TerViT: An Efficient Ternary Vision Transformer

Sheng Xu¹, Yanjing Li¹, Teli Ma², Bohan Zeng¹, Baochang Zhang¹, Peng Gao², Jinhu Lü¹
¹Beihang University  
²Shanghai Artificial Intelligence Laboratory

Abstract

Vision transformers (ViTs) have demonstrated great potential in various visual tasks, but suffer from expensive computational and memory cost problems when deployed on resource-constrained devices. In this paper, we introduce a ternary vision transformer (TerViT) to ternarize the weights in ViTs, which are challenged by the large loss surface gap between real-valued and ternary parameters. To address the issue, we introduce a progressive training scheme by first training 8-bit transformers and then TerViT, and achieve a better optimization than conventional methods. Furthermore, we introduce channel-wise ternarization, by partitioning each matrix to different channels, each of which is with an unique distribution and ternarization interval. We apply our methods to popular DeiT and Swin backbones, and extensive results show that we can achieve competitive performance. For example, TerViT can quantize Swin-S to 13.1MB model size while achieving above 79% Top-1 accuracy on ImageNet dataset.

1 Introduction

Inspired by the success in Natural Language Processing (NLP) tasks, transformer-based models have shown great power in various Computer Vision (CV) tasks, such as image classification [Dosovitskiy et al., 2021] and object detection [Carion et al., 2020]. Pre-trained with large-scale data, these models usually have tremendous number of parameters. For example, there are 632M parameters taking up 2528MB memory usage and 162G FLOPs in the ViT-H model, which is both memory and computation expensive during inference. This limits these models for the deployment on resource-limited platforms. Therefore, compressed transformers are urgently needed for real applications.

Substantial efforts have been made to compress and accelerate neural networks for efficient online inference. Methods include compact network design [Howard et al., 2017], network pruning [He et al., 2018], low-rank decomposition [Denil et al., 2013], quantization [Qin et al., 2020], and knowledge distillation [Romero et al., 2015]. Quantization is particularly suitable for deployment on AI chips because it reduces the bit-width of network parameters and activations for efficient inference. Prior post-training quantization (PTQ) methods [Liu et al., 2021c] on ViTs directly compute quantized parameters based on pre-trained full-precision models, which constrains model performance to a sub-optimized level without fine-tuning. Furthermore, quantizing these models based on PTQ methods to ultra-low bits (e.g., 1 or 2 bits) is inefficient and suffer from significant performance reduction.

Differently, quantization-aware training (QAT) [Liu et al., 2020] methods perform quantization during back propagation and achieve much lower performance drop and generally higher compression rate. QAT is shown to be effective for CNN models [Liu et al., 2018] for CV tasks. However, QAT methods remain unexplored for ternary quantization of vision transformers, due to the global information extracting mechanism and densely connected structure.

In this paper, we following the visualizing method in [Li et
following issues and our contributions are as from channel to channel. In a summary, we address the different distribution range of real-valued weights over, we introduce channel-wise ternarization, considering that the ternary and real-valued models. Specifically, progressively take the 8-bit model as a proxy to bridge the gap between the ternary and real-valued models. Specifically, this phenomena inspires us that higher sensitivity to binarization, which can attribute to the loss landscape of TerViTs for a better optimization. We introduce a ternary vision transformer for the quantization framework, including channel-wise weight ternarization and 8-bit activation quantization. We elaborate the ‘dead weight’ problem and visualize the two-dimensional loss landscapes to analyze how our progressive training helps conquer the zero-gradient local minima.

2 Methodology

This section describes our TerViT in details. We first overview vision transformers and then describe our quantization problem, including channel-wise weight ternarization and 8-bit activation quantization. We elaborate the ‘dead weight’ problem and visualize the two-dimensional loss landscapes to analyze how our progressive training helps conquer the zero-gradient local minima.

2.1 Preliminary

A standard transformer block includes two main modules: Self Attention (SA) module and Multi-Layer Perceptron (MLP) module. For a specific transformer layer, supposed its input is denoted as \( f_{in} \in \mathbb{R}^{n \times d} \), the corresponding query, key, and value are calculated as

\[
 f_Q = f_{in} W_Q, \quad f_K = f_{in} W_K, \quad f_V = f_{in} W_V, \tag{1}
\]

where \( W_Q, W_K, W_V \) denote matrixes to generate queries, keys and values respectively. Then, the attention scores computed by the dot product of queries and keys can be formulated as

\[
 A = \text{softmax} \left( \frac{f_Q f_K^T}{\sqrt{d}} \right). \tag{2}
\]

Finally, we calculate the weighted sum of attention weights \( A \) and \( f_V \), thus obtaining integrated features as

\[
 f_{out} = A f_V \cdot W_O. \tag{3}
\]

where \( W_O \) denotes the projection matrix. The MLP module contains two linear layers parameterized by \( W_1 \in \mathbb{R}^{d \times (\mathcal{E}d)} \), \( b_1 \in \mathbb{R}^d \) and \( W_2 \in \mathbb{R}^{(\mathcal{E}d)d} \), \( b_2 \in \mathbb{R}^d \) respectively, where \( \mathcal{E} \) is the expand ratio of MLP layers. Denote the input to MLP as \( f_{in} \in \mathbb{R}^{n \times d} \), the output is then computed as

\[
 f_{out} = \text{GeLU}(f_{in} W_1 + b_1) W_2 + b_2. \tag{4}
\]

The most computational costs of vision transformer lie in the large matrix multiplication in SA and MLP module. Following the mainstream quantization methods for CNNs [Rastegari et al., 2016], we quantize all the weights and inputs involved in matrix multiplication. For weight, we ternarize the weights \( W_Q, W_K, W_V, W_O, W_1, W_2 \) in Equ. 1 and 4 for all transformer layers. Besides these weights, we also quantize the inputs of all linear layers and matrix multiplication operations into 8-bit. Following the methods in [Zhang et al., 2020], we do not quantize the softmax operation and layer normalization, because the parameters contained in these operations are negligible and quantizing them may bring significant accuracy degradation.

Figure 2: Channel-wise weight distribution of the first MHSA layer of (a) DeiT-T, (b) DeiT-S, (c) DeiT-B, (d) Swin-T, and (e) Swin-S. We select the 1-st, 100-th, 200-th, 300-th and 400-th channels to visualize. It is obviously shown that the channel-wise distribution variance varies to a large extent, which motivates us to introduce channel-wise ternarization.


2.2 Quantization for Vision Transformers

In this section, we address key issues in quantization for ViTs, including a channel-wise weight ternarization together with a comprehensive investigation into weights distribution range, and also the activation quantization method in details.

Channel-wise weight ternarization. Following the Equs. 1 ~ 4, the input feature map has \( n \) patches and each patch has \( d \)-dim embedding channels. In the multi-head self-attention (MHSA) module, each self-attention head has 3 weight matrices, i.e., \( W_Q, W_K, W_V \in \mathbb{R}^{N_h \times d} \), where \( N_h \) is the number of attention heads. In the MHSA module, the computation process is rewritten as

\[
    A^i = \text{softmax}(f_{in}W_Q^iW_K^iT_{in}f_{in}^T), \tag{5}
\]

and

\[
    f_{out} = \text{concatenate}\{A^1W_Q^3, \ldots, A^{N_h}W_Q^{N_h}\} \cdot W_O. \tag{6}
\]

Directly quantizing each 3 matrices in MHSA as an entirety with the same quantization range can significantly degrade the accuracy, since there are \( 3 \times n \times d \) parameters in total, and the weights corresponding to each channel may lie in different range of real-valued numbers.

As shown in Fig. 2, the distribution range of real-valued weights in a MHSA module vary from channel to channel with non negligible differences. Thus, in previous work [Li et al., 2016], the layer-wise ternarization quantizes all parameters in a single layer with the same threshold \( \Delta W \) dismissing the channel-wise feature representation ability of ViTs, leading to performance degradation. Based on the discussion and observation mentioned above, we introduce a channel-wise ternarization as

\[
    T^W = \alpha \circ \text{Ternarize}(W), \tag{7}
\]

where \( \circ \) denotes the channel-wise multiplication. \( \alpha \) is the channel-wise scale factor defined by the channel-wise absolute mean (CAM) as

\[
    \alpha^j = \frac{1}{n} \sum_{k=1}^{n} |W^{k,j}|, \tag{8}
\]

and the ternarization function \( \text{Ternarize}(\cdot) \) in Eqn. 7 is further defined as

\[
    \text{Ternarize}(W^j) = \begin{cases} 
        -1 & W^j < -\Delta W^j \\
        0 & -\Delta W^j \leq W^j < \Delta W^j \\
        +1 & W^j \geq \Delta W^j
    \end{cases} \tag{9}
\]

where the real-valued weights are ternarized with a threshold \( \Delta W^j = 0.7 \frac{\|W^j\|_1}{n} \) corresponding to the distribution range of real-valued parameters.

Activation quantization. To make the most expensive matrix multiplication operation faster, following the prior works
of NLP task [Shen et al., 2020], we also quantize activations, i.e., inputs of all linear layers ($f_{in}$) and matrix multiplication ($f_Q$, $f_K$, $f_V$ and $A$) to 8 bits. There are two kinds of commonly used 8-bit quantization methods: symmetric and min-max 8-bit quantization. The quantized values of the symmetric 8-bit quantization distribute symmetrically in both sides of 0, while those of min-max 8-bit quantization distribute uniformly in a range determined by the minimum and maximum values. Specifically, for one element $x$ in the activation $f$, denote $x_{\text{max}} = \max(f)$ and $x_{\text{min}} = \min(f)$, the min-max 8-bit quantization function is

$$Q_f = \text{round}(\frac{f - x_{\text{min}}}{s}) \times s + x_{\text{min}},$$

where $s = \frac{x_{\text{max}} - x_{\text{min}}}{255}$ is the quantization scale.

### 2.3 Progressive Training

In this section, we illustrate the phenomena and statistics to show the benefit of our proposed progressive training (PT) on vision transformers.

**Reducing dead weights.** 'Dead weights' is the phenomenon that the weights in some channels are not optimized to learn meaningful representations, which are intuitively described as 'dead'. In these channels, the gradient will always stay small [Xu et al., 2021; Liu et al., 2021b], which causes insufficient training. Note that the weights refer to the real-valued latent weights, i.e., $W_Q$, $W_K$, $W_V$, $W_O$, $W_1$, and $W_2$. The magnitude of these real-valued weights are regarded as 'inertial' [Heilwegen et al., 2019], indicating how likely the corresponding ternary weights are going to change the value.

An universal measurement of the 'dead weights' is the Channel-wise Absolute Mean (CAM), which captures the average amplitude of real-valued weights within a kernel. The value of CAM is the same channel-wise scale factor as defined in Sec. 3.2. As shown in Fig. 3, we observe that the CAM of latent weights in the same networks without PT are small in their values compared with the one with PT. Another quantitative result is shown in Fig. 4, the DeiT-T and DeiT-S without PT generates 36.7% and 53.7% CAM lower than the lowest CAM of the counterparts with PT, respectively. Thus, there exists unbalanced weight fine-tuning when the model is trained without PT.

To further measure the distribution of the trained latent real-valued weight magnitude, the Standard Deviation of the Absolute Mean (SDAM) of the real-valued weight magnitude on each output channel is also calculated in Fig. 3. It is evident that the SDAM of fine-tuning with PT is lower than that of fine-tuning without PT, revealing much fewer 'dead weights' when fine-tuning the model with PT.

**Optimization.** For a better illustration, we plot a two-dimensional loss surface of the networks with two training methods which are distinguished by whether PT method is employed, following [Li et al., 2018]. As shown in the top line in Fig. 5, directly ternarizing real-valued pretrained models is challenging, for the cluttered and non-convex loss landscape. To facilitate the optimization of TerViT, our PT method first quantized the pre-trained real-valued models to 8-bit for reducing the performance gap between full-precision weights and low-precision counterparts. After training 8-bit models for specific epochs, the parameters can be smoothly transferred to 2-bit without tremendous performance drop and loss increase and achieve more stable quantized process. According to the bottom line in Fig. 5, our PT method can optimize the ternary models to obtain better results and be more robust with loss landscape closer to convex function, compared to baseline training methods.

### 3 Experiments

In this section, we evaluate the performance of the proposed TerViT model for image classification task using popular DeiT [Touvron et al., 2021] and Swin [Liu et al., 2021a] backbones. To the best of our knowledge, there is no published work done on quantization-aware training of vision transformer at this point, so we implement the baseline TWN [Li et al., 2016] methods for CNNs as described in the papers by ourselves. It is shown that the proposed method outperforms the conventional TWN method and even achieves comparable performance with significant compression ratio as the post-training quantization methods on some models. Moreover, extensive experiments of ablation study have shown that the proposed progressive training and channel-wise quantization method are beneficial for the ternary vision transformer.
Table 1: Evaluation of the setups of progressive training (PT). ‘From scratch’ denotes directly training ternary networks from scratch for 300 epochs. ‘From Real-valued’ denotes training ternary networks from real-valued pre-trained models for 300 epochs, i.e., without PT. ‘(a, b)’ denotes the step size of 8-bit pre-training and ternary fine-tuning, respectively.

| Model   | From scratch | From real-valued | Setup          |
|---------|--------------|------------------|----------------|
| TerDeiT-T | 59.3         | 65.0             | 50,250(100,200)(150,150)(200,100) |
| TerDeiT-S | 66.9         | 72.1             | 74.2           |

Table 2: The effects of different components in TerViT using DeiT-T backbone on the final accuracy. We select TWN as the baseline method. ‘PT’ denotes the progressive training. The first and last layer are quantized to 8-bit.

| Method                  | Top-1 Accuracy |
|-------------------------|----------------|
| Real-valued             | 72.2           |
| TWN (layer-wise)        | 64.4           |
| TWN + channel-wise      | 65.0           |
| TWN + PT                | 65.8           |
| TWN + channel-wise + PT (TerViT) | **66.6** |

3.1 Datasets and Implementation Details

Datasets. The experiments are carried out on the ILSVRC12 ImageNet classification dataset [Krizhevsky et al., 2012], which is more challenging than small datasets such as CIFAR [Krizhevsky et al., 2009] and MNIST [Netzer et al., 2011]. The ImageNet dataset is more challenging due to its large scale and greater diversity. There are 1000 classes and 1.2 million training images, and 50k validation images in it. In our experiments, we use the classic data augmentation method described in [Touvron et al., 2021].

Experimental settings. Given a well-trained real-valued ViT model, we first fine-tune it with 8-bit quantization on both weights and activations for 50 epochs. Then, we convert the 8-bit weight quantization to ternarization by 250 epochs. Thus, the PT method fine-tunes the models for total 300 epochs. For fair comparison, we directly fine-tune the counterparts without PT 300 epochs to validate effectiveness of our PT. The original TWN [Li et al., 2016] method is employed to validate the effectiveness of our channel-wise quantization. We quantize the first layer (patch embedding) and last layer (classification head) to 8 bits. The training hyperparameters is selected following [Touvron et al., 2021]. The AdamW optimizer is employed.

Baseline. We evaluate our quantization method on two popular vision transformer implementation: DeiT [Touvron et al., 2021] and Swin [Liu et al., 2021a]. The DeiT-T, DeiT-S, DeiT-B, Swin-T, Swin-S and Swin-S are adopted as the baseline model, whose Top-1 accuracy on ImageNet dataset are 72.2%, 79.9%, 81.8%, 81.2%, and 83.2% respectively. For a fair comparison, we utilize the official implementation of DeiT and Swin, without using other techniques like knowledge distillation.

3.2 Ablation Study

Selecting progressive training setups. Progressive training (PT) is the main contribution of our paper, we evaluate the different setup of PT, i.e., the length of the 8-bit pre-training and ternary network fine-tuning. As shown in the first 3 columns of Tab. 1, fine-tune from pre-trained model boosts the performance by about 5% ~ 6% Top-1 accuracy. As shown in the last columns, the best performance occurs when we first fine-tune the 8-bit model for 50 epochs and then train the ternary model for 250 epochs.

Evaluating the components. In this part, we evaluate every critical part of TerViT to show how we compose the novel and effective TerViT. We first introduce our baseline network, i.e., TWN [Li et al., 2016], achieving 64.4% Top-1 accuracy. As shown in Tab. 2, the introduction of channel-wise ternarization and PT improves the accuracy by 0.6% and 1.4% respectively over the baseline network, as shown in the second section of Tab. 2. By adding all the channel-wise ternarization and PT, our TerViT achieves 2.2% higher accuracy than the baseline, notably narrowing the gap between ternary DeiT-T and the real-valued counterpart.

Quantizing the first and last layer. Prior low-bit works in CNN [Liu et al., 2020] always save the first convolution layer and last fully-connected layer as real-valued due to the sensitivity. However, these real-valued layers counts a large proportion of the total model size. In our TerViT, we analyze the sensitivity of the first convolution layer (patch embedding layer) and last fully-connected layer (classification head) following [Dong et al., 2019]. As shown in the Fig. 6, the Hessian matrix eigenvalue of the patch embedding layer is obviously larger than the other layers, which indicates that the patch embedding layer is more sensitive than other layers. The classification head directly influences the output of network, thus we save it to 8-bit.

We also conduct controlled experiments on the first and last layer.
Table 4: Quantization results on ImageNet dataset. We abbreviate quantization bits used for weights as ‘W-bits’ and activations as ‘A-bits’. In particular, we first compare with the 8-bit approaches. Then we compare ours with the 2-bit baseline TWN. Our method achieves acceptable testing performance drop with significantly high compression ratio. Also note that we use 8-bit for the first and last layers.

| Model | Method | W-bit | A-bit | Model size (MB) | Compression Ratio | Top-1 Accuracy |
|-------|--------|-------|-------|-----------------|-------------------|----------------|
| DeiT-T | Real-valued | 32 | 32 | 22.7 | - | 72.2 |
| MinMax | 8 | 8 | 5.7 | 3.98× | 67.9 |
| Percentile | 8 | 8 | 5.7 | 3.98× | 69.5 |
| TWN | 2 | 8 | 1.6 | 13.35× | 64.4 |
| TerViT | 2 | 8 | 1.6 | 13.35× | 66.6(-5.6) |
| DeiT-S | Real-valued | 32 | 32 | 88.2 | - | 79.9 |
| Percentile | 8 | 8 | 22.2 | 3.97× | 74.0 |
| VT-PTQ | 2 | 8 | 22.2 | 3.97× | 77.5 |
| TWN | 2 | 8 | 6.0 | 14.70× | 70.2 |
| TerViT | 2 | 8 | 6.0 | 14.70× | 74.2(-5.7) |
| DeiT-B | Real-valued | 32 | 32 | 346.2 | - | 81.9 |
| Percentile | 8 | 8 | 90.6 | 3.82× | 77.5 |
| VT-PTQ | 2 | 8 | 90.6 | 3.82× | 81.3 |
| TWN | 2 | 8 | 22.7 | 15.25× | 72.9 |
| TerViT | 2 | 8 | 22.7 | 15.25× | 76.1(-5.8) |
| Swin-T | Real-valued | 32 | 32 | 114.2 | - | 81.4 |
| Percentile | 8 | 8 | 28.5 | 3.99× | 78.8 |
| VT-PTQ | 2 | 8 | 28.5 | 3.99× | 79.3 |
| TWN | 2 | 8 | 7.7 | 14.83× | 73.7 |
| TerViT | 2 | 8 | 7.7 | 14.83× | 77.5(-3.9) |
| Swin-S | Real-valued | 32 | 32 | 199.8 | - | 83.2 |
| Percentile | 8 | 8 | 49.9 | 3.99× | 79.2 |
| VT-PTQ | 2 | 8 | 49.9 | 3.99× | 79.6 |
| TWN | 2 | 8 | 13.1 | 15.25× | 76.1 |
| TerViT | 2 | 8 | 13.1 | 15.25× | 79.5(-3.7) |

3.3 Main Results

The experimental results are shown in Tab. 4. We compare our method with 2-bit baseline TWN [Li et al., 2016] on the same framework for the task of image classification on the ImageNet dataset. We also report the classification performance of the 8-bit quantized networks percentile [Li et al., 2019], OMSE [Choukroun et al., 2019], and VT-PTQ [Liu et al., 2021c].

We firstly evaluate the proposed method on DeiT-T, DeiT-S and DeiT-B model. For DeiT-T backbone, compared with 8-bit percentile-based method, our TerDeiT-T achieves a much larger compression ratio than 8-bit percentile. However, it is worth noting that the proposed 2-bit model significantly compresses the DeiT-T by 13.35×. The proposed method boosts the performance of TWN by 2.2% with the same architecture and bit-width, which is significant on the ImageNet dataset. For larger DeiT-S, as shown in Tab. 4, the performance of the proposed method outperforms the TWN method by 4.0%, a large margin. Compared with 8-bit methods, our method achieves significantly higher compression rate, and the performance gap is rather small. For DeiT-B, as shown in Tab. 4, the Top-1 accuracy of TWN is 72.9%. And the proposed scheme improves the performance of the ternary model by 3.2%, up to 76.1%. It is worth mentioning that the TerDeiT-B outperforms real-valued DeiT-T backbone by 3.9% with the same model size, which also proves the significance of our method.

Also, our method generates convincing results on Swin-transformers. As shown in Tab. 4, the performance of the proposed method outperforms the TWN method by 3.8% and 3.4%, a large margin. Compared with 8-bit methods, our method achieves significantly higher compression rate, and comparable performance. Note that our method achieves a small performance gap within 4% compared with the real-valued counterpart using Swin transformers, which demonstrates the significance of our TerViT.

4 Conclusion

In this paper, we introduce ternary vision transformers to improve the quantized ViTs with higher compression ratio and competitive performance. The presented TerViT introduces channel-wise ternarization and progressive training method. Notably, we also provide comprehensive and in-depth investigation in quantizing ViTs. As a result, the performance gap between TerViTs and real-valued counterparts can be significantly reduced. Extensive experiments validate the superiority of TerViT in image classification task compared with multiple mainstream backbones. Future work will focus on binarizing and more effective training methods on quantized vision transformers.
References

[Carion et al., 2020] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In Proc. of ECCV, 2020.

[Choukroun et al., 2019] Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. Low-bit quantization of neural networks for efficient inference. In Proc. of ICCV Workshops, 2019.

[Denil et al., 2013] Misha Denil, Babak Shahki, Laurent Dinh, Marc’ Aurelio Ranzato, and Nando De Freitas. Predicting parameters in deep learning. In Proc. of NIPS, 2013.

[Dong et al., 2019] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In Proc. of CVPR, 2019.

[Dosovitskiy et al., 2021] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, and Thomas Unterthiner et al. An image is worth 16x16 words: Transformers for image recognition at scale. In Proc. of ICLR, 2021.

[He et al., 2018] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. In Proc. of IJCAI, 2018.

[Helwegen et al., 2019] Koen Helwegen, James Widdicombe, Lukas Geiger, Zechun Liu, Kwang-Ting Cheng, and Roeland Nusselder. Latent weights do not exist: Re-thinking binarized neural network optimization. In Proc. of NIPS, 2019.

[Howard et al., 2017] 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. arXiv:1704.04861, 2017.

[Krizhevsky et al., 2009] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[Krizhevsky et al., 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Proc. of NIPS, 2012.

[Li et al., 2016] Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. In Proc. of NIPS Workshop, 2016.

[Li et al., 2018] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In Proc. of NIPS, 2018.

[Li et al., 2019] Rundong Li, Yan Wang, Feng Liang, Hongwei Qin, Junjie Yan, and Rui Fan. Fully quantized network for object detection. In Proc. of CVPR, 2019.

[Liu et al., 2018] Zechun Liu, Baoyuan Wu, Wenhao Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In Proc. of ECCV, 2018.