# Discovering Low-Precision Networks Close to Full-Precision Networks for Efficient Inference

**Jeffrey L. McKinstry**[*] **Steven K. Esser** **Rathinakumar Appuswamy** **Deepika Bablani**

**John V. Arthur** **Izzet B. Yildiz** **Dharmendra S. Modha**

IBM Research

## Abstract

To realize the promise of ubiquitous embedded deep network inference, it is essential to seek limits of energy and area efficiency. Low-precision networks offer promise as energy and area scale down quadratically with precision. We demonstrate 8- and 4-bit networks that meet or exceed the accuracy of their full-precision versions on the ImageNet classification benchmark. We hypothesize that gradient noise due to quantization during training increases with reduced precision, and seek ways to overcome this. The number of iterations required by SGD to achieve a given training error is related to the square of (a) the distance of the initial solution from the final and (b) the maximum variance of the gradient estimates. Accordingly, we reduce solution distance by starting with pretrained fp32 baseline networks, and combat noise introduced by quantizing weights and activations during training by training longer and reducing learning rates. Sensitivity analysis indicates that these techniques, coupled with activation function range calibration, are sufficient to discover low-precision networks close to fp32 precision baseline networks. Our results provide evidence that 4-bits suffice for classification.

## 1 Introduction

To harness the power of deep networks in embedded and large-scale application domains requires energy-efficient implementation, leading to great interest in low-precision networks suitable for deployment with low-precision hardware accelerators. Consequently there have been a flurry of methods for quantizing both the weights and activations of these networks (Jacob et al., 2017; Courbariaux et al., 2015; Polino et al., 2018; Xu et al., 2018; Baskin et al., 2018; Mishra et al., 2017; Choi et al., 2018). A common perception is that 8-bit networks offer the promise of decreased computational complexity with little loss in accuracy, without need to retrain. However, the published accuracies are typically lower for quantized networks than for the corresponding full-precision net (Migacz, 2017). Even training 8-bit networks from scratch fails to close this gap (Jacob et al., 2017) (See Table 1). At 4 bits, for ImageNet, only one method has been able to match the accuracy of the corresponding full-precision network when quantizing both weights and activations (Zhuang et al., 2018). The complexity and ad-hoc nature of prior methods motivates the search for a simpler technique that consistently matches full-precision baseline networks.

Guided by theoretical convergence bounds for stochastic gradient descent (SGD), we discover that Fine-tuning – training pre-trained high-precision networks for low-precision inference – After Quantization – proper range calibration of weights and activations – can discover 8- and 4-bit integer networks. We evaluate the proposed technique on the ImageNet benchmark on a representative set

[*]Correspondance to jlmckins@us.ibm.com

Table 1: **FAQ exceeds or matches fp32 network accuracy on Imagenet for 8 and 4 bits, outperforming all previous approaches in all but one instance.** Baselines are popular architectures from the PyTorch model zoo. Other results reported in literature shown for comparison, with methods exceeding or matching their top-1 baseline in bold. Precision in bits, w= weight, a= activation.

| **Network** | **Method** | **Precision (w,a)** | **Accuracy (% top-1)** | **Accuracy (% top-5)** |
|---|---|---|---|---|
| ResNet-18 | baseline | 32,32 | 69.76 | 89.08 |
| **ResNet-18** | **Apprentice** | **4,8** | **70.40** | **-** |
| **ResNet-18** | **FAQ (This paper)** | **8,8** | **70.02** | **89.32** |
| **ResNet-18** | **FAQ (This paper)** | **4,4** | **69.78±0.04** | **89.11±0.03** |
| **ResNet-18** | Joint Training | 4,4 | 69.3 | - |
| ResNet-18 | UNIQ | 4,8 | 67.02 | - |
| ResNet-18 | Distillation | 4,32 | 64.20 | - |
| ResNet-34 | baseline | 32,32 | 73.30 | 91.42 |
| **ResNet-34** | **FAQ (This paper)** | **8,8** | **73.71** | **91.63** |
| **ResNet-34** | **FAQ (This paper)** | **4,4** | **73.31** | **91.32** |
| ResNet-34 | UNIQ | 4,32 | 73.1 | - |
| ResNet-34 | Apprentice | 4,8 | 73.1 | - |
| ResNet-34 | UNIQ | 4,8 | 71.09 | - |
| ResNet-50 | baseline | 32,32 | 76.15 | 92.87 |
| **ResNet-50** | **FAQ (This paper)** | **8,8** | **76.52** | **93.09** |
| **ResNet-50** | **FAQ (This paper)** | **4,4** | **76.25** | **93.00** |
| **ResNet-50** | **EL-Net** | **4,4** | **75.9** | **92.4** |
| ResNet-50 | IOA | 8,8 | 74.9 | - |
| ResNet-50 | Apprentice | 4,8 | 74.7 | - |
| ResNet-50 | UNIQ | 4,8 | 73.37 | - |
| ResNet-152 | baseline | 32,32 | 78.31 | 94.06 |
| **ResNet-152** | **FAQ (This paper)** | **8,8** | **78.54** | **94.07** |
| **ResNet-152** | **FAQ (This paper)** | **4,4** | **78.35** | **94.11** |
| Inception-v3 | baseline | 32,32 | 77.45 | 93.56 |
| **Inception-v3** | **FAQ (This paper)** | **8,8** | **77.60** | **93.59** |
| Inception-v3 | FAQ (This paper) | 4,4 | 77.33 | 93.59 |
| Inception-v3 | IOA | 8,8 | 74.2 | 92.2 |
| Densenet-161 | baseline | 32,32 | 77.65 | 93.80 |
| **Densenet-161** | **FAQ (This paper)** | **4,4** | **77.90** | **93.83** |
| **Densenet-161** | **FAQ (This paper)** | **8,8** | **77.84** | **93.91** |
| VGG-16bn | baseline | 32,32 | 73.36 | 91.50 |
| **VGG-16bn** | **FAQ (This paper)** | **4,4** | **73.87** | **91.67** |
| **VGG-16bn** | **FAQ (This paper)** | **8,8** | **73.66** | **91.56** |

of networks (Table 1) and demonstrate 8-bit scores exceeding fp32 scores after just one epoch of fine-tuning. We also present evidence of 4 bit, fully integer networks, that match the accuracy of the original fp32 networks, setting the new state-of-the-art.

Our goal is to quantize existing networks to 8 and 4 bits for both weights and activations, without increasing the computational complexity of the network to compensate, e.g. with modifications such as feature expansion, while achieving accuracies that at least match the original network. We aim to find a way to train low-precision networks to obtain the best possible score subject to capacity limits.

We hypothesize that noise introduced by quantizing weights and activations during training is the crux of the problem and is a second source of noise similar to gradient noise inherent to SGD (Polino et al. (2018)). The problem is then to find ways to overcome this noise. SGD requires [2]

$$k \leq (\sigma^2 + L * \|x_0 - x^*\|_2^2)^2/\epsilon^2 \tag{1}$$

iterations to find a $2\epsilon$-approximate optimal value, where $\sigma^2$ is the gradient noise level, $L$ is related to the curvature of the convex function, $x_0$ and $x^*$ are the initial and optimal network parameters,

[2] This assumes a convex loss function, a simpler case.

respectively, and $\epsilon$ is the error tolerance (Meka, 2017). This suggests two ways to minimize final error. First, start closer to the solution, i.e. minimize $x_0 - x^*$. We therefore start with pretrained models rather than training from scratch (Zhou et al., 2017; Baskin et al., 2018). Second, minimize $\sigma^2$. To do this, we combine well-known techniques to combat noise: 1) larger batches which reduce gradient noise proportional to the square root of batch size (Goodfellow et al., 2016), and 2) learning rate annealing to lower learning rates ($10^{-6}$), effectively averaging over more batches (batch size increases and learning rate decreases are known to behave similarly (Smith et al., 2017)). Additionally, in the case of 4-bit precision, we fine-tune longer to achieve better accuracy, according to equation 1. Finally, we use the initial pretrained network to determine the proper ranges for quantizing both the weights and activations. We refer to this technique as Fine-tuning after quantization, or FAQ.

## 2 Background

There are a variety of methods for producing low-precision networks. These include allowing non-uniform quantization of weights and activations (Miyashita et al., 2016; Zhou et al., 2017), and stochastic quantization (Polino et al., 2018; Courbariaux et al., 2015). Approaches to training include distillation (Polino et al., 2018), layer-wise quantization and retraining (Xu et al., 2018), introducing noise during training (Baskin et al., 2018), increasing features (Mishra et al., 2017), learning quantization-specific parameters using backpropagation (Choi et al., 2018), and fine-tuning (Baskin et al., 2018; Zhuang et al., 2018), among others.

We focus on training networks with weights and activations constrained to be either 4 bit, or 8-bit fixed-point integers, and restrict all other scalar multiplicative constants (for example, batch-normalization) in the network to be 8-bit integers and additive constants (for example, bias values) to be 32-bit integers. Compared to previous work, our approach offers a major advantage in the 8-bit space, by requiring only a single epoch of post quantization training, and in the 4-bit space by matching baseline scores with a simpler approach, exceeding published results on several state-of-the-art networks.

## 3 Low-precision Fine-tuning Methods

We start with pretrained, high-precision networks from the PyTorch model zoo, quantize, and then fine-tune for a variable number of epochs depending on precision. The quantizer we use is parametrized by the precision (in number of bits) $b$, and the location of the least significant-bit relative to the radix $l$, denoted by $Q_{b,l}$. A calibration phase during initialization is used to determine a unique $l$ for each layer of activations, which remains fixed subsequently throughout fine-tuning. Similarly, each layer's weights as well as other parameters are assigned a unique $l$ and this quantity is determined during each training iteration. The procedures for determining $l$ for activations and other parameters are described in the following subsections. A given scalar $x$ is quantized to a fixed-point integer $\hat{x} = Q_{b,l}(x) = \min(\lfloor x \times 2^{-l} \rceil, 2^b - 1) \times 2^l$ for unsigned values, and $\hat{x} = \max(\min(\lfloor x \times 2^{-l} \rceil, 2^{b-1} - 1), -2^{b-1} + 1)) \times 2^l$ for signed values.

All network weights and activations are quantized to 4 or 8 bits except first and last layer weights and the input to last layer, which are kept at 8 bit as is common in literature. The output of the last fully connected layer is full precision (Courbariaux et al. (2015); Esser et al. (2016)).

Given a weight tensor $w$, a fixed-point version is used for inference and gradient calculation, obtained by applying $Q_{b,l}$ element-wise. The quantization parameter $l$ for a weight tensor is updated during every iteration and computed as follows: We first determine a desired quantization step-size $\Delta$ by first clipping the weight tensor at a constant multiple[3] of its numerically estimated standard-deviation, and then dividing this range into equally-sized bins. Finally, the required constant $l$ is calculated as $l = \lceil \log_2(\Delta) \rceil$. All other parameters, including those used in batch-normalization, use $l = -b/2$.

Networks are initialized from available pretrained models. Next, the quantization parameter $l$ for each layer of activation is calibrated using the following procedure: Following Jacob et al. (2017), we run several (5) training data batches through the unquantized network to determine the maximum range for uniform quantization. For each layer, $y_{max}$ is the maximum across all batches of the 99.99th percentile of the batch of activation tensor of that layer, rounded up to the next even power of two. This percentile level was found to give the best initial validation score for 8-bit, while 99.9 was

[3]The constant, in general, depends on the precision. We used a constant of 4.12 for all our 4-bit experiments.

best for 4-bit ReLUs. The estimated $y_{max}$ determines the quantization parameter $l$ for that tensor. For ReLU layers, the clipped tensor in the range $[0, y_{max}]$ is then quantized using $Q_{b,l}$. Activation function parameters for training are kept fixed during fine-tuning.

To train a quantized network we use the typical procedure of keeping an fp32 copy of the weights which are updated with the gradients, and quantize weights and activations in the forward pass (Courbariaux et al., 2015; Esser et al., 2016). We also use the straight through estimator (Bengio et al., 2013) to pass the gradient through the quantization operator. For 8-bit networks, we need only a single additional epoch of training, with a learning rate of $10^{-4}$. For 4-bit networks we trained for 110 epochs using exponential learning rate decay starting from the initial rate of 0.0015 (slightly higher than the final learning rate used to train the pretrained net) to a final value of $10^{-6}$ [4]. The batch size used was 256. SGD with momentum was used for optimization.

## 4 Experiments

FAQ trained 8-bit networks outperform all comparable quantization methods in all but one instance and exceed fp32 network accuracy for all networks explored. Following quantization, the accuracy was very close to the fp32 networks (data not shown) with one exception (Inception-v3). Hence, they did not require extensive fine-tuning. FAQ trained 4-bit network accuracy exceeds all comparable quantization methods, surpassing the next closest approach by nearly $0.5\%$ for ResNet-18 (Jung et al., 2018), and at least matched fp32 accuracy. In contrast to 8-bit, accuracy dropped precipitously following quantization, requiring significant fine-tuning to match pretrained networks.

4-bit network accuracy is sensitive to several hyperparameters. Table 2 demonstrates sensitivity experiments. Our findings from these experiments are summarized below.

- For the 4-bit ResNet-18, longer fine-tuning improved accuracy, potentially by averaging out gradient noise introduced by discretization (Polino et al., 2018).
- Initializing networks with a discretized pretrained network followed by fine-tuning improved accuracy compared with training a quantized network from random initialization for the same duration, suggesting proximity to a full-precision network enhances low-precision fine-tuning.
- Reducing noise with larger batch size improves accuracy. The results in Table 2 are consistent with the idea that gradient noise limits low-precision training and increasing batch size helps counter this noise.
- Reducing weight decay improves accuracy for 4-bit ResNet-18, as it may lack sufficient capacity to compensate for low-precision weights and activations with the same weight decay as the full precision network. In contrast, for the larger 4-bit networks, best results were obtained with weight decay $10^{-4}$.

## 5 Discussion

We show that low-precision quantization followed by finetuning, when properly compensating for noise, is sufficient to achieve state of the art performance for networks employing 4- and 8-bit weights and activations. This work demonstrates a straightforward, scalable approach for quantization, a critical step towards harnessing the energy-efficiency of low-precision hardware. The results herein provide evidence that 4-bits suffice for classification.

## References

Chaim Baskin, Eli Schwartz, Evgenii Zheltonozhskii, Natan Liss, Raja Giryes, Alexander M. Bronstein, and Avi Mendelson. UNIQ: uniform noise injection for the quantization of neural networks. *CoRR*, abs/1804.10969, 2018. URL `http://arxiv.org/abs/1804.10969`.

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

[4]For ResNet-50, ResNet-152, and Inception-v3, a step learning rate schedule was used with an initial learning rate of 0.02, reduced by a factor of 0.1 after 30, 60, and 90 epochs.

Table 2: **4-bit ResNet-18 sensitivity experiments** Standard parameters on row 1. Subsequent rows show scores for one changed parameter in bold. * to keep number of weight updates approximately same, number of epochs was increased as larger batches result in fewer updates per epoch.

| **Epochs** | **Pre-trained** | **Batch size** | **Learning rate schedule** | **Weight decay** | **Activation calibration** | **Accuracy (% top-1)** | **Change** |
|---|---|---|---|---|---|---|---|
| 110 | Yes | 256 | exp. | 0.00005 | Yes | 69.82 | - |
| **60** | Yes | **400** | exp. | 0.00005 | Yes | 69.40 | -0.22 |
| 110 | **No** | 256 | exp. | 0.00005 | Yes | 69.24 | -0.58 |
| 165* | Yes | **256-2048** | exp. | 0.00005 | Yes | 69.96 | +0.14 |
| 110 | Yes | 256 | **step** | 0.00005 | Yes | 69.90 | +0.08 |
| 110 | Yes | 256 | exp. | **0.0001** | Yes | 69.59 | -0.23 |
| 110 | Yes | 256 | exp. | 0.00005 | **No** | 69.19 | -0.63 |

Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. PACT: parameterized clipping activation for quantized neural networks. *CoRR*, abs/1805.06085, 2018. URL `http://arxiv.org/abs/1805.06085`.

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*, pp. 3123–3131, 2015.

SK Esser, PA Merolla, JV Arthur, AS Cassidy, R Appuswamy, A Andreopoulos, DJ Berg, JL McKinstry, T Melano, DR Barch, et al. Convolutional networks for fast, energy-efficient neuromorphic computing. 2016. *Preprint on ArXiv. http://arxiv. org/abs/1603.08270. Accessed*, 27, 2016.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *arXiv preprint arXiv:1712.05877*, 2017.

Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Youngjun Kwak, Jae-Joon Han, and Changkyu Choi. Joint training of low-precision neural network with quantization interval parameters. *arXiv preprint arXiv:1808.05779*, 2018.

Raghu Meka. Cs289ml: Notes on convergence of gradient descent. https://raghumeka.github.io/CS289ML/gdnotes.pdf, 2017.

Szymon Migacz. Nvidia 8-bit inference with tensorrt. GPU Technology Conference, 2017.

Asit K. Mishra, Eriko Nurvitadhi, Jeffrey J. Cook, and Debbie Marr. WRPN: wide reduced-precision networks. *CoRR*, abs/1709.01134, 2017. URL `http://arxiv.org/abs/1709.01134`.

Daisuke Miyashita, Edward H. Lee, and Boris Murmann. Convolutional neural networks using logarithmic data representation. *CoRR*, abs/1603.01025, 2016. URL `http://arxiv.org/abs/1603.01025`.

Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *CoRR*, abs/1802.05668, 2018. URL `http://arxiv.org/abs/1802.05668`.

Samuel L Smith, Pieter-Jan Kindermans, and Quoc V Le. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017.

Yuhui Xu, Yongzhuang Wang, Aojun Zhou, Weiyao Lin, and Hongkai Xiong. Deep neural network compression with single and multiple level quantization. *CoRR*, abs/1803.03289, 2018. URL `http://arxiv.org/abs/1803.03289`.

Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *CoRR*, abs/1702.03044, 2017. URL `http://arxiv.org/abs/1702.03044`.

Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian Reid. Towards effective low-bitwidth convolutional neural networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.