each layer. This combination of steps becomes the core of our bit-gradient sensitivity driven MPQ method (BMPQ). BMPQ can be integrated into most training methods without any significant increase in training time because these extra steps are relatively low cost and are performed at regular intervals separated by multiple epochs of normal training.

To demonstrate the efficacy of the proposed method, we perform extensive experimental evaluations on CIFAR-10, CIFAR-100, and Tiny-ImageNet with quantized VGG16 and ResNet18 models. Our experiments show that BMPQ can yield quantized models that require up to $15.4\times$ less storage compared to the FP-32 counterparts with comparable classification performance. Compared to the existing single-shot quantization approach [12], our models can yield up to $2.9\times$ higher compression with an accuracy improvement of up to 14.54%.

II. Related Work

A. Quantization

Quantization has improved the trade-off between accuracy and efficiency of NN models. Early works [14] used rule-based strategies that required human expertise to quantize a model. Subsequent works focused on HPQ [8], [15] but often yielded

[†]This work was supported in parts by NSF and DARPA with grant numbers 1763747 and HR00112190120, respectively.

[1]All layer weights/activations have the same bit widths.

reduced accuracy compared to the FP-32 baseline. To address this issue, works including non-uniform quantization [16], channel-wise quantization [17], and progressive quantization-aware fine-tuning [18] were proposed.

Recently, researchers have proposed using layer significance to guide quantization of different bit widths to different layers and thus boost the accuracy. However, the space of possible assignments is large. In particular, for a model of L layers with N_B bit-width choices, there can be $(N_B)^L$ options to consider. To handle this issue, [5] converted the quantization problem into a reinforcement learning problem based on an actor-critic model. [11] proposed a differentiable neural architecture search. Others use sensitivity analysis metrics [9], [10] to determine the layer importance and bit-width assignment but require a FP-32 pre-trained model or rely on iterative training [6]. Thus, despite all these efforts, a single-shot during-training MPQ approach that can yield similar to baseline performance while achieving ultra high compression has been largely missing.

B. PACT Non-Linearity Function for Activation

The unbounded nature of the ReLU non-linear function can introduce significant approximation error when quantifying activations, particularly for low bit-precision activations [19]. Alternatively, clipped ReLU activations can provide bounded output, but finding a global clipping factor that maintains the model accuracy is quite challenging [19]. A Parameterized Clipping Activation (PACT) function [20] with a per-layer parameterized clipping level α has been proven effective in the low-precision domain. For an input a_i, PACT non-linearity produces an output a_o as follows:

$$a_o = 0.5(|a_i| - |a_i - \alpha| + \alpha) \quad = \begin{cases} 0, & a_i \in (-\infty, 0) \\ a_i, & a_i \in [0, \alpha) \\ \alpha, & a_i \in [\alpha, +\infty) \end{cases} \tag{1}$$

The output is then linearly quantized to provide ${a_o}^q$. The computation of ${a_o}^q$ for a k-bit activation is

$${a_o}^q = \text{round}(a_o \cdot \frac{2^k - 1}{\alpha}) \frac{\alpha}{2^k - 1} \tag{2}$$

The resemblance of PACT with ReLU increases with the increase of the value α. Note that α is a trainable parameter and, during backpropagation, $\frac{\partial {a_o}^q}{\partial \alpha}$ is computed by using a straight-through estimator (STE) (see [21] for details).

III. Methodology

A. Notation

Let a L-layer DNN be denoted by $\mathbf{Y} = f(\mathbf{X}; \mathbf{W})$, with each layer l parameterized by $\mathbf{W}_l$. A quantized version of f is denoted as $f^Q(\mathbf{X}^Q; \mathbf{W}^Q)$ in which each layer l is parameterized by $\mathbf{W}_l^{q_l}$ with input activation tensor $\mathbf{A}_l^{q_l}$. We train our model to minimize a loss $\mathcal{L}$ such that f^Q can closely mimic f, and thus minimize any performance degradation. For HPQ, q_l is fixed for any l, whereas for MPQ, q_l may vary with l.

Definition 1. *Support bit widths*: For mixed-precision quantization, we define the support bit widths S_q as the set of possible bit widths that can be assigned to the parameters of any layer l of the model, excluding the first and last layers that are fixed to 16 bits as in [12].

B. Loss Bit Gradient

The gradient of the loss with respect to a weight scalar $\frac{\partial \mathcal{L}}{\partial w}$ indicates the direction that reduces the output error at the highest rate. Moreover, larger magnitude gradients lead to more significant changes in the weights, and thus correlate well with larger weight significance [22]. Inspired by this observation, we extend the notion of loss gradients to the bit-level for a quantized weight tensor, and propose a layer sensitivity metric that is driven by normalized bit gradients (NBG).

For the l^{th} layer of a DNN, the quantization of a floating-point tensor $\mathbf{W}_l$ to a fixed-point (signed) tensor $\mathbf{W}_l^{q_l}$ is

$$S_{w_l} = \max(|\mathbf{W}_l|)/(2^{q_l - 1} - 1); \mathbf{W}_l \in \mathbb{R}^{d_l} \tag{3}$$

$$\mathbf{W}_l^{q_l} = \text{round}(\mathbf{W}_l / S_{w_l}) \cdot S_{w_l}, \tag{4}$$

where d_l is the tensor dimension and S_{w_l} is the quantization scaling factor. The quantized weights have a staircase function which is non-differentiable. To solve this problem we use STE, similar to other quantization methods [9], [10]. To reduce storage, it is recommended to store the fixed-point $(\mathbf{W}_l^{q_l}/S_{w_l}) \in \{-2^{q_l - 1}, ..., 2^{q_l - 1}\}^{d_l}$ instead of $\mathbf{W}_l^{q_l}$.

As is typical, we convert the scaled quantized weights to their corresponding 2's complement representation [23] as given by

$$w_l^{q_l}/S_{w_l} = -2^{q_l - 1} \cdot b_{q_l - 1} + \sum_{i=0}^{q_l - 2} 2^i \cdot b_i. \tag{5}$$

The loss bit gradient for each bit position is then derived as

$$\nabla_b \mathcal{L} = [\frac{\partial \mathcal{L}}{\partial b_{q_l - 1}}, \frac{\partial \mathcal{L}}{\partial b_{q_l - 2}}, ..., \frac{\partial \mathcal{L}}{\partial b_0}] \tag{6}$$

where

$$\frac{\partial \mathcal{L}}{\partial b_i} = \frac{\partial \mathcal{L}}{\partial w_l^{q_l}} \cdot \frac{\partial w_l^{q_l}}{\partial b_i}. \tag{7}$$

For a layer l with maximum support bit width q_{max}, we compute the loss bit gradient for each of the q_{max} bits, yielding a $d_l \times q_{max}$ floating-point matrix. We sum the absolute values of each row to produce a $d_l \times 1$ vector. The layer NBG is set to be the average value of this vector. Finally, we compute the epoch-normalized bit gradient (ENBG) of a layer as the mean of its NBGs over i epochs. First-order derivatives like gradients can sometimes fail to capture the importance of a weight value based on its magnitude. However, the amortized nature of the ENBG computation over all weights and epochs makes the probability of this phenomenon occurring vanishingly small.

Definition 2. *Epoch Intervals (EI):* We define the epoch intervals as the ranges of epochs over which we collect the NBG of each layer to calculate the ENBG. For training a model with periodic epoch intervals, the k^{th} interval starts with the $(\sum_{i=0}^{k-1} ep_{int}^i + 1)^{th}$ epoch where $ep_{int}^i = ep_{int}$ for any i. For aperiodic epoch intervals, we can use different ep_{int}^i for different interval indices i. We use periodic EI $ep_{int} = 20$.

C. ILP-Driven Iterative Bit-Width Assignment

After each epoch interval, we re-assign the bit widths to maximize performance by formulating an ILP that maximizes the total layer sensitivity subject to a given hardware constraint C. In particular, to capture the bit-width assignment at the start of the k^{th} interval for a layer l, we introduce a constrained

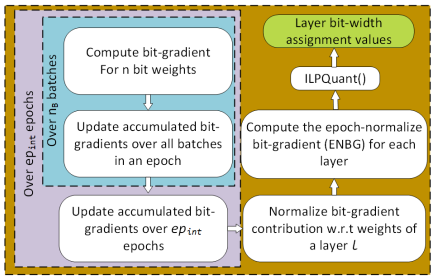

Fig. 1. Step-wise description of layer bit-width evaluation.

integer variable ${\Omega^k}_l$, which is multiplied by the negative ENBG of that layer g_1^{k-1}. Next we solve the following problem:

$$\text{Objective: minimize} \sum_{l=0}^{L-1} (g_l^{k-1}.{\Omega^k}_l), \tag{8}$$

$$\text{Subject to:} \sum_{l=0}^{L-1} \Phi(\Psi\{{\Omega^k}_l\}) \leq C \tag{9}$$

Here, $\Psi(.)$ is a function that translates the assignment variable ${\Omega^k}_l$ to the corresponding bit width q_l^k and $\Phi(.)$ is a function that translates the bit width to a cost associated with the layer. For example, if C is a memory-constraint, then $\Phi(.)$ converts the bit-width assignment to a measure of memory usage. Notice that, in reality g_l^{k-1} is computed assuming the bit-width assignments of other layers are fixed despite the fact that they may change after the ILP. This approximation enables our ILP optimization to search over a large combinatorial space and any error associated with the approximation is mitigated by the repeated re-evaluation of the ILP after each epoch interval.

Recently, an iterative training approach [10] has also used an ILP to assign layer bit widths but only as a post-training optimization.[2] Moreover, reference [10] computed a layer-variable coefficient based on an L_2-norm of the difference between FP-32 and quantized weights of that layer, which may further increase the storage overhead. In contrast, we use the negative ENBG values as coefficients of the respective ${\Omega^k}_l$ that does not require any L_2-difference compute/storage cost.

D. BMPQ Training

For the initial ep_w warm-up epochs, we train the model with each layer quantized to $max(N_1, N_2, \ldots, N_m)$ bits where $S_q = [N_1, \ldots, N_m]$. Throughout the training, we follow the recommendation of [12] to quantize the activation of layer l with the same number of bits as that used for the weights of that layer. During the weight update, we first allow the weights to be the updated FP-32 values, and then quantize to a specific bit-precision and compute the loss based on forward-pass evaluation with the quantized weights/activations. We use the ReLU non-linearity for the last layer and use the PACT nonlinearity for all other intermediate layers with low bit-precision. For a bit width $N_i = 2$, we use ternary weight quantization [24] to minimize the Euclidean distance between the FP-32 and quantized weights. The fact that we perform quantized training for ep_{int} epochs after every bit-width assignment iteration helps us avoid post-training fine-tuning operations to maintain accuracy. The step-wise details of the evalENBG(.) function is presented in Fig. 1.

[2]We do not quantitatively compare our work with this work because this work uses an iterative (rather than a single-shot) training approach and thus it does not constitute an "apple-to-apple" comparison to our work.

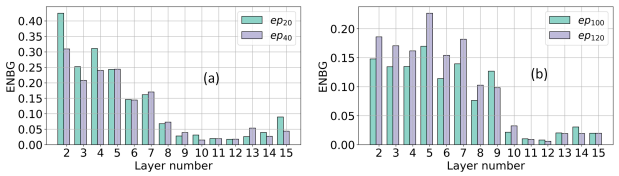

Fig. 2. (a), (b) Layer sensitivities based on ENBG for VGG16 on CIFAR10. ep_i indicates that the normalization was performed after i^{th} epoch.

IV. Experiments

A. Experimental Setup

Models and Datasets. We selected three widely used datasets, CIFAR-10, CIFAR-100, and Tiny-ImageNet and chose popular CNN models for image classification, VGG16 [25] and ResNet18 [26]. For all the datasets we used standard data augmentations (horizontal flip and random crop with reflective padding) to train the models with a batch size of 128.

BMPQ training settings. For CIFAR-10 and CIFAR-100, we trained our models for 200 epochs with initial learning rate (LR) of 0.1 that decayed by 0.1 after 80 and 140 epochs. For Tiny-ImageNet, we used 100 training epochs and used decay epochs as 40 and 70 keeping other hyperparameters the same as that for CIFAR. We used S_q of 4 and 2 bits for all our training but fixed the first and last layers to have 16-bits. For the ResNet models, we ensured the downsampling layers have the same bit-width assignment as its input layer [12].

B. Results

Table I shows the performance of the BMPQ trained models compared to their respective full-precision counterparts. In particular, for CIFAR-10 dataset the BMPQ trained models can yield a model compression ratio[3] of up to $15.4\times$ while sacrificing a drop in absolute accuracy of up to only 0.70%. Similarly, for CIFAR-100 and Tiny-ImageNet, the BMPQ generated models provide near full precision performances while yielding a compression ratio of up to $15.4\times$ and $10\times$, respectively. This clearly shows the efficacy of BMPQ generated models to retain baseline performance while requiring lower model storage cost.

Analysis of ENBG at various iterations. We analyzed the ENBG snapshots of VGG16 (on CIFAR-10) at the end of different epoch values. In particular, we chose two early-stage training epochs 20 and 40 and two mid-level training epochs 100 and 120. As shown in Fig. 2(a), the ENBG represented layer sensitivity changes significantly between epoch 20 and 40, hinting at the need for iterative re-evaluation during training. Also, a similar trend is observed at the mid-level training epochs (Fig. 2(b)), that forces the ILP to re-assign the 10^{th} and 14^{th} layer from 2-b to 4-b and 4-b to 2-b, respectively.

C. Comparison with Single-Shot Training

Table II presents the comparison of the presented BMPQ with the recently proposed single-shot MPQ method [12] on CIFAR-10[4], CIFAR-100, and Tiny-ImageNet. In particular, we

[3]We define compression ratio as the ratio of bits required to store a FP-32 model to that required by the BMPQ model of same architecture.

[4]The original paper used VGG19 compared to VGG16 of ours.

Dataset	Model	layer-wise bit width	Test Acc (%)	Compression ratio (r_M^{32})
CIFAR-10	VGG16	Full precision	93.9	1×
		[16, 4, 4, 4, 4, 4, 4, 4, 4, 2, 2, 2, 2, 4, 16]	93.56	10.5×
		[16, 4, 2, 4, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 16]	93.21	15.4×
	ResNet18	Full precision	95.14	1×
		[16, 2, 2, 4, 2, 4, 4, 4, 2, 2, 2, 2, 2, 2, 16]	94.54	13.4×
CIFAR-100		Full precision	73	1×
		[16, 4, 4, 4, 4, 2, 4, 2, 2, 2, 2, 2, 2, 4, 16]	72.2	14.6×
		[16, 4, 2, 4, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 16]	71.26	15.4×
	ResNet18	Full precision	77.5	1×
		[16, 2, 2, 4, 2, 4, 4, 4, 2, 4, 4, 4, 4, 4, 2, 16]	75.98	9.4×
Tiny-ImageNet	VGG16	Full precision	60.82	1×
		[16, 4, 4, 4, 4, 4, 4, 4, 2, 2, 4, 4, 2, 4, 16]	59.29	10×
	ResNet18	Full precision	64.15	1 ×
		[16, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 4, 16]	63.27	8.8×

TABLE I
PERFORMANCE OF BMPQ GENERATED MODELS COMPARED TO THE RESPECTIVE BASELINE FULL PRECISION (FP-32) MODELS.

can see that BMPQ models provide an improved accuracy of up to 14.54% with up to 2.9× less parameter bits. Note, considering the training epochs of [12], to have a fair comparison we report the accuracy of our models after 120, 120, and 60 epochs for CIFAR-10, CIFAR-100, and Tiny-ImageNet, respectively.

Model	Dataset	AD [12] Acc (%)	BMPQ Acc (%)	Improved compression
VGG16	CIFAR-10	91.62	**92.28**	2.1×
ResNet18	CIFAR-100	71.51	**73.96**	2.2×
ResNet18	Tiny-ImageNet	44	**58.54**	2.9×

TABLE II
COMPARISON WITH SINGLE-SHOT MPQ ACHIEVED THROUGH ANALYSIS OF ACTIVATION DENSITY (AD).

D. Discussion

Memory saving for inference. Let layer l of an L-layer model have p_l parameters. Represented using homogeneous FP-32, the total storage requirement (MB) of the model weights can be given by

$$M_{fp32} = 4 * (\sum_{i=0}^{L-1} \frac{p_l}{2^{20}}). \tag{10}$$

For a BMPQ generated model, the weight storage cost (MB) and corresponding compression ratio r_M^{32} (and r_M^{16} compared to a 16-b baseline) can be computed as

$$M_{BMPQ} = (\frac{4}{32}) * (\sum_{i=0}^{L-1} \frac{p_l \cdot q_l}{2^{20}}) \tag{11}$$

$$r_M^{32} = \frac{M_{fp32}}{M_{BMPQ}} \text{ and } r_M^{16} = 0.5 * r_M^{32}. \tag{12}$$

Note that, for each layer, we need to store only one scaling factor in FP-32. Hence, its overhead is negligible and ignored. Column 5 in Table I shows the corresponding r_M^{32} for our BMPQ models.

V. CONCLUSIONS

This paper presented a MPQ training method driven by epoch-normalized bit-gradients without the requirement of any pre-training. Our proposed ENBGs capture the sensitivity of DNN layers and drive an ILP formulation that iteratively assigns MPQ bit widths after certain epochs during training. Our results demonstrated the efficacy of BMPQ models when compared to both FP-32 baselines and existing single-shot MPQ schemes. With the growing demand of privacy-preserving, on-device training and inference, we believe this work will act as a foundation for energy-efficient, on-device quantization.

REFERENCES

[1] S. Kundu, M. Nazemi, P. A. Beerel, and M. Pedram, "DNR: A tunable robust pruning framework through dynamic network rewiring of DNNs," in *Proceedings of the 26th Asia and South Pacific Design Automation Conference*, 2021, pp. 344–350.

[2] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model compression and hardware acceleration for neural networks: A comprehensive survey," *Proceedings of the IEEE*, vol. 108, no. 4, pp. 485–532, 2020.

[3] S. Kundu, Q. Sun, Y. Fu, M. Pedram, and P. A. Beerel, "Analyzing the confidentiality of undistillable teachers in knowledge distillation," *Advances in Neural Information Processing Systems (NeurIPS 2021)*, vol. 34, 2021.

[4] S. Kundu, M. Nazemi, M. Pedram, K. M. Chugg, and P. A. Beerel, "Pre-defined sparsity for low-complexity convolutional neural networks," *IEEE Transactions on Computers*, vol. 69, no. 7, pp. 1045–1058, 2020.

[5] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "HAQ: Hardware-aware automated quantization with mixed precision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8612–8620.

[6] H. Yang, L. Duan, Y. Chen, and H. Li, "BSQ: Exploring bit-level sparsity for mixed-precision neural network quantization," *arXiv preprint arXiv:2102.10462*, 2021.

[7] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," *Advances in neural information processing systems*, vol. 29, 2016.

[8] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: Imagenet classification using binary convolutional neural networks," in *European conference on computer vision*. Springer, 2016, pp. 525–542.

[9] Z. Dong, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer, "HAWQ: Hessian aware quantization of neural networks with mixed-precision," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 293–302.

[10] Z. Yao, Z. Dong, Z. Zheng, A. Gholami, J. Yu, E. Tan, L. Wang, Q. Huang, Y. Wang, M. Mahoney *et al.*, "HAWQ-V3: Dyadic neural network quantization," in *International Conference on Machine Learning*. PMLR, 2021, pp. 11 875–11 886.

[11] B. Wu, Y. Wang, P. Zhang, Y. Tian, P. Vajda, and K. Keutzer, "Mixed precision quantization of ConvNets via differentiable neural architecture search," *arXiv preprint arXiv:1812.00090*, 2018.

[12] K. Vasquez, Y. Venkatesha, A. Bhattacharjee, A. Moitra, and P. Panda, "Activation density based mixed-precision quantization for energy efficient neural networks," *DATE*, 2021.

[13] A. Inc., "An on-device deep neural network for face detection," Nov 2017. [Online]. Available: https://machinelearning.apple.com/research/face-detection

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

[15] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," *arXiv preprint arXiv:1612.01064*, 2016.

[16] D. Zhang, J. Yang, D. Ye, and G. Hua, "LQ-Nets: Learned quantization for highly accurate and compact deep neural networks," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 365–382.

[17] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," *arXiv preprint arXiv:1806.08342*, 2018.

[18] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless CNNs with low-precision weights," *arXiv preprint arXiv:1702.03044*, 2017.

[19] J. Choi, S. Venkataramani, V. Srinivasan, K. Gopalakrishnan, Z. Wang, and P. Chuang, "Accurate and efficient 2-bit quantized neural networks." in *MLSys*, 2019.

[20] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, "PACT: Parameterized clipping activation for quantized neural networks," *arXiv preprint arXiv:1805.06085*, 2018.

[21] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[22] T. Dettmers and L. Zettlemoyer, "Sparse networks from scratch: Faster training without losing performance," *arXiv preprint arXiv:1907.04840*, 2019.

[23] A. S. Rakin, Z. He, and D. Fan, "Bit-flip attack: Crushing neural network with progressive bit search," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1211–1220.

[24] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," *arXiv preprint arXiv:1605.04711*, 2016.

[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[26] 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.