Dynamic Activation Step Size for Post-Training Quantization

Yuanpei Chen[1*], Yuanyuan Ou[2*], YingLei Wang[1], Rongli Zhao[1], Xiaode Liu[1], and Yufei Guo[1⊠]

[1] X Lab, The Second Academy of China Aerospace Science and Industry Corporation, Beijing 100854, China
[2] Chongqing University, Chongqing, China
rop477@163.com, 20161874@cqu.edu.cn, yfguo@pku.edu.cn

Abstract

Post-training quantization (PTQ) has been an efficient approach to quantize the off-the-shelf model. However, without any end-to-end finetuning, the quantized model drops much accuracy as bit-width goes down, especially for the activation quantization. Our preliminary experiments suggest that the activation distribution changes with different input data points. To close the large gap produced by activation quantization, we propose dynamic activation step size for PTQ. Specifically, we assign a router function that depends on the incoming activation and generates a suitable step size for activation quantization. Moreover, we come up with a two-stage tuning algorithm for learning the router function. Our method can achieve good results without the help of the labeled full training dataset. We test our methods on the CIFAR10, CIFAR100, and ImageNet datasets, empirical results demonstrate that our method can recover most accuracy drop of activation quantization.

Introduction

Deep neural networks (DNNs) have achieved great success in many tasks including image classification (Krizhevsky, Sutskever, and Hinton 2012; He et al. 2016), object detection (Ren et al. 2017), object segmentation (Ronneberger, Fischer, and Brox 2015), object tracking (Bewley et al. 2016), etc. However, for the requirement of high accuracy, DNNs usually suffer from massive parameters and high computational complexity, which limits their deployment on mobile devices. To address this problem, several model compression methods have been proposed (Zhang, He, and Jian 2017; Molchanov et al. 2020; Polino, Pascanu, and Alistarh 2018; Zhang et al. 2017; Nagel et al. 2019). Among these methods, the network quantization technique draws more and more attention since it can largely lessen the network storage and meanwhile accelerate the inference speed by reducing the bitwidth of the network parameters and activations.

Quantization will cause severe performance degradation. To narrow the accuracy gap between the original model and the quantized model, most popular works resort to retraining the quantized model. However, in most actual scenarios, training data are not always available for fine-tuning considering the privacy problem. Besides, retraining will inevitably increase the development time of models and delay the production cycle in the industry. In the paper, we focus on quantizing the neural networks without retraining, which is called Post-training Quantization (PTQ).

In practice, activation step size contributes greatly to the quantization loss. Therefore, to alleviate the performance degradation, it is better to find an appropriate activation step size. However, we find that the activation distributions for different input samples differ greatly. Hence, it is unrealistic to resort to a fixed value to accomplish the target of finding an appropriate activation step size. To this end, we propose dynamic activation step size for PTQ. The method can change the activation step size dynamically according to incoming activation and reduce the quantization by a learning way.

In particular, we make the following contributions:

- we propose a router function that takes the incoming activation as input to predict the optimal step size for activation quantization. Furthermore, an elaborate designed two-stage algorithm is also introduced to learn the router function.
- Our method needs not the usage of data labels, which make it enjoy a wide range of application scenarios.
- We evaluate the proposed method on the CIFAR10, CIFAR100, and the large scale ImageNet datasets for comprehensive comparison with state-of-the-art quantization neural networks in image classification. Extensive experimental results show that our method can recover the most accuracy drop caused by activation quantization and performs remarkably.

Preliminary

Quantization

Quantization of neural network weights and activation is a 32-bit floating-point to low-bit integers mapping process. In this study, we study uniform, per-layer, and symmetric quantization (see the classification of quantization in (Li et al. 2021d; Li, Dong, and Wang 2019; Shen et al. 2021; Li et al. 2021e)). In uniform symmetric quantization, the mapped fixed-point integers have the same interval and are symmetrically distributed. The quantization function can be written

[*]Equal Contributions. [⊠]Corresponding author.

by:

$$q(\mathbf{w}) = s \times \text{clip}\left(\lfloor \frac{\mathbf{w}}{s} \rceil, N_{min}, N_{max}\right) \quad (1)$$

Here, s is the step size between two grids and N_{min}, N_{max} are the minimum and maximum integer range, determined by the bit-width b. For activation quantization, we have $N_{min} = 0, N_{max} = 2^b - 1$, while for weights quantization we have $N_{min} = -2^{b-1}, N_{max} = 2^{b-1} - 1$. Quantization function is generally designed to minimize the quantization error (Rastegari et al. 2016; Cai et al. 2017):

$$\min ||\hat{\mathbf{w}} - \mathbf{w}||_F^2. \text{ s.t. } \hat{\mathbf{w}} = q(\mathbf{w}). \quad (2)$$

Generally, In PTQ one can directly get the quantized by solving this minimization problem (for activation quantization, a small subset of the training dataset called calibration dataset is adopted for obtaining the statistics of the activation distribution). However, this method is not optimal in extremely low-bit cases. A naive rounding-to-nearest cannot even handle the 4-bit quantization. For example, in data-free quantization (Nagel et al. 2019), the 4-bit ResNet-18 only has ∼40% accuracy on ImageNet dataset.

Attention in CNN

Attention is a technique that mimics cognitive attention (Wikipedia 2021). The effect enhances the important parts of the input data and fades out the rest—the thought being that the network should devote more computing power to that small but important part of the data. Attention is originally developed in (Vaswani et al. 2017) in the natural language processing area. Nowadays, the attention mechanism is also adopted in the vision model, especially the convolutional network. Squeeze and excitation block (Hu, Shen, and Sun 2018), as a representative work of using attention in quantization neural network, dynamically decide the importance of each channel and assign a scaling factor to them. As a result, the important channels are highlighted. Our work is inspired by this attention mechanism. Specifically, for activation quantization in the ℓ-th layer, the MSE minimization goal is formulated by

$$\min \mathbb{E}_{\mathbf{x}^0 \in \mathcal{S}} ||\hat{\mathbf{x}}^\ell - \mathbf{x}^\ell||_F^2, \quad (3)$$

where $\mathbf{x}^0$ is the input sample to the network and sampled from the limited training dataset $\mathcal{S}$. We can see that this objective is an expectation minimization problem, which means the optimal step size s must find an intermediate solution that is good for every sample. However, an alternative way is to find a dynamic step that varies along with the input sample. In this work, we explore such a possibility and improve the model quantization performance without bells and whistles.

Methodology

We first present some empirical observations to illustrate our motivation. First, we train a ResNet-20 (He et al. 2016) on the CIFAR10 dataset (Krizhevsky, Nair, and Hinton). And observe the activation distribution with different input samples. In table 1, we randomly select 6 samples in the validation dataset and record their maximum activation, 99.9 percentile activation, 99 percentile activation and 90 percentile activation. It can be seen that the activation distribution differs among input samples. In particular, the outlier, *i.e.*, the high activation has a very different value. $\mathbf{x}_3$ has 1.03 max activation but $\mathbf{x}_6^0$ only has 0.74 max activation. This difference shows that a single and fixed step size is not good for the model quantization.

Samples	Max	99.9%	99%	90%
$\mathbf{x}_1^0$	0.8572	0.5550	0.3423	0.1401
$\mathbf{x}_2^0$	0.7998	0.5061	0.3150	0.1218
$\mathbf{x}_3^0$	1.0333	0.4707	0.2848	0.1366
$\mathbf{x}_4^0$	0.8739	0.5486	0.3211	0.1380
$\mathbf{x}_5^0$	0.9887	0.6600	0.3914	0.1672
$\mathbf{x}_6^0$	0.7403	0.4534	0.2400	0.1169

Table 1: Activation distribution sample in the 5-th layer of ResNet-20, given 6 random input samples.

m	1	3	5	10	20
Acc.	79.6%	82.5%	85.5%	86.6%	87.5%

Table 2: Accuracy comparison of 4-bit quantized ResNet-20, given different number of step size candidates.

To verify this observation, we conduct a simple experiment. We use the same ResNet-20 trained on the CIFAR10 dataset which has 93.2% accuracy and quantizes it to 4-bit. With 4-bit weight only quantization, the model only has 87.5% accuracy. Furthermore, we will quantize its activation accuracy, with traditional MSE minimized step size on the calibration set (i.e. the 3), the accuracy drops to 84.1%. Now, we would like to test multiple-step size. First, we get the maximum activation value m, and then split c values equally between $[0.2m, m]$. For example, say we have $c = 1.0$ and $m = 5$, then we get 4 step size from $\{0.2, 0.4, 0.6, 0.8, 1.0\}$. Second, we verify all these options, and we mark i-th test sample as correctly classified as long as one of these options has the right classification. To be more specific, we record the accuracy of the union of all step size trials. In table 2, we summarize the results. Note that here $c = 1$ means the step size is calculated by maximum activation value, therefore its accuracy is lower than the MSE minimized step size. We can see that, as the number of step sizes increases, meaning that a larger range is covered, the accuracy gradually increases. With only 5 candidates, the accuracy surpasses the MSE minimized step size. And continuing to increase the step size, the accuracy will approach the accuracy of the weight-only quantized model. These experiments verify that, by choosing the right step size for each test sample, we can eliminate the accuracy drop caused by activation quantization.

We propose the dynamic step size for post-training quantization. Specifically, we design a router function to predict the optimal step size for each input sample. The schematic view is shown in the Fig. 1. In each layer, we add a router function $\sigma(\cdot)$ which determines the optimal step size by outputting a scaling factor $\sigma(\mathbf{x}) \in (0, 1)$. This scaling factor is then multiplied to the maximum step size $s_{\max}$ to gen-

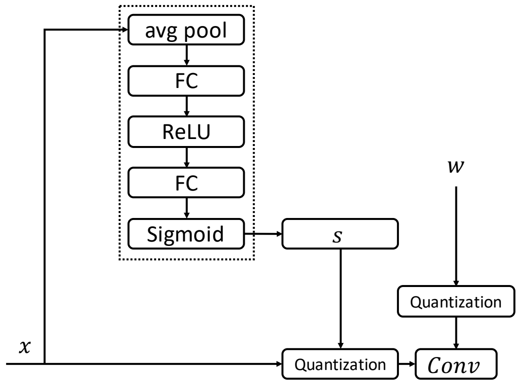

Figure 1: Method of dynamic step size for activation quantization, where each sample will generate a unique step size.

erate the step size for this layer. The router function is designed as a global average pooling layer followed by two fully-connected layers and one rectified linear unit, which is similar to the squeeze-and-excitation channel attention in CNN. Note that we do not use per-channel activation quantization since it is inefficient in the hardware. Formally, our activation can be written as:

$$\hat{\mathbf{x}}^\ell = s_{\max}\sigma(\mathbf{x}^\ell) \times \mathrm{clip}\left(\lfloor \frac{\mathbf{x}^\ell}{s_{\max}\sigma(\mathbf{x})}\rceil, N_{\min}, N_{\min}\right). \quad (4)$$

Ideally, this router function can help us find the best step size for each sample. However, how to learn this router function still remains unsettled. In this paper, we propose a two-stage algorithm for learning the router function. The first stage is the per-layer reconstruction. Given a calibration dataset $\mathcal{C}$, we traverse each layer in the model and minimize the output between original the full-precision network and the quantized network, given by

$$\min_{\theta, s_w} ||\hat{\mathbf{y}}^\ell - \mathbf{y}^\ell||_F^2, \quad (5)$$

where $\mathbf{y}^\ell$ is the full-precision output at ℓ-th layer, and the $\hat{\mathbf{y}}$ is the output from the convolution between quantized weights and quantized activation. θ is the parameters in the router function block and s_w is the step size for weight quantization. Moreover, it can combine other PTQ algorithms like bias correction (Finkelstein, Almog, and Grobman 2019). The second stage is a global tuning algorithm. Using the data from the calibration dataset, we can calculate the KL divergence between two networks' output, and minimize the divergence. Note that both two stages avoid the usage of data labels. Our two-stage algorithm can be realized by gradient descent. For rounding operation, we use the Straight-through Estimator (Bengio, Léonard, and Courville 2013; Yin et al. 2019), which applies $\frac{\partial\lfloor x\rceil}{\partial x} = 1$ in backpropagation. According to the gradient calculation in (Esser et al. 2019), the gradient of $\sigma(\mathbf{x})$ can be computed by:

$$\frac{\partial \hat{\mathbf{x}}}{\partial \sigma(\mathbf{x})} = \left(\lfloor \frac{\mathbf{x}}{s_{\max}\sigma(\mathbf{x})}\rceil - \frac{\mathbf{x}}{s_{\max}\sigma(\mathbf{x})}\right) \times s_{\max}. \quad (6)$$

Note that this gradient is satisfied when the activation is not clipped. Together with the stochastic gradient descent algorithm, we are able to learn the dynamic activation step size. The whole procedure is described in Algorithm 1.

```
Algorithm 1: Dynamic Activation Step Size
Input: Full precision model, calibration dataset $\mathcal{C}$.
Parameter: initialized router block parameters $\theta$ and weight quantization step size $s_w$
Output: Quantized model.
 1: Input calibration set and get maximum activation values.
 2: Stage 1: Per-layer reconstruction
 3: for $i$ in all $l$ layers do
 4:     Store full-precision layer input and output.
 5:     for $e$ in Total Training Iterations do
 6:         Compute the output the MSE loss (Eq. (5)).
 7:         Update $\theta$ and $s_w$ by backpropagation and SGD.
 8:     end for
 9: end for
10: Stage 2: Global Tuning
11: for $e$ in Total Training Iterations do
12:     Compute the KL divergence loss.
13:     Update all layers' $\theta$ and $s_w$ by backpropagation and SGD.
14: end for
15: return Quantized model with dynamic activation step size.
```

Experiments

In this section, we verify the effectiveness and efficiency of our proposed method. All the experiments are run with PyTorch package (Paszke et al. 2019). We mainly quantize the weights and the activation on pre-trained ResNets (He et al. 2016). We use the dataset from CIFAR10, CIFAR100 (Krizhevsky, Nair, and Hinton) and ImageNet (Deng et al. 2009):

The ImageNet dataset consists of 1.2M training and 50K validation images. For creating validation dataset, we randomly sampled 1024 images as did in (Li et al. 2021b) which are processed by standard augmentation used in (He et al. 2016). We use the Pytorch official code 4 to construct ResNets, and they are initialized from the released pre-trained model. In the first stage, we use stochastic gradient descent (SGD) with the momentum of 0.9 to optimize all parameters. The batch size is set to 32. The learning rate is set to 10^{-3} followed by a cosine annealing schedule (Loshchilov and Hutter 2016) for 10k iterations of every layer's reconstruction process. No L2 regularization is imposed. In the second stage, we also optimize the whole network for 10k iterations and use the same training hyper-parameters.

CIFAR10 & CIFAR100 contain 50k training images and 10k test images for 10/100 classes. Each image is 32×32. For the calibration dataset, we randomly sample 256 images for PTQ. These images are randomly resized then cropped and randomly flipped. Other training hyper-parameters are the same with ImageNet experiments.

Table 3: CIFAR10 results on quantization, given different quantization algorithms and network architecture.

Model	Method	Bit (W/A)	Acc.
ResNet-20	Full precision	32/32	93.2
	DFQ	4/4	83.2
	DFQ + Ours	4/4	**85.9**
	BRECQ	4/4	90.4
	BRECQ + Ours	4/4	**92.2**
	BRECQ	4/2	79.9
	BRECQ + Ours	4/2	**86.3**
ResNet-110	Full precision	32/32	95.4
	DFQ	4/4	81.2
	DFQ + Ours	4/4	**87.0**
	BRECQ	4/4	93.1
	BRECQ + Ours	4/4	**94.0**

Table 4: CIFAR100 results on quantization, given different quantization algorithms and network architecture.

Model	Method	Bit (W/A)	Acc.
ResNet-20	Full precision	32/32	74.0
	DFQ	4/4	68.6
	DFQ + Ours	4/4	**70.7**
	BRECQ	4/4	72.4
	BRECQ + Ours	4/4	**72.8**
	BRECQ	4/2	61.0
	BRECQ + Ours	4/2	**69.9**
ResNet-110	Full precision	32/32	77.1
	DFQ	4/4	70.2
	DFQ + Ours	4/4	**73.8**
	BRECQ	4/4	74.7
	BRECQ + Ours	4/4	**76.0**

Results are CIFAR10 & CIFAR100

We compare our algorithm with data-free quantization (Nagel et al. 2019)[1] and block reconstruction quantization (Li et al. 2021b)[2]. For bitwidth we mainly test 4-bit weights and 4- or 2-bit activation. We summarize the results on CIFAR10 in table 3 and the results on CIFAR100 in table 4. We can find that DFQ is unable to handle 4-bit quantization, with a 7.3% accuracy drop on CIFAR10. Our method can increase the accuracy by 2.7%. It is worthwhile to note that our method improves only activation quantization. Therefore, when we apply BRECQ which adjusts the weights during calibration, we can still improve 1.8% accuracy. We also try ResNet-110, a much deeper network than ResNet-20. In this network, the activation quantization drops more accuracy. For example, DFQ achieves higher quantization accuracy on ResNet-20 but the full-precision of ResNet-20 is lower. In this case, our method exhibits the ability to recover the activation accuracy. In addition, we examine a more challenging case, the 2-bit activation quantization. Here, traditional activation quantization produces a significantly large gap. Our method, combined with BRECQ, can improve 6.4% accuracy.

[1]https://github.com/jakc4103/DFQ

[2]https://github.com/yhhhli/BRECQ

Table 5: ImageNet results on quantization, given different quantization algorithms and network architecture.

Model	Method	Bit (W/A)	Acc.
ResNet-18	Full precision	32/32	71.0
	BRECQ	4/4	69.60
	BRECQ + Ours	4/4	**70.21**
	BRECQ	4/2	59.65
	BRECQ + Ours	4/2	**64.23**
MobileNetV2	Full precision	32/32	72.49
	BRECQ	4/4	66.57
	BRECQ + Ours	4/4	**69.54**

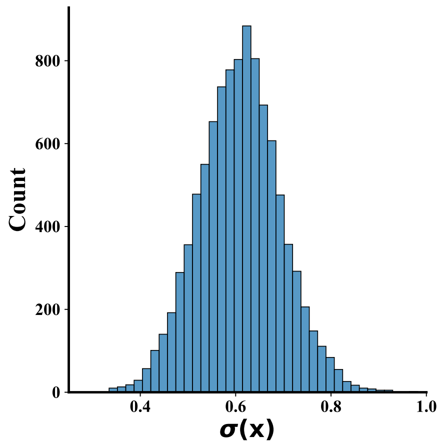

Figure 2: The dynamics of our router function.

ImageNet Results

We verify our method on both ResNet-18 as well as MobileNetV2 (Sandler et al. 2018). The results are summarized in the table 5. We mainly compare BRECQ in this section since the DFQ fails to achieve a comparable performance. For 4-bit quantization of ResNet-18, our method uplifts 0.61% accuracy, with only 0.8% accuracy gap to full precision model. Following the experiments in CIFAR10, we test 4/2 bits model, our method achieves a 4.6% accuracy boost. Finally, we verify the algorithm on the MobileNetV2, which is notoriously hard to get quantized. We show that our method can, for the first time, achieve QAT-level accuracy (as a comparison, Quantization Aware Training (QAT) only achieves 61.4%, 64.8%, 71.5% in Choi et al. (2018); Gong et al. (2019); Park and Yoo (2020), respectively.)

Dynamics Visualization

In this section, we visualize the distribution of $\sigma(\mathbf{x})$ over 10k test dataset. The result is presented in Fig. 2. We can see that the majority of the images choose 0.6 as their optimal scaling factor. However, there are also small enough values like 0.3 and large enough values like 0.9.

Conclusion

In this work, we noticed that the activation distribution would change with different input samples. Hence, we argued that keeping a fixed activation step size for PTQ maybe not be a good idea. Then the dynamic activation step size was proposed. Specifically, we appointed a router function for every layer to generate a suitable step size on basis of incoming activation and designed a two-stage training method to learn the function. Extensive experiments showed that our method consistently achieved better performance than the other SOTA.

References

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

Bewley, A.; Ge, Z.; Ott, L.; Ramos, F.; and Upcroft, B. 2016. Simple Online and Realtime Tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*.

Cai, Z.; He, X.; Sun, J.; and Vasconcelos, N. 2017. Deep learning with low precision by half-wave gaussian quantization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5918–5926.

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

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Esser, S. K.; McKinstry, J. L.; Bablani, D.; Appuswamy, R.; and Modha, D. S. 2019. Learned step size quantization. *arXiv preprint arXiv:1902.08153*.

Finkelstein, A.; Almog, U.; and Grobman, M. 2019. Fighting quantization bias with bias. *arXiv preprint arXiv:1906.03193*.

Gong, R.; Liu, X.; Jiang, S.; Li, T.; Hu, P.; Lin, J.; Yu, F.; and Yan, J. 2019. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4852–4861.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hu, J.; Shen, L.; and Sun, G. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7132–7141.

Krizhevsky, A.; Nair, V.; and Hinton, G. ???? CIFAR-10 (Canadian Institute for Advanced Research).

Krizhevsky, A.; Sutskever, I.; and Hinton, G. 2012. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in neural information processing systems*, 25(2).

Li, Y.; Deng, S.; Dong, X.; Gong, R.; and Gu, S. 2021a. A Free Lunch From ANN: Towards Efficient, Accurate Spiking Neural Networks Calibration. *arXiv preprint arXiv:2106.06984*.

Li, Y.; Dong, X.; and Wang, W. 2019. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. *arXiv preprint arXiv:1909.13144*.

Li, Y.; Gong, R.; Tan, X.; Yang, Y.; Hu, P.; Zhang, Q.; Yu, F.; Wang, W.; and Gu, S. 2021b. Brecq: Pushing the limit of post-training quantization by block reconstruction. *arXiv preprint arXiv:2102.05426*.

Li, Y.; Guo, Y.; Zhang, S.; Deng, S.; Hai, Y.; and Gu, S. 2021c. Differentiable Spike: Rethinking Gradient-Descent for Training Spiking Neural Networks. In *Thirty-Fifth Conference on Neural Information Processing Systems*.

Li, Y.; Shen, M.; Ma, J.; Ren, Y.; Zhao, M.; Zhang, Q.; Gong, R.; Yu, F.; and Yan, J. 2021d. MQBench: Towards Reproducible and Deployable Model Quantization Benchmark.

Li, Y.; Zhu, F.; Gong, R.; Shen, M.; Dong, X.; Yu, F.; Lu, S.; and Gu, S. 2021e. Mixmix: All you need for data-free compression are feature and data mixing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4410–4419.

Loshchilov, I.; and Hutter, F. 2016. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*.

Molchanov, P.; Mallya, A.; Tyree, S.; Frosio, I.; and Kautz, J. 2020. Importance Estimation for Neural Network Pruning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Nagel, M.; Baalen, M. v.; Blankevoort, T.; and Welling, M. 2019. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1325–1334.

Park, E.; and Yoo, S. 2020. Profit: A novel training method for sub-4-bit mobilenet models. In *European Conference on Computer Vision*, 430–446. Springer.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32: 8026–8037.

Polino, A.; Pascanu, R.; and Alistarh, D. 2018. Model compression via distillation and quantization. arXiv:1802.05668.

Rastegari, M.; Ordonez, V.; Redmon, J.; and Farhadi, A. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, 525–542. Springer.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2017. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 39(6): 1137–1149.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. *Springer International Publishing*.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4510–4520.

Shen, M.; Liang, F.; Gong, R.; Li, Y.; Li, C.; Lin, C.; Yu, F.; Yan, J.; and Ouyang, W. 2021. Once Quantization-Aware Training: High Performance Extremely Low-Bit Architecture Search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5340–5349.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Wikipedia. 2021. Attention (machine learning). https://en.wikipedia.org/wiki/Attention_(machine_learning). Accessed: 2021-11-11.

Yin, P.; Lyu, J.; Zhang, S.; Osher, S.; Qi, Y.; and Xin, J. 2019. Understanding straight-through estimator in training activation quantized neural nets. *arXiv preprint arXiv:1903.05662*.

Zhang, X.; He, Y.; and Jian, S. 2017. Channel Pruning for Accelerating Very Deep Neural Networks. In *IEEE International Conference on Computer Vision*.

Zhang, X.; Zhou, X.; Lin, M.; and Sun, J. 2017. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. arXiv:1707.01083.