PRUNE OR QUANTIZE? STRATEGY FOR PARETO-OPTIMALLY LOW-COST AND ACCURATE CNN

Anonymous authors
Paper under double-blind review

ABSTRACT

Pruning and quantization are typical approaches to reduce the computational cost of CNN inference. Although the idea to combine them together seems natural, it is being unexpectedly difficult to figure out the resultant effect of the combination unless measuring the performance on a certain hardware which a user is going to use. This is because the benefits of pruning and quantization strongly depend on the hardware architecture where the model is executed. For example, a CPU-like architecture without any parallelization may fully exploit the reduction of computations by unstructured pruning for speeding up, but a GPU-like massive parallel architecture would not. Besides, there have been emerging proposals of novel hardware architectures such as one supporting variable bit precision quantization. From an engineering viewpoint, optimization for each hardware architecture is useful and important in practice, but this is quite a brute-force approach. Therefore, in this paper, we first propose hardware-agnostic metric to measure the computational cost. And using the metric, we demonstrate that Pareto-optimal performance, where the best accuracy is obtained at a given computational cost, is achieved when a slim model with smaller number of parameters is quantized moderately rather than a fat model with huge number of parameters is quantized to extremely low bit precision such as binary or ternary. Furthermore, we empirically found the possible quantitative relation between the proposed metric and the signal to noise ratio during SGD training, by which the information obtained during SGD training provides the optimal policy of quantization and pruning. We show the Pareto frontier is improved by $4\times$ in post-training quantization scenario based on these findings. These findings are available not only to improve the Pareto frontier for accuracy vs. computational cost, but also give us some new insights on deep neural network.

1 INTRODUCTION

Reducing execution cost of deep learning inference is one of the most active research topics for deploying its super human recognition to embedded IoT devices and robots. Employing memory- and/or computation-efficient components such as separable convolution, which is combination of depth-wise and point-wise convolutions (Iandola et al., 2016; Zoph et al., 2018; Zhang et al., 2018; Howard et al., 2017), structured/unstructured pruning of connections and/or activations, and quantizing activation, weight and/or their vectors (Stock et al., 2019; Jegou et al., 2011; Gong et al., 2014) are typical approaches. Among them, separable convolution and structured pruning are on the same direction, because separable convolution can be viewed as pruned convolutions in handcrafted manner. From pruning viewpoint, since the separable convolution structure is made after applying aggressive pruning to normal convolution, the resultant accuracy drop tends to be large (Stock et al., 2019) while drastically reducing memory and computational cost. On the other hand, structured pruning and quantization are seemingly orthogonal approaches and able to be naturally combined (Tung & Mori, 2018; Han et al., 2016). However, it is still not well-studied how they interact each other. For instance, as an extreme quantization, employing single bit representation has been actively explored. Since non-negligible accuracy drop is inevitable in that extreme quantization, some papers proposed to increase the number of channels to compensate the lack of expressivity (Lin et al., 2017). In other words, quantization approach can reduce the number of bit further by compromising the increase of the number of channels, or the increase of the number of computations.

This indicates that, conversely, the reduction of channels by pruning may limit the capability of quantization. This discussion arises a controversial question *which is better, a fat model with lower bit width or a slim model with larger bit width?* In order to answer this question, we need a metric which fairly measures the both effects of pruning and quantization. One such metric that already appears in literature is the inference speed when the model is executed on a certain specific hardware. Although this metric is practically useful or even ideal when the target hardware is determined in advance, it strongly depends on the feature of the hardware architecture. In Yang et al. (2018), optimal architecture was searched using inference time as objective to optimize and found different architectures depending on target devices. For example, if a hardware can not handle extremely low bit width such as 1 or 2 bits, and handle them as 8 bit integer with upper redundant bits filled with 0, we can not exploit the reduction of bit width to improve the inference speed at all. Figuring out how much we can reduce the computational complexity of deep neural networks is also an important open question from the theoretical viewpoint.

The discussion so far urges us to develop a hardware-agnostic and theoretically reasonable metric for measuring the computational cost of the neural network architecture. In this paper, we propose the Frobenius norm of effective value of weight parameters as one of such metric. This metric is proportional to the total energy when the model is executed on an *ideal* hardware, where the energy consumption for a single MAC computation is proportional to the squared effective amplitude of the individual weight parameter used for the MAC computation. The basic idea of the metric is analogous to highly efficient class-B amplifier circuit whose energy consumption is determined by the instant signal amplitude (Sechi, 1976). This metric successfully reflects the effects of the both quantization and structured/unstructured pruning in accordance with intuition.

By using this proposed metric, we empirically find that slimmer model can achieve far better Pareto frontier in lower computational cost region than fatter model after quantization, while fat model is advantageous to achieve higher accuracy in larger computational cost region. Finally, we execute experiments under a post-training quantization scenario (Banner et al., 2018) on ImageNet dataset (Deng et al., 2009) in order to verify the validity of our claim : "Prune-then-quantize" is superior to "Quantize-only" or "Prune-only" to achieve better Pareto frontier.

Additionally, based on the fact that this metric is relevant to signal to noise ratio (S/N), the metric is measurable during SGD training in which the absolute value of weights and the random walk of weight parameters are corresponding to the signal and the noise, respectively. We observe that the dependencies of the metric on the validation accuracy seem to be correlated between those during training and those applying quantization after the training. From this observation, we point out some possibilities that we could expect the robustness of a model to quantization from the information obtained during training, we could determine the optimal policy for quantization of that model, and we could develop novel optimization or regularization scheme.

Our main contributions are:

- We define the hardware-agnostic metric to measure the computational cost of pruned and quantized models.
- We empirically found that models with fewer parameters achieve far better accuracy in low computational cost region after quantization.
- We show the potential quantitative relation between the quantization noise and the perturbation of weight parameters during SGD training.

And as implications, we hope to exploit our findings for:

- Thorough comparison of various neural network architectures using the proposed hardware-agnostic metric.
- Development of a method to extract quantization policy from the information obtained during SGD training.
- Development of training algorithm or regularization scheme for producing robust model based on the relation between the quantization noise and the perturbation of weight parameters during SGD training.

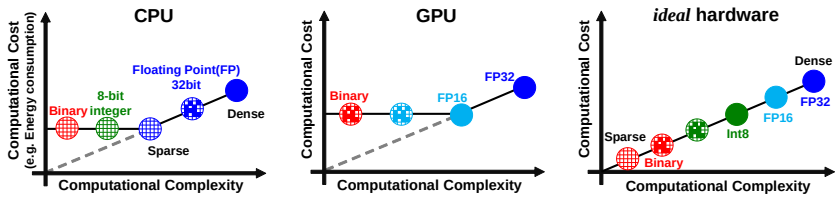

Figure 1: Left/Center: computational cost strongly depends on hardware architecture where the model is executed. Right: proposed computational cost for the purpose of analysis or theoretical research, assuming *ideal* hardware architecture.

2 Effective Signal Norm

We need a metric which properly reflects the effect of both quantization and pruning. Conventionally, the effectiveness of the quantization has been evaluated by the number of bit that is required to achieve certain accuracy or the achieved accuracy by using certain bit numbers for each specific network architecture (Stock et al., 2019). We can not use this to compare the efficiencies between different architecture models (e.g. MobileNet vs. ResNet-18). On the other hand, the number of MAC computations and/or the number of parameters are used to compare different architectures. However, the number of MAC computations basically does not take quantization into account, and the number of parameters is not directly related to the inference time.

Recently, it has been proposed to use actual or estimated inference speed as a metric for comparing different network architectures (Yang et al., 2018; Wang et al., 2019a; Cai et al., 2019). This metric is very useful when the target hardware is determined in advance and, ideal for those who would like to use the model that performs best on that hardware. However, this metric is strongly hardware-dependent. Actually in Yang et al. (2018); Wang et al. (2019a); Cai et al. (2019), they found that the optimal architectures for different types of target hardwares are totally different. From the viewpoint of the interest on e.g. how simple we can make a deep neural network model while achieving required accuracy, hardware-agnostic metric is preferred.

The characteristics that the metric for evaluating a model should have are to correlate with the energy consumption that is consumed when the model is executed on an *ideal* hardware. Here, we assume an *ideal* hardware whose energy consumption monotonically decreases when the bit width is reduced by quantization, and also monotonically decreases when the number of non-zeros in weight parameters is reduced by pruning. For example, a hardware which is composed of 8-bit integer MAC array can not be accelerated further even if the bit width is reduced from 8 bits to 1 or 2 bits. Thus, the energy consumption measured using such hardware does not satisfy the aforementioned requirement, and it can not be our metric. A hardware like CPU, which processes each computation in serial, can naturally exploit the unstructured or structured sparsity of the weight parameters by skipping the computations with zeroed weights. However, since it is difficult to parallelize computations while keeping such strategy, a hardware like GPU, which employs massively parallel MAC units, is hard to benefit from the sparsity in general. A hardware dedicated to sparse convolution (Lu et al., 2019) tends to show better performance only when the sparsity is sufficiently high due to relatively large overhead for encoding and decoding sparse weight parameters in special format.

Therefore, the benefit of sparsity by pruning and low bit width by quantization largely depends on the hardware architectures as long as considering only existing hardware. Because our requirement is hardware-agnostic metric, we assume an *ideal* hardware of which the energy consumption is linearly proportional to the number of non-zero weight parameters and monotonically depends on the bit width of weight parameters as shown in Figure 1, leaving the feasibility of such *ideal* hardware for a while.

2.1 The definition of effective signal norm

We define the metric as follows, and call it as *Effective Signal Norm* (ESN),

$$\mathrm{ESN} = \sum_l ||c^l f(\mathbf{W}^l_{\mathrm{int}})||_F^2, \tag{1}$$

where $\mathbf{W}^l_{\mathrm{int}} = \lfloor \mathbf{W}^l / \Delta^l \rfloor + 0.5$, where $\mathbf{W}^l$ is the weight tensor and Δ^l is the quantization step size of lth layer, c^l is a coefficient depending on layer; if $c^l = 1$, ESN is related to the number of parameters (cf. memory footprint), and if c^l is the number of computations per each parameter at lth layer, ESN is related to the number of computations (cf. FLOP). $f(\cdot)$ is element-wise function, which determines how the metric responses to the value of each weight parameter. In this paper, we propose two functions for $f(\cdot)$. The first one is $f(\mathbf{W}^l_{\mathrm{int}}) = \mathbf{W}^l_{\mathrm{int}}$, which is based on the assumption that the energy consumption increases with the square of the value for each weight parameter or for each computation. When $c^l = 1$, the definition is as below,

$$\mathrm{ESN}_a = \sum_l ||\mathbf{W}^l_{\mathrm{int}}||_F^2. \tag{2}$$

This assumption is reasonable when we employ analog (or in-memory) MAC computation engine (Shafiee et al., 2016; Miyashita et al., 2017), because the energy consumption is proportional to the square of the signal amplitude when the signal is represented in analog quantity such as voltage or current. Assuming *ideal* hardware, we adopt the definition that the energy consumption varies according to the instant amplitude (cf. class-B amplifier), which is more energy efficient than the case where the energy consumption is constant and the value is determined by the maximal amplitude (cf. class-A amplifier) (Sechi, 1976).

The second one is $f(\mathbf{W}^l_{\mathrm{int}}) = \lceil \log_2(\mathrm{abs}(\mathbf{W}^l_{\mathrm{int}})) + 1 \rceil$, where $\log_2(\cdot)$ and $\mathrm{abs}(\cdot)$ functions are applied to each element of a tensor argument. This is based on the assumption that the energy consumption increases with the binary logarithm of the value for each weight parameter. When $c^l = 1$, the definition is as below,

$$\mathrm{ESN}_d = \sum_l ||\lceil \log_2(\mathrm{abs}(\mathbf{W}^l_{\mathrm{int}})) + 1 \rceil||_F^2. \tag{3}$$

In digital circuit, since the number is represented in binary digit (bit), the energy consumption for moving or processing signal is roughly proportional to the number of bit, which is binary logarithm of the value. Then it is reasonable to use equation (3) for digital circuit.

2.2 Relation between ESN and S/N

The effective signal norm defined in equation (2) is connected with signal to noise ratio (S/N) when the noise is dominated by quantization noise and the noise is approximated by uniform distribution (Gray & Neuhoff, 1998).

$$\mathrm{ESN}_a = \sum_l \left(\frac{||\mathbf{W}^l||_F^2}{||\mathbf{W}^l - \Delta \cdot \mathbf{W}^l_{\mathrm{int}}||_F^2} \cdot \frac{||\mathbf{W}^l||_0}{12} \right) = \sum_l \left(S/N_l \cdot \frac{||\mathbf{W}^l||_0}{12} \right), \tag{4}$$

where l is layer index, S/N_l is the signal to quantization noise ratio of lth layer defined by $S/N_l = \frac{||\mathbf{W}^l||_F^2}{||\mathbf{W}^l - \Delta \cdot \mathbf{W}^l_{\mathrm{int}}||_F^2}$, and $||\mathbf{W}^l||_0$ is the number of non-zero elements in the tensor $\mathbf{W}^l$. The derivation of equation (4) is shown in Appendix E. This equation allows us to calculate ESN_a as long as S/N is defined.

2.3 ESN_a during training process

For example, we can define S/N by regarding the perturbation of weight parameters during training as noise. Formally, we define the signal and noise at jth epoch as follows,

$$S_j = \sum_l \sum_i ||\mathbf{W}^l_{j,i}||_F^2, \quad N_j = \sum_l \sum_i ||\mathbf{W}^l_{j,i} - \mathbf{W}^l_{\mathrm{init}_j}||_F^2, \tag{5}$$

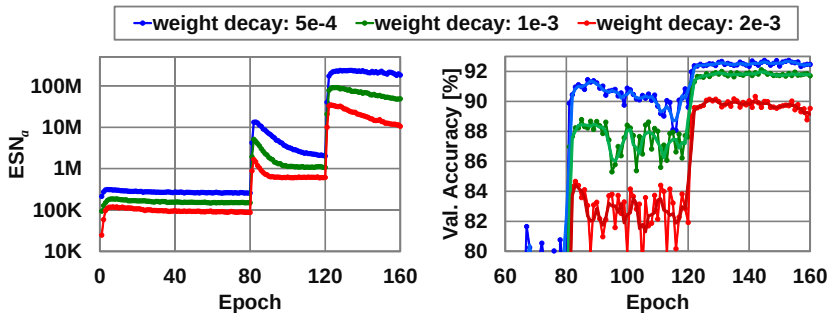

Figure 2: Training curves of ESN_a (left) and validation accuracy (right) on CIFAR-10. In the right graph, moving average curve between 3-epochs is overlapped on each plot.

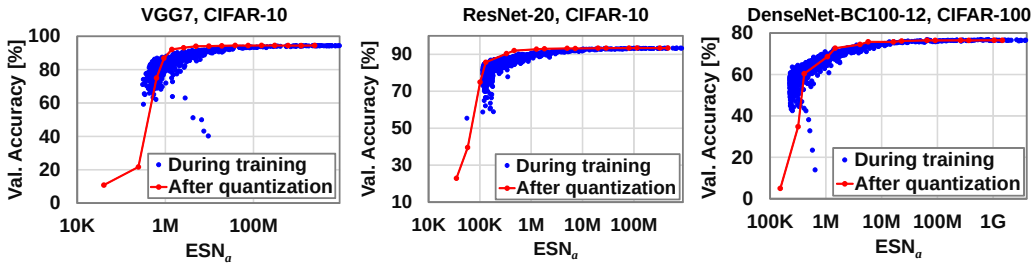

Figure 3: Relation between validation accuracy vs. ESN_a during training and after quantization. ESN_a of the former one is computed using equation (5) and that of the latter one is computed by equation (2)

where $\mathbf{W}^l_{j,i}$ is the weight parameters in lth layer at ith iteration in jth epoch, $\mathbf{W}^l_{\mathrm{init}_j}$ is the snapshot of the weight at the beginning of jth epoch. Then, N_j means the effective value of random walk noise of weight parameters in one epoch.

The left figure in Figure 2 shows the ESN_a curve during training, which is calculated by equation (4) and S/N defined by (5). In this experiment, we use ResNet-20 (He et al., 2016) on CIFAR-10 dataset (Krizhevsky, 2009). We use SGD with momentum of 0.9 and we set the mini-batch size to 100 and the initial learning rate to 0.1, and shift it at 80 and 120 epochs by $1/10$. We vary the weight decay factor from 5×10^{-4} to 2×10^{-3} as shown in different colors in Figure 2.

Interestingly, the ESN_a curves look very similar to the validation accuracy curves shown in the right figure in Figure 2. The both ESN_a and validation accuracy steeply increase at the point where the learning rate is decreased by $1/10$. This is because decreasing learning rate reduces perturbation (or random walk) of weight values during one epoch, and as a result, S/N increases. And we can also see that as the weight decay factor is larger, ESN_a tends to be small. This is also reasonable because weight decay works such that $||\mathbf{W}||_F^2$ or signal level decreases, which leads to reducing S/N. More interestingly, even the trivial tendency looks correlated with the validation accuracy, e.g. after steep rise at 80 epoch, the both ESN_a and validation accuracy decrease gradually as if they converge to stable values after overshooting, and the relation between the amount of overshooting and weight decay factor is also similar, that is, larger overshooting is observed with smaller weight decay factor in this experiment. Although this findings might be useful for developing novel optimization algorithm, we leave it as a future work.

In Figure 3, blue plots show the ESN_a vs. validation accuracy. In this experiment, we use VGG7 (Simonyan & Zisserman, 2014) and ResNet-20 on CIFAR-10, and DenseNet-BC with $l = 100$, $k = 12$ (Huang et al., 2017) on CIFAR-100. We employ SGD with momentum of 0.9 and set the initial learning rate to 0.1 followed by cosine annealing without restart (Ilya Loshchilov, 2017). The mini-batch size and weight decay factor are fixed to 125 and 5×10^{-4}, respectively.

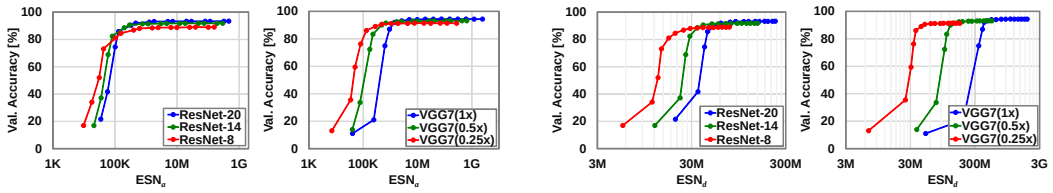

Figure 4: Accuracy vs. ESN_a(left) and ESN_d(right) for various network depth and width.

We also plot in red curve the ESN_a vs. validation accuracy curve when we apply quantization to the weight parameters trained by the training process where the blue plots are obtained. As shown in the figure, we can see that the red curve and blue curve seem to be correlated. This indicates a possibility that the model acquires the robustness to the quantization by having experienced similar perturbation due to random walking during training process. If this is true, ESN_a vs. validation accuracy curve during training is available as a loose boundary of the accuracy degradation due to decrease of ESN_a by quantization. We discuss this hypothesis at section 2.7 again.

2.4 ESN VS. ACCURACY FOR VARIOUS MODEL SIZE

Now that we define our metric, we attempt to use it to investigate which is better, a slim (the number of weight parameters and computations is reduced by pruning) and mildly quantized (e.g. $4 \sim 5$ bits) model, or a fat (the number of weight parameters and computations is large) and radically quantized (e.g. $1 \sim 2$ bits) model. We evaluate 6 network architectures with different depth and width: ResNet-8, 14, 20 and VGG7 with different channel width ($1\times$, $0.5\times$, $0.25\times$) on CIFAR-10 dataset. In this experiment, we employ networks with originally small number of parameters as slim models instead of pruning fat models. The network architecture of ResNet is based on the original one without bottleneck architecture (He et al., 2016). The detailed network architecture of VGG7 is shown in Appendix F.

Figure 4 shows the validation accuracy vs. ESN_a and ESN_d. Here we consider the difference of the ESN_a and ESN_d. For simplicity, we assume the all weight parameters have the same value of v, which is larger than one, and the total number of the weight parameters is N. Then the definitions of ESN_a and ESN_d are expressed as follows,

$$\mathrm{ESN}_a = v^2 \times N, \quad \mathrm{ESN}_d = \lceil \log_2 v + 1 \rceil \times N. \tag{6}$$

Therefore, as we compare ESN_a and ESN_d, the both ESN_a and ESN_d are linearly proportional to the number of weight parameters, whereas values of weight parameters affect to the metric with squared scale and logarithmic scale, respectively. This means that ESN_a is more strongly dependent on how large the values are rather than how many weight parameters there are, and vice versa. In other words, ESN_d is relatively more sensitive to pruning than to quantization, and vice versa. These tendencies are clearly shown in Figure 4. And we can see that, even using ESN_a metric, which is advantageous for quantization, slimmer model shows better validation accuracy in lower ESN region. This experiments indicate that in lower ESN region, or with lower energy consumption on the *ideal* hardware especially employing digital computing, it is advisable strategy to prune the model until the limit where the desired accuracy is achieved, and then to apply quantization, in order to obtain as high accuracy as possible. In contrast, it is bad strategy to apply quantization to a fat model even if the fat model show much better accuracy than the pruned slim model before quantization. Consequently, the answer to the question described in the beginning of this subsection, *which is better, a slim and mildly quantized model, or a fat and radically quantized model?*, is the former one, which may in a sense raise a question on the trend of researches on quantization. Note that the discussion here also indicates that the relative importance of quantization is increased if analog computing become practical in the future.

2.5 TRAINING UNDER LOW ESN CONDITION

We observe how the weight parameters evolve in training process. The loss, ESN_a, and the number of pruned filters (output channels) are shown from left to right in Figure 5. In this experiment, we

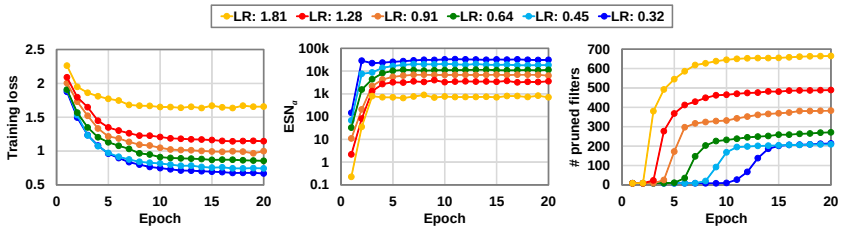

Figure 5: Training curves of loss, ESN_a and the number of pruned filters under different learning rate (LR) conditions. ESN_a is calculated by equation (4) and (5).

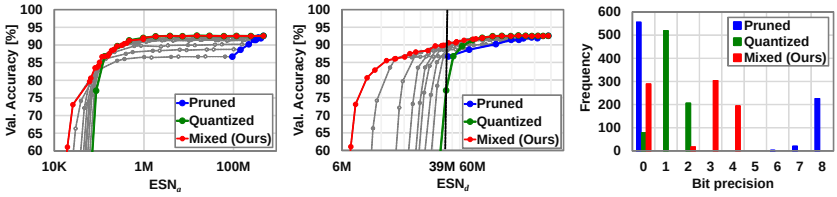

Figure 6: Accuracy vs. $\mathrm{ESN}_{a/d}$ (left/center) and histogram of the maximal bit width for each filters (right) in pruning only, quantization only, and prune-then-quantize mixed cases.

train ResNet-20 on CIFAR-10 dataset with initial weight parameters of a trained model. We employ SGD with momentum of 0.9, weight decay factor of 5×10^{-4} and mini-batch size of 200, and update for 20 epochs. Learning rate (LR) is fixed to a constant value, and the value is varied from 0.32 to 1.81 for each training. Note that in this experiment, we don't intentionally prune filters, but filter pruning spontaneously occurs along with the sequential updates of weight parameters by SGD. The center graph shows as the learning rate increases, ESN_a decreases due to the larger noise. The number of pruned filters after 20 epoch is larger for the larger learning rate, which is corresponding to smaller ESN_a. In a training process, the number of pruned filters steeply increases within a few epochs and then is maintained without noticeable change, rather than continues to increase with constant rate throughout the training process.

This indicates that under small ESN_a condition controlled by the learning rate, the optimizer finds a solution of reducing the number of parameters by pruning filters and giving priority to increase the amplitude of remaining parameters, rather than letting the all parameters survived with small amplitude.

2.6 PRUNE *then* QUANTIZE

In the previous two subsections, we empirically show that in order to achieve optimal accuracy in low ESN region, we need to prepare a slim model before applying quantization. A slim model can be generated by pruning a fat model or by designing a slim architecture. Based on the lottery ticket hypothesis (Frankle & Carbin, 2019), where a fat or overparameterized network can have higher potential to find better optimal parameters by SGD training, we adopt the former one, that is, pruning weight parameters from a pretrained fat model. We apply ADMM regularization for structured pruning (Wang et al., 2019b) since it achieves state-of-the-art performance.

In this experiment, we evaluate ResNet-20 on CIFAR-10 dataset. We take pretrained weight parameters and update them through SGD algorithm with ADMM regularization (Wang et al., 2019b) for a few additional epochs. And then, we fine tune the resultant pruned model for 160 epochs. In the fine tuning, we employ SGD with momentum of 0.9 and we set the mini-batch size to 100 and the initial learning rate to 0.1 followed by cosine annealing without restart, and weight decay factor is set to 5×10^{-4}. Since the weight parameters are not yet quantized in this phase, we update the all parameters, except that we set the weights for the filters (output channels) and channels (input channels) that are pruned at the previous phase to be fixed to zero. Finally, we apply quantization.

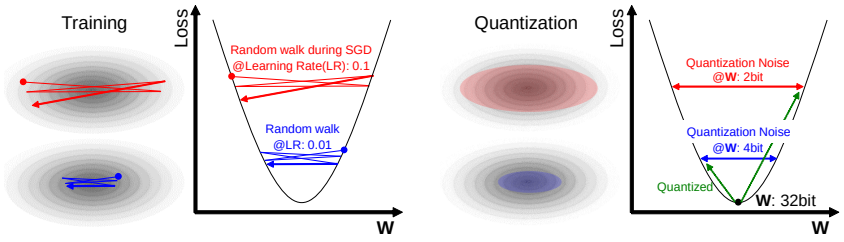

Figure 7: Relation between random walk during SGD and quantization noise. We often see that the loss steeply decreases when LR is shifted by e.g. $1/10$. In this situation, the loss before shifting LR is perhaps governed by the random walk noise due to the large LR as shown in the left figure. On the other hand, when we quantize weight parameters of the trained model, the loss or the accuracy should be determined by the noise induced by quantization as shown in the right figure. If we can assume that the loss landscapes of the two cases should not be diverged so much, the loss (or the accuracy) after quantization is possibly predicted by the dependency of the loss (or the accuracy) on ESN_a during training.

As shown in Figure 6, the Pareto-frontier significantly improves by applying prune-then-quantize (red curve) compared to applying extreme quantization (green curve). The right graph in Figure 6 shows the histogram of the maximal bit width in each filter (output channel) of three models pointed by breaking line in the center graph ($\mathrm{ESN}_d = 39\mathrm{M}(\mathrm{bits})$). Since the pretrained model has 800 filters in total before applying pruning and quantization, the summations of frequencies for each color are equal to 800. In pruning only (blue bars) case, the weight parameters in 555 filters, which is the largest number among three methods, are represented with 0 bits, or pruned. And the most of remaining filters are 8 bits. In quantization only (green bars) case, the number of pruned filters is as small as 77, while the bit width of remaining filters are mostly 1 bit. In our proposed prune-then-quantize mixed (red bars) case, the number of pruned filters is 288 and the bit width of the most of remaining filters are 3 or 4 bits, that is, the remaining filters are mildly quantized. As shown in this case, while the values of ESN_d are similar, the validation accuracies are different depending on the number and the bit width of weight parameters. The prune-then-quantize mixed method produces properly pruned and mildly quantized model, which achieve far better validation accuracy especially in low ESN region compared to the model that is produced by extremely quantizing fat model without pruning.

2.7 ESN_a FOR QUANTIZATION

As we see in section 2.3, there is a possible quantitative relation between ESN_a during training a model and ESN_a of the pruned and quantized version of that model after the training is completed. This finding inspires us to exploit the information of ESN_a obtained during training to determine the quantization policy; e.g. how many bits we should allocate for each layer. The intuition behind the idea is shown in Figure 7. The straightforward idea is to use ESN_as of each layer at a certain epoch during training as targets for quantization. A preliminary experiment for verification of this idea is shown in Appendix B.

3 EXPERIMENT ON IMAGENET

In order to verify the validity of our claims, we execute experiments under a post-training quantization scenario (Banner et al., 2018) on ImageNet dataset (Deng et al., 2009). The idea of post-training quantization assumes a use-case scenario where a user (e.g. at edge side) quantizes a given pretrained CNN model by him/herself with limited number of unlabeled training samples for obtaining lightweight model fitting to their own situation. As concluded in 2.4, in low ESN region or with low energy consumption, we can achieve higher accuracy when we quantize a slim model than when we quantize a fat model even if the latter is more accurate. In order to provide optimal accuracy in the low computational cost region, we apply the proposed prune-then-quantize method to the post-training quantization scenario.

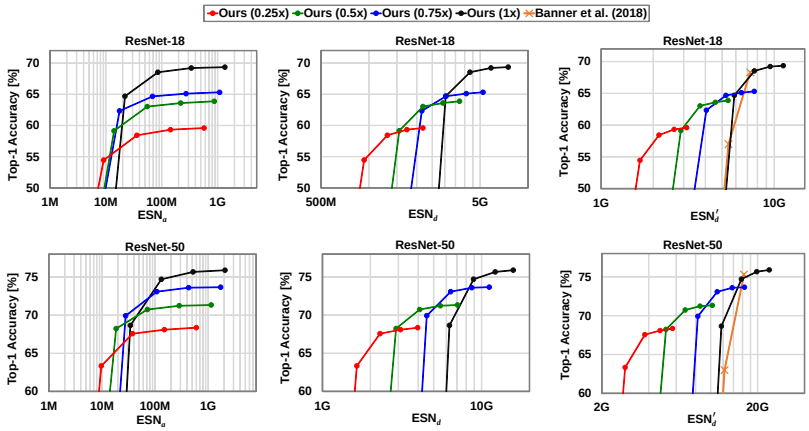

Figure 8: Top-1 accuracy on ImageNet vs. $\mathrm{ESN}_{a/d}$ (left/center) and estimated ESN'_d (right), applying prune-then-quantize method to ResNet-18/50 (upper/lower).

Figure 8 shows the result when our method is applied to ResNet-18 and ResNet-50 on ImageNet dataset. We use pretrained models[1][2] as basic models with $1\times$-channel width and prune the models to various numbers of channel width ($0.25\times$, $0.5\times$ and $0.75\times$). In this experiment, we determine which filters (output channels) to be pruned, based on the signal norm of weights calculated by equation (5). We prune filters from the smallest signal norm in order, and tune each model for the pruned structure through SGD algorithm for 4 epochs. The mini-batch size is fixed to 64 and the learning rate is initialized to 0.1 and divided by 10 at 2 and 3 epochs. And then, we quantize each model. For quantization, we only fine tune statistic parameters in BatchNorm layers (called `running_mean` and `running_var` in pytorch (Paszke et al., 2017)) without any labeled data (Sasaki et al., 2019), which is same condition as conventional work (Banner et al., 2018).

In addition to evaluations by ESN_a and ESN_d, in order to compare our method with the conventional work, we estimate its computational cost using equation (1) with c^l and $f(\cdot)$ properly defined,

$$\mathrm{ESN}'_d = \sum_l \sum_m \mathrm{MAC}_{l,m} \times \lceil \log_2(\max(\mathrm{abs}(\mathbf{W}^{l,m}_{\mathrm{int}}))) + 1 \rceil, \tag{7}$$

where $\mathrm{MAC}_{l,m}$ is the number of computation per each filter and $\mathbf{W}^{l,m}_{\mathrm{int}}$ is the mth filter of $\mathbf{W}^{l}_{\mathrm{int}}$. By applying $\max(\cdot)$ function, we restrict the number of bit to be same in each filter.

This metric[3] is applied to the recently proposed efficient MAC array architecture with capability of handling variable bit precision (Maki et al., 2018).

As shown in Figure 8, the Pareto-frontier is significantly improved by using slim models after pruned some filters compared to Banner et al. (2018). For example, in ResNet-18, 57 % top-1 accuracy is achieved at $2\times$ lower computational cost, and in ResNet-50, 63 % top-1 accuracy is at $4\times$ lower computational cost. These results suggest that we adopt slim model in order to achieve best accuracy in low computational cost region, instead of quantizing weight parameters to extremely low bit precision.

4 Conclusion

In this paper, we proposed the hardware-agnostic metric called effective signal norm (ESN) to measure the computational cost. Using the metric, we demonstrated that a slim model with fewer weight parameters achieves a Pareto frontier performance in low computational cost region rather than a

[1] https://download.pytorch.org/models/resnet18-5c106cde.pth
[2] https://download.pytorch.org/models/resnet50-19c8e357.pth
[3] In Maki et al. (2018), the metric is defined as "MAC×bit", which is essentially same as ESN'_d.

extremely quantized fat model. Furthermore, we showed potential quantitative relation between the weight perturbation during SGD training and the quantization noise or the robustness against quantization.

By defining the metric, we can target to realize a hardware whose energy consumption is proportional to the metric from hardware architecture side, as well as we can target to reduce the metric from algorithmic side. We expect to accelerate progress of researches on both algorithm and hardware architecture by sharing the consensus about the metric for computational cost.

5 Related works

Quantization Courbariaux & Bengio (2016); Rastegari et al. (2016); Zhu et al. (2017) quantize weight parameters and activations into 1 or 2. Since these extreme quantization deteriorates accuracy to non-negligible extent, quantization methods with e.g. $4 \sim 8$ bits are also actively explored (Banner et al., 2018; Lin et al., 2017) to avoid the accuracy drop. Miyashita et al. (2016) attempts to quantize weight parameters and activations in logarithmic domain aiming at not only reducing information loss but also replacing multiplication with bit-shift to simplify computation. To reduce *average* bit width as much as possible while maintaining accuracy, and exploit it to accelerate inference speed, Maki et al. (2018) proposes variable bit width quantization co-optimized with hardware architecture. Although non-uniform quantization (Tung & Mori, 2018; Han et al., 2016) and vector or product quantization (Gong et al., 2014; Jegou et al., 2011; Stock et al., 2019) are also actively studied, these approaches are effective for reducing memory footprint but not for reducing computational cost directly. As use cases, there are quantization-aware training (Courbariaux & Bengio, 2016; Rastegari et al., 2016; Zhu et al., 2017; Lin et al., 2017; Zhang et al., 2018) and post-training quantization (Banner et al., 2018). The quantization-aware training achieves better accuracy with lower bit width, and is almost inevitable for those with extreme quantization. However, it tends to require more training cost in most cases. The post-training quantization scenario, which is executable with less computational resources and training dataset, has potentially more applications, and then, we also target this scenario.

Pruning There are structured pruning, which prune whole layers, filters, or channels to keep the regular structure so that the computations are easily parallelized, and unstructured pruning, which randomly prunes individual weight parameter. Since it is difficult to benefit from the unstructured pruning unless the sparsity is sufficiently large due to its incompatibility to parallelization (Lu et al., 2019), our target is structured pruning. A lot of pruning method are also proposed such as criteria based approaches (LeCun et al., 1990), regularization based approaches (Han et al., 2015; Wang et al., 2019b), and method employing reinforcement learning (Zhong et al., 2018). In this paper, we argue that a pruned slim model performs better after quantization in lower computational cost region. To produce pruned slim model, any pruning method can be applied.

Metric for measuring cost In most quantization papers, the standard metric is the achieved accuracy under a certain bit width. For pruning, the standard metric is the number of parameters and/or the number of computations (FLOP). Some papers use inference time when the model runs on a certain hardware (Cai et al., 2019; Yang et al., 2018), but this metric strongly depends on the hardware. In this paper, we propose to use hardware-agnostic metric.

Noise during training The relation between learning rate and batch size and noise during training is a lot discussed (Keskar et al., 2017; Xing et al., 2018). To make the model more robust to quantization, some papers propose to intentionally add noise to gradient or weight parameters (Spallanzani et al., 2019; Baskin et al., 2018a;b).

References

Ron Banner, Yury Nahshan, Elad Hoffer, and Daniel Soudry. ACIQ: analytical clipping for integer quantization of neural networks. *CoRR*, abs/1810.05723, 2018. URL `http://arxiv.org/abs/1810.05723`.

Chaim Baskin, Natan Liss, Yoav Chai, Evgenii Zheltonozhskii, Eli Schwartz, Raja Giryes, Avi Mendelson, and Alexander M. Bronstein. NICE: noise injection and clamping estimation for

neural network quantization. *CoRR*, abs/1810.00162, 2018a. URL http://arxiv.org/abs/1810.00162.

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, 2018b. URL http://arxiv.org/abs/1804.10969.

Han Cai, Chuang Gan, and Song Han. Once for all: Train one network and specialize it for efficient deployment. *CoRR*, abs/1908.09791, 2019. URL http://arxiv.org/abs/1908.09791.

Matthieu Courbariaux and Yoshua Bengio. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. *CoRR*, abs/1602.02830, 2016. URL http://arxiv.org/abs/1602.02830.

J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, June 2009. doi: 10.1109/CVPR.2009.5206848.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

A. Gersho. Quantization. *IEEE Communications Society Magazine*, 15(5):16–16, Sep. 1977. doi: 10.1109/MCOM.1977.1089500.

Yunchao Gong, Liu Liu, Ming Yang, and Lubomir D. Bourdev. Compressing deep convolutional networks using vector quantization. *CoRR*, abs/1412.6115, 2014. URL http://arxiv.org/abs/1412.6115.

R. M. Gray and D. L. Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44(6): 2325–2383, Oct 1998. doi: 10.1109/18.720541.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural network. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 28*, pp. 1135–1143. Curran Associates, Inc., 2015.

Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations (ICLR)*, 2016.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016. doi: 10.1109/CVPR.2016.90.

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. URL http://arxiv.org/abs/1704.04861.

G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, July 2017. doi: 10.1109/CVPR.2017.243.

Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and $<$1mb model size. *CoRR*, abs/1602.07360, 2016. URL http://arxiv.org/abs/1602.07360.

Frank Hutter Ilya Loshchilov. SGDR: stochastic gradient descent with warm restarts. *International Conference on Learning Representations (ICLR)*, 2017.

H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, Jan 2011. ISSN 0162-8828. doi: 10.1109/TPAMI.2010.57.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *International Conference on Learning Representations (ICLR)*, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In D. S. Touretzky (ed.), *Advances in Neural Information Processing Systems 2*, pp. 598–605. Morgan-Kaufmann, 1990. URL http://papers.nips.cc/paper/250-optimal-brain-damage.pdf.

Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 345–353. Curran Associates, Inc., 2017.

L. Lu, J. Xie, R. Huang, J. Zhang, W. Lin, and Y. Liang. An efficient hardware accelerator for sparse convolutional neural networks on fpgas. In *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 17–25, April 2019. doi: 10.1109/FCCM.2019.00013.

A. Maki, D. Miyashita, K. Nakata, F. Tachibana, T. Suzuki, and J. Deguchi. Fpga-based cnn processor with filter-wise-optimized bit precision. In *2018 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, pp. 47–50, Nov 2018. doi: 10.1109/ASSCC.2018.8579342.

D. Miyashita, S. Kousai, T. Suzuki, and J. Deguchi. A neuromorphic chip optimized for deep learning and cmos technology with time-domain analog and digital mixed-signal processing. *IEEE Journal of Solid-State Circuits*, 52(10):2679–2689, Oct 2017. ISSN 0018-9200. doi: 10.1109/JSSC.2017.2712626.

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.

Travis Oliphant. NumPy: A guide to NumPy. USA: Trelgol Publishing, 2006. URL http://www.numpy.org/.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, 2016.

S. Sasaki, A. Maki, D. Miyashita, and J. Deguchi. Post training weight compression with distribution-based filter-wise quantization step. In *2019 IEEE Symposium in Low-Power and High-Speed Chips (COOL CHIPS)*, pp. 1–3, April 2019. doi: 10.1109/CoolChips.2019.8721356.

F. N. Sechi. Linearised class-b transistor amplifiers. *IEEE Journal of Solid-State Circuits*, 11(2): 264–270, April 1976. ISSN 0018-9200. doi: 10.1109/JSSC.1976.1050713.

A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar. Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pp. 14–26, June 2016. doi: 10.1109/ISCA.2016.12.

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

Matteo Spallanzani, Lukas Cavigelli, Gian Paolo Leonardi, Marko Bertogna, and Luca Benini. Additive noise annealing and approximation properties of quantized neural networks. *CoRR*, abs/1905.10452, 2019. URL http://arxiv.org/abs/1905.10452.

Pierre Stock, Armand Joulin, Rémi Gribonval, Benjamin Graham, and Hervé Jégou. And the bit goes down: Revisiting the quantization of neural networks. *CoRR*, abs/1907.05686, 2019. URL http://arxiv.org/abs/1907.05686.

F. Tung and G. Mori. Clip-q: Deep network compression learning by in-parallel pruning-quantization. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7873–7882, June 2018. doi: 10.1109/CVPR.2018.00821.

Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019a.

Yanzhi Wang, Shaokai Ye, Zhezhi He, Xiaolong Ma, Linfeng Zhang, Sheng Lin, Geng Yuan, Sia Huat Tan, Zhengang Li, Deliang Fan, Xuehai Qian, Xue Lin, and Kaisheng Ma. Non-structured DNN weight pruning considered harmful. *CoRR*, abs/1907.02124, 2019b. URL http://arxiv.org/abs/1907.02124.

Chen Xing, Devansh Arpit, Christos Tsirigotis, and Yoshua Bengio. A walk with sgd. *CoRR*, abs/1802.08770, 2018. URL http://arxiv.org/abs/1802.08770.

Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss (eds.), *Computer Vision – ECCV 2018*, pp. 289–304, Cham, 2018. Springer International Publishing. ISBN 978-3-030-01249-6.

Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *European Conference on Computer Vision (ECCV)*, 2018.

X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6848–6856, June 2018. doi: 10.1109/CVPR.2018.00716.

Jing Zhong, Guiguang Ding, Yuchen Guo, Jungong Han, and Bin Wang. Where to prune: Using lstm to guide end-to-end pruning. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 3205–3211. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi: 10.24963/ijcai.2018/445. URL https://doi.org/10.24963/ijcai.2018/445.

Chenzhuo Zhu, Song Han, Huizi Mao, and William J. Dally. Trained ternary quantization. *International Conference on Learning Representations (ICLR)*, 2017.

B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8697–8710, June 2018. doi: 10.1109/CVPR.2018.00907.

A DETAIL OF QUANTIZATION PROCESS

We basically apply mid-rise type quantization (Gersho, 1977), but modify it as follows. In quantization phase, we prune a filter if the all weight parameters in that filter are within $\pm\Delta/2$ as described in the following pseudo code with Numpy (Oliphant, 2006) notation.

```
# Quantize
# W: (filter(out_ch), channel(in_ch), ky, kx)
pruned = numpy.all(abs(W) < delta/2, axis=(1,2,3))
W_int = numpy.floor(W/delta) + 0.5
W_int[pruned, :, :, :] = 0
```

B ESN_a FOR OPTIMAL POLICY OF QUANTIZATION

We attempt to exploit the ESN_a obtained during training to determine the quantization policy using VGG7 where the numbers of filters of all convolutional layers are reduced to $0.25\times$ on CIFAR-10. The left figure in Figure 9 show the validation accuracy vs. ESN_a during training and after quantization. As an initial policy, we apply the same bit width for the all layers. The result shown in gray does not fit the blue plot during training, and this may indicate the policy is suboptimal. Then we investigate the ESN_a of each layer, and we find that the misfit is caused by 6th layer as shown in the right figure. When we see the validation accuracy vs. 6th layer ESN_a shown in green symbols, the validation accuracy after quantization deteriorates at much higher ESN_a than during training. It can be interpreted that since this deterioration is caused by other layers (e.g. 1st layer) than 6th layer, the weight parameters in 6th layer should be quantized more aggressively. Based on this observation, we modify the quantization policy such that the bit width of 6th layer become smaller than the other layer, and it results in improving performance as shown in red plot in the left figure. This preliminary experimental result shows the possibility that we can use ESN_a information during training for finding optimal quantization policy.

C RESNET-18 VS. RESNET-50

The proposed metric allows us to compare different network architectures. In Figure 10, we compare the Pareto frontier of validation accuracy vs. $\mathrm{ESN}_{a/d}$ curves for ResNet-18 and ResNet-50. The plots in Figure 10 are extracted from Figure 8. This result reveals that ResNet-50 has better Pareto-frontier than ResNet-18. One convincing reason is that ResNet-50 employs the parameter-efficient bottleneck structure whereas ResNet-18 does not. On the other hand, in $\mathrm{ESN}_d < 2\mathrm{G}$, ResNet-18 shows better performance partly due to the higher sensitivity of ESN_d to the number of parameters. Therefore, this suggests us to use ResNet-18 if we employ digital computing in this ESN_d region. Such comparisons enabled by the proposed metric are useful for searching or developing efficient structure.

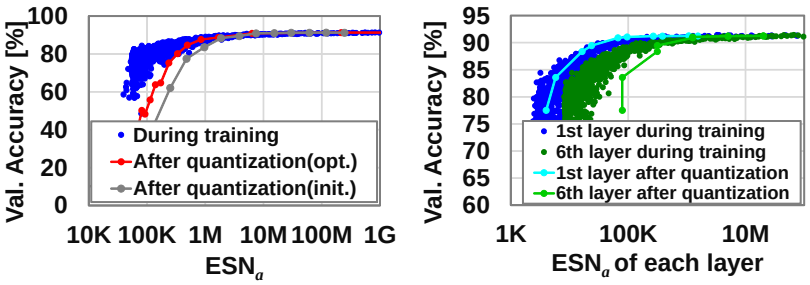

Figure 9: Relation between validation accuracy vs. ESN_a during training and after quantization.

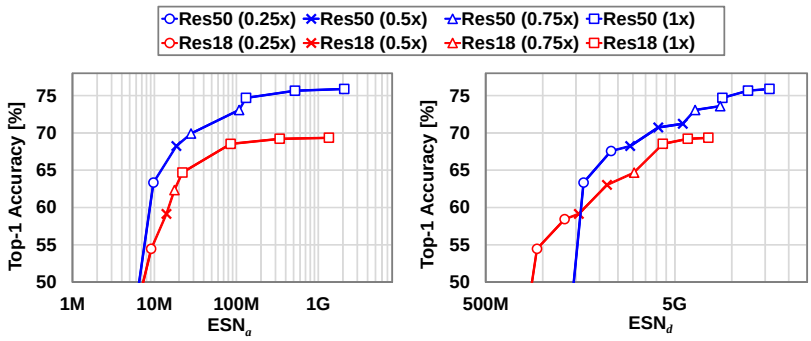

Figure 10: Comparison of ResNet-18 and ResNet-50 with respect to Top-1 accuracy on ImageNet vs. $\mathrm{ESN}_{a/d}$ (left/right).

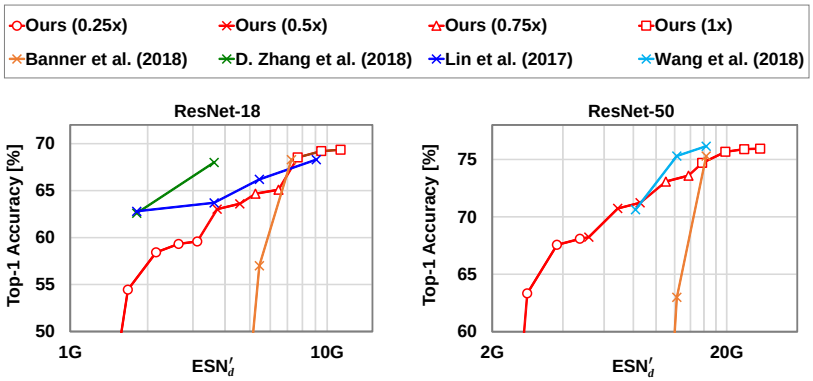

Figure 11: Comparison to other methods with respect to Top-1 accuracy on ImageNet vs. estimated ESN'_d.

D COMPARISON TO OTHER METHODS

In Figure 11, we overwrite some results from other works to Figure 8. Lin et al. (2017) and Zhang et al. (2018) show better accuracy in small ESN'_d region than ours, but they apply quantization-aware training, whereas our result is obtained by post-training quantization. Although we expect that our findings presented in this paper will improve the performance of quantization-aware training as well, we leave it as future works.

E DERIVATION OF EQUATION (4)

The mean squared noise energy N due to quantization over the all elements in a tensor $\mathbf{W}$ is computed by,

$$N = ||\mathbf{W} - \Delta \cdot \mathbf{W}_{\mathrm{int}}||_F^2 / ||\mathbf{W}||_0, \tag{8}$$

where $\mathbf{W}_{\mathrm{int}} = \lfloor \mathbf{W}/\Delta \rfloor + 0.5$, Δ is quantization step size, and $||\mathbf{W}||_0$ is the number of non-zero elements in $\mathbf{W}$. When quantization noise is approximated by uniform distribution, the mean squared noise energy is (Gray & Neuhoff, 1998),

$$N = \int_{-\Delta/2}^{\Delta/2} \frac{1}{\Delta} n^2 dn = \frac{\Delta^2}{12}. \tag{9}$$

By equation (8) and equation (9),

$$\Delta^2 = ||\mathbf{W} - \Delta \cdot \mathbf{W}_{\text{int}}||_F^2 \times \frac{12}{||\mathbf{W}||_0}. \tag{10}$$

Then,

$$||\mathbf{W}_{\text{int}}||_F^2 \sim ||\mathbf{W}/\Delta||_F^2 = \frac{||\mathbf{W}||_F^2}{\Delta^2} = \frac{||\mathbf{W}||_F^2}{||\mathbf{W} - \Delta \cdot \mathbf{W}_{\text{int}}||_F^2} \cdot \frac{||\mathbf{W}||_0}{12}. \tag{11}$$

Here we use the assumption that quantization noise is uniformly distributed again. In this equation, since $||\mathbf{W}||$ and $||\mathbf{W} - \Delta \cdot \mathbf{W}||_F^2$ are signal and noise norm, respectively, the term of $\frac{||\mathbf{W}||_F^2}{||\mathbf{W} - \Delta \cdot \mathbf{W}_{\text{int}}||_F^2}$ is considered to be S/N, thus,

$$||\mathbf{W}_{\text{int}}||_F^2 = \frac{||\mathbf{W}||_F^2}{||\mathbf{W} - \Delta \cdot \mathbf{W}_{\text{int}}||_F^2} \cdot \frac{||\mathbf{W}||_0}{12} = S/N \cdot \frac{||\mathbf{W}||_0}{12}. \tag{12}$$

By summing up this value over the all layers, we obtain equation (4).

F DETAILS OF NETWORK ARCHITECTURES

The network architecures of VGG7 ($1\times$, $0.5\times$, $0.25\times$) which we use in section 2.4 are shown in table 1. These architectures are based on the original model of VGG (Simonyan & Zisserman, 2014).

Table 1: Detailed network architecture of VGG7 for CIFAR-10. C_I and C_O represent the number of input and output channels, respectively in conv-layer. In fc-layer, they represent the number of input and output neurons

Layer name	VGG ($1\times$)		VGG ($0.5\times$)		VGG ($0.25\times$)		kernel size	padding	stride
	C_I	C_O	C_I	C_O	C_I	C_O			
conv1	3	128	3	64	3	32	3×3	1	1
conv2	128	128	64	64	32	32	3×3	1	1
maxpool	128	128	64	64	32	32	2×2	0	2
conv3	256	256	128	128	64	64	3×3	1	1
conv4	256	256	128	128	64	64	3×3	1	1
maxpool	256	256	128	128	64	64	2×2	0	2
conv5	512	512	256	256	128	128	3×3	1	1
conv6	512	512	256	256	128	128	3×3	1	1
maxpool	512	512	256	256	128	128	2×2	0	2
fc	8192	10	4096	10	2048	10	-	-	-